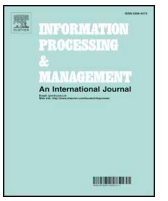
Information Processing and Management 57 (2020) 102212

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman



Incorporating stock prices and news sentiments for stock market prediction: A case of Hong Kong

Xiaodong Li*, Pangjing Wu, Wenpeng Wang

College of Computer and Information, Hohai University, Nanjing, China

ARTICLE INFO

Keywords:
Stock prediction
News sentiment analysis
Deep learning

2019 MSC:
00-01
99-00

ABSTRACT

Stock prediction via market data analysis is an attractive research topic. Both stock prices and news articles have been employed in the prediction processes. However, how to combine technical indicators from stock prices and news sentiments from textual news articles, and make the prediction model be able to learn sequential information within time series in an intelligent way, is still an unsolved problem. In this paper, we build up a stock prediction system and propose an approach that 1) represents numerical price data by technical indicators via technical analysis, and represents textual news articles by sentiment vectors via sentiment analysis, 2) setup a layered deep learning model to learn the sequential information within market snapshot series which is constructed by the technical indicators and news sentiments, 3) setup a fully connected neural network to make stock predictions. Experiments have been conducted on more than five years of Hong Kong Stock Exchange data using four different sentiment dictionaries, and results show that 1) the proposed approach outperforms the baselines in both validation and test sets using two different evaluation metrics, 2) models incorporating prices and news sentiments outperform models that only use either technical indicators or news sentiments, in both individual stock level and sector level, 3) among the four sentiment dictionaries, finance domain-specific sentiment dictionary (Loughran–McDonald Financial Dictionary) models the news sentiments better, which brings more prediction performance improvements than the other three dictionaries.

1. Introduction

Fama assumes that market information has been embedded in market price if it is an efficient market (Malkiel & Fama, 1970). However, constituents of the market information set, including their contents and types, are quite different along with the development of technologies. In early stock market, numerical stock prices on a blackboard, which need to be collected via telephone lines from different places in a country, are believed to be all the contents of the market information. Later on, besides stock prices, government and company textual announcements on newspapers are included in the market information set. With the advancement of Web technologies, more and more news articles are released on the Web, and their sentimental polarities have attracted more and more attention from investors. Many researches have verified the effectiveness of public sentiments in stock trend predictions (Bollen, Mao, & Zeng, 2011), even in Bitcoin exchange market (Yu, Kang, & Park, 2019). Although the market information set has been believed to be infinite in real markets and grows in a dynamic way, how to integrate and make use of as many contents as possible in current circumstances, and develop a helpful stock prediction approach, becomes an interesting research

* Corresponding author.
E-mail address: xiaodong.c.li@outlook.com (X. Li).

https://doi.org/10.1016/j.ipm.2020.102212
Received 30 July 2019; Received in revised form 20 January 2020; Accepted 21 January 2020
Available online 12 February 2020
0306-4573/

problem.

There are many previous research works focusing on one kind of market information, e.g. using some text mining technologies to predict stock price. Schumaker and Chen (2009) analyze word frequencies of intra-day news and make quantitative estimations for stock short-term prices. Tetlock (2007) and Tetlock, Saar-Tsechansky, and Macskassy (2008) exploit the Harvard IV-4 Dictionary in the processing of *Wall Street Journal* reports. Optimism and pessimism sentiment dimensions are extracted and quantified to help the analysis of the relationship between trading volume and price trends after the reports are released. Some other works start to integrate two kinds of market information, i.e. stock prices and financial news. Li et al. (2011) and Li, Huang, Deng, and Zhu (2014a) employ multi-kernel learning models to measure distances between instances in sub-kernel level and integrate all the sub-kernels to achieve market information fusion. However, to the best of our knowledge, there is few research work that deals with stock prices and financial news sentiments in one stock prediction system.

Another issue of some previous works is that they only use snapshots of the information set at time point t to predict targets in time point $t + 1$ or $t + \delta$. Approaches and models in this category ignore sequential relationships between consecutive snapshots before t, which is also an important component hidden in the information set. Long short-term memory (LSTM) network (Hochreiter & Schmidhuber, 1997) is a deep learning model that is able to learn the sequential information, which we believe will help the analysis and improve stock prediction performances.

To address those two issues, we propose an approach in this paper that 1) converts stock prices into technical indicators that summarize many aspects of the historical prices, 2) analyzes the sentiments of financial news articles, and constructs sentiment representations for all the news, and 3) builds a two-layer LSTM network to learn the sequential information within the series of the indicators and news sentiments, together with a fully connected network that makes final stock price trend predictions.

We design and conduct experiments based on more than five years of Hong Kong Stock Exchange data using four different sentiment dictionaries. Two baseline models are employed to compare with our proposed approach. From the experimental results, we find that,

1. The proposed approach outperforms the baselines in both validation and test sets using two different evaluation metrics.
2. The LSTM incorporating both information sources outperforms models that only use single one information source, in both individual stock level and sector level.
3. Among the four sentiment dictionaries, finance domain-specific sentiment dictionary (Loughran–McDonald Financial Dictionary) models the new sentiments better, which brings more prediction performance improvements (at most 120%) than the other three dictionaries (at most 50%).

The rest of this paper is organized as follows. Section 2 reviews literature on traditional approaches for stock prediction based on prices and news respectively, and also sentiment-based predictions using sentiment dictionaries. Section 3 presents our proposed system. Section 4 describes the design of experiments, and shows experimental results and discussions. Section 5 gives our conclusions and future work directions.

2. Related works

Stock predictions based on financial data, including traditional stock price time series and recently hot-focused news articles, have been studied by many research works. Machine learning models have been exploited as major prediction functions in these approaches developed. In recent papers, deep neural networks, especially Recurrent Neural Network (RNN) (Hopfield, 1982) and its evolution models such as LSTM (Hochreiter, Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Cho et al., 2014), have shown powerful learning ability while modeling a great mount of sequential data. In this section, related works of stock predictions using deep neural networks based on both stock prices and news articles are reviewed.

2.1. Predictions based on stock prices

Stock markets generate a lot of transaction data every day, which provides deep neural networks with a large amount of data to train and improve their prediction abilities. Zhang and Tan (2018) use historical price data to predict stocks' future return ranking via a novel stock selection model based on a deep neural network. Chen, Zhou, and Dai (2015) use a deep neural network trained by price data to predict daily volatilities of stocks in the Chinese A-share market. Their research results show that normalization is very useful for improving accuracy, and the prediction accuracy is significantly improved with the increment of price data dimensions. They also suggest running the predictions for different types of stocks separately, which can further improve accuracy. Li, Cao, and Pan (2019) build up a system that exploits a deep learning architecture to improve feature representations and adopts the extreme learning machine to predict market impacts. They conclude that deeply learned feature representations together with extreme learning machine can give better market impact prediction accuracy.

It is found by many researches that the deep neural network has better learning ability for time series data than other machine learning methods because of its memory ability. Nelson, Pereira, and De Oliveira (2017) predict future trends of stocks in the Brazilian Stock Exchange based on historical price and technical indicators. They reveal that the deep neural network has considerable gains in terms of stock prediction accuracy when compared to the other machine learning methods. Li and Liao (2018) also compare the learning ability between deep learning and classical machine learning methods in the Chinese A-share market. They train models with price data and technical indicators, and classification results show that deep neural networks have higher accuracy.

Fischer and Krauss (2018) confirm that the deep neural network performs better than memory-free classifiers in S&P 500 constituents.

2.2. Predictions based on textual news

With the advancement of word embedding in Natural Language Processing (NLP), deep neural networks can efficiently grasp information in texts by learning word vectors. Ding, Zhang, Liu, and Duan (2015) propose a method that uses a neural tensor network to train event embedding of news headlines, and uses a convolutional neural network to predict volatilities of S&P 500 and its constituent stocks. Akita, Yoshihara, Matsubara, and Uehara (2016) represent news paragraphs by vectors. They train a deep neural network to predict closing prices of 50 stocks in Tokyo Stock Exchange respectively. Hu, Liu, Bian, Liu, and Liu (2018) predict daily volatilities of stocks by training a bi-directional GRU model with news vectors and price data. Shi et al. (2019) build a system called DeepClue to bridge the text-based deep neural networks and users for stock price predictions. Schumaker and Chen (2009) build up AZFinText system to analyze word frequencies of intra-day news and make quantitative estimations for stock short-term prices. Using more data sources, such as both news texts and stock prices, has become a trend in stock predictions. Li et al. (2014a) quantify information of market news by text sentiment analysis technology. Their results show that integrated information from both market news and stock prices can improve the accuracy of prediction on stock future price returns in an intra-day trading context. They further show that using extreme learning machine for predictions can give better results (Li et al., 2016). Another one of their works (Li et al., 2015) discusses whether news article summarization is more effective for improving prediction performances than using full-length articles, and their comparative experiments show that summarization can give positive improvements.

2.3. Sentiment dictionaries

Word-frequency based textual features are considered as shallow features, while sentiments within the texts are much more important in stock predictions. Sentiment analysis is a way of deeper news analysis, where sentiment dictionaries become vital. The existing methods of sentiment dictionary construction can be categorized into two categories.

The first category is to mark text sentiments manually by linguists, such as Vader sentiment lexicon (Hutto & Gilbert, 2014), Harvard IV-4 Dictionary[1] and Loughran–McDonald Financial Dictionary (Loughran & McDonald, 2011). The Vader sentiment lexicon is built manually by 10 independent human raters. The authors also propose five rules for analyzing sentiments in social media that embodies grammatical and syntactical conventions for expressing and emphasizing sentiment intensities. The Harvard IV-4 Dictionary is built under the assistance of General Inquirer (J. Stone, C. Dunphy, Smith, & M. Ogilvie, 1966), a computer-assisted approach for the content analysis of textual data. There are more than 10,000 words and 182 sentiment dimensions in the Harvard IV-4 Dictionary. The Loughran–McDonald Financial Dictionary is a finance domain-specific dictionary and manually made by Loughran and McDonald. It contains 7 sentiment dimensions in the last version.

The other category is to manually mark text sentiments in a small dataset in the first place and then use automated extension methods to build a sentiment dictionary semi-automatically. The representative works based on this method are SentiWordNet 3.0 (Baccianella, Esuli, & Sebastiani, 2010) and SenticNet 5 (Cambria, Poria, Hazarika, & Kwok, 2018). SentiWordNet is constructed by automatically annotating all WordNet synsets according to their degrees of positivity, negativity, and neutrality. SenticNet 5 is the latest version of SenticNet, which uses implicit semantics of associated commonsense to analyze the connotation and extension of objects, behaviors, events, and characters in the real world. In the latest version, they couple sub-symbolic and symbolic AI to automatically discover conceptual primitives from text and link them to commonsense concepts and named entities by a new three-level knowledge representation.

2.4. Predictions based on news sentiments

Besides using word vectors to predict stock future trends, many studies have moved to analyze semantic information in text, e.g., mining sentiments in news articles and social media, to make predictions. Sehgal and Song, (2007) obtain sentiment tags of public moods on Yahoo message boards (a community for investors to discuss) to predict trend of stocks. It shows that the stock performances are closely correlated to their sentiments on the message boards. Nguyen, Shirai, and Velcin (2015) also analyze the sentiment tags on Yahoo message boards by Latent Dirichlet Allocation (LDA) for stock price predictions. Their results show that these sentiment tags can improve predicting performances. Xiong, Nichols, and Shen (2015) use Google domestic trends, a dataset of daily search volume related to various aspects of macroeconomics, as public mood indicators and macroeconomic factors, and use LSTM for the predictions of daily S&P 500 volatilities. In addition to these studies using existing sentiment indicators on the Internet as investors' preferences, there are some researches that use NLP methods or classification models to analyze sentiments in news articles. Wu, Su, Yu, and Chang (2012) use pointwise mutual information to mine public sentiments in news articles and propose a stock price prediction model that uses features from technical analysis and sentiments. Junqué De Fortuny, De Smedt, Martens, and Daelemans (2014) design a stock price prediction model based on text mining techniques. They preprocess news text by TF-IDF to select important words for sentiment analysis and use sentiment polarity to assert the prediction of stock price movement. Zhang, Xu, and Xue (2017) divide public sentiments into positive or negative via the Naïve Bayes method.

[1] http://www.wjh.harvard.edu/~inquirer/homecat.htm .

With the perfection of sentiment lexicons, it provides powerful tools for analyzing the sentiments of various texts. Bollen et al. (2011) use Opinion Finder lexicon and Google-Profile of Mood States lexicon to analyze the sentiments of investors on Twitter. They predict Dow Jones Industrial Average values by training a Self-Organizing Fuzzy Neural Network with the sentiments. Their results show that the accuracy of predictions can be significantly improved by the employment of specific public mood dimensions. Deng, Mitsubuchi, Shioda, Shimada, and Sakurai, (2011) use a sentiment lexicon - SentiWordNet 3.0 - to analyze the sentiments of news, and predict stock prices by training a multiple kernel learning (MKL) regression model with the fusion of sentiments and prices. Malandri, Xing, Orsenigo, Vercellis, and Cambria (2018) use LSTM to allocate portfolio based on stock returns and public moods. Li, Xie, Chen, Wang, and Deng (2014b) use the Harvard IV-4 Dictionary and the Loughran–McDonald Financial Dictionary to construct a sentiment space to represent news, and predictions are made by training classification models based on the news representations. Results show that at individual stock level, sector level and index level, models with sentiment analysis outperform the bag-of-words models. In their following work Li, Xie, Lau, Wong, and Wang (2018), they attempt to transfer the learning ability of sentiment from news-rich stock to news-poor ones.

To summarize previous approaches, predictions based on either price time series, news texts or news sentiments can be formulated as,

$$\begin{cases} f: \{P\} \rightarrow \{L\}, \\ f: \{N\} \rightarrow \{L\}, \\ f: \{S\} \rightarrow \{L\}, \end{cases} \tag{1}$$

where $\{L\}$ is a set of prediction labels, $\{P\}$, $\{N\}$, and $\{S\}$ are the information sets of past stock prices, various news texts and news sentiment representations, respectively. In contrast with previous approaches, we formulate the problem as,

$$lstm: \{P, S\} \rightarrow \{L\}, \tag{2}$$

where we consider past stock prices and news articles (with sentiments) as two sequences of events, and use LSTM to learn joint features of both event sequences to predict future price movements.

3. LSTM based stock prediction

The framework of our LSTM based stock prediction model is shown in Fig. 1. Details of each part of the model are explained in the following subsections.

3.1. Stock technical indicators

The stock price time series data consists of two dimensions of transaction data: stock daily open, high, low, close price and volume, where close price and volume are most popularly used, i.e.,

$$P = \{P_i | P_i = [Close_i, Volume_i], i = 1, ..., n\}, \tag{3}$$

where i stands for a trading day.

Technical indicators are often used by investors to analyze market states. In this paper, besides P, we also use several popular technical indicators during the preprocessing of LSTM neural network. The names and meanings of the technical indicators employed are shown in Table 1. The indicators, denoted by T and configured by different parameters, such as time lengths etc., are expected to reflect trends or fluctuations of stocks from different aspects, which provides the LSTM neural network with rich market signals. However, due to the different time length configurations, different technical indicators do not start from the same timestamp. Therefore, T_is (day i) with all indicator values are preserved while others with missing values are neglected.

3.2. News preprocessing and sentiment analysis

The *Preprocessor & Sentiments Analyzer* module in Fig. 1 transforms news articles into representations that can be adopted by sentiment analysis, by using sentiment dictionaries.

The sentiment dictionaries can support queries of words or phrases, while news articles on social media are mainly in the form of long texts. Therefore, tokenization is needed in the first step of preprocessing news articles. For each news article N, it is firstly tokenized and converted into a word vector $W = [w_1, \cdots, w_j, \cdots, w_m]$, where w_j is the j-th unique word in the article. Secondly, it checks whether the tokenized word vector contains any phrase. Using original phrases instead of tokenized words could increase recall of sentiment retrieval from the dictionaries, and thus helps improve prediction performances of the whole workflow. Therefore, word combinations are checked during the tokenization, based on keys that exist in the sentiment dictionaries. Thirdly, the module marks words with negation tags such as “not”, “no”, “neither”, etc. Finally, the module excludes stop words and lemmatizes word variations.

Each word vector W is then mapped into a sentiment space by a specific dictionary D,

$$f_D: \mathcal{W} \rightarrow \mathcal{S}, \tag{4}$$

where $\mathcal{W}$ is the word vector space and $\mathcal{S}$ is the sentiment vector space. Assume that each value in the sentiment vector is in

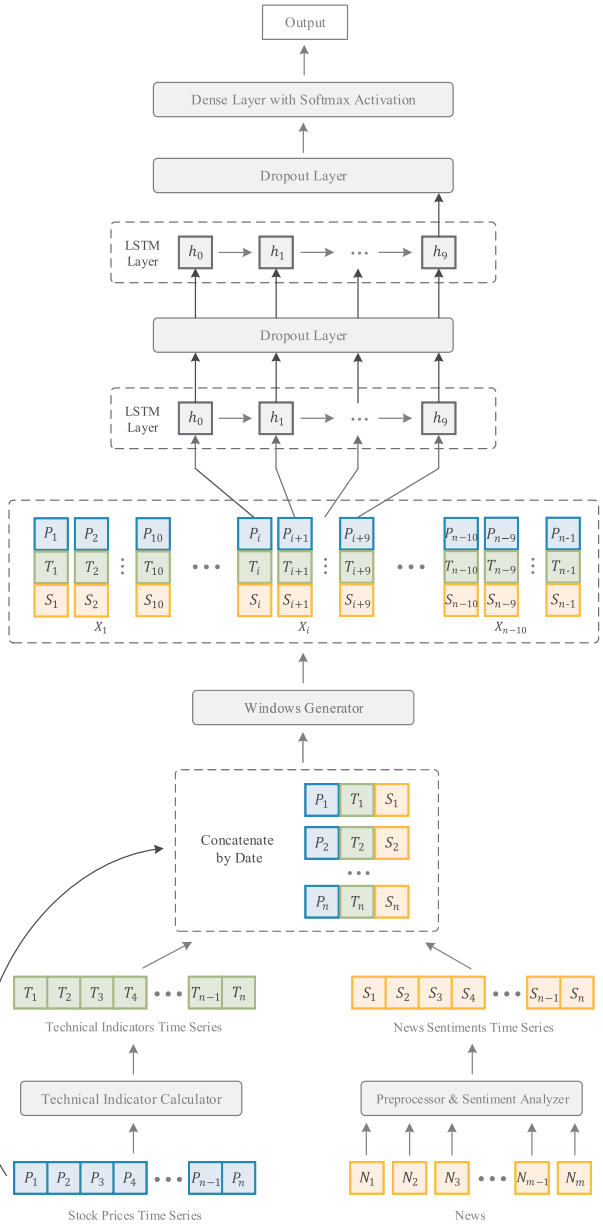


Fig. 1. LSTM based workflow of stock prediction incorporating prices and news sentiments.

Table 1
Technical indicators for stock prices.

Class	Name	Meaning
MA	MA10	10-day close price moving average
MA	MA20	20-day close price moving average
MA	MA30	30-day close price moving average
MACD	DIFF	The difference between EMA12 and EMA26
MACD	DEA	9-day exponential moving average of DIFF
MACD	MACD	Moving average convergence and divergence
RSI	RSI6	6-day relative strength index
RSI	RSI12	12-day relative strength index
RSI	RSI24	24-day relative strength index
MFI	MFI	Money flow index

accordance with linear superposition, each word w_j can be represented by a sentiment vector s_j,

$$s_j = f_D(w_j), \tag{5}$$

and each W_k can be represented by a sentiment vector S_k via summing up all s_j of each w_j, since the dimension of the s_j is fixed,

$$S_k = \sum_{w_j \in W_k} s_j. \tag{6}$$

Finally, since the number of news each day is usually more than one, daily news sentiments vector S_i (day i) is calculated by averaging S_ks in the same day. It is observed that both the T_is and the S_is are daily basis so that vectors of the same trading day can be matched by their timestamps.

3.3. Sequential snapshots generation

In this part, we integrate features from technical indicators and news sentiments, and generate sequential snapshots as inputs of the LSTM neural network. Firstly, the price vector P_i, technical indicator vector T_i and news sentiment vector S_i within each trading day are concatenated, which is denoted as Q_i. Then, the series of Q_is is divided by a rolling window with window size τ and 1-day step size, where each window is denoted by $Q_i^\tau = [Q_i, \cdots, Q_{i+\tau-1}]$.

Since each feature value's range is diverse, it causes the descending path in gradient more tortuous and the network hard to converge. For a better training of the network, we normalize each feature using Z-score on the basis of a local mean and standard deviation within each τ. The Z-score of feature vector Q_i in Q_i^τ is defined as,

$$x_i = \frac{Q_i - \mu(Q_i^\tau)}{\sigma(Q_i^\tau)}, \tag{7}$$

where x_i is the Z-score, $\mu(Q_i^\tau)$ and $\sigma(Q_i^\tau)$ represent the mean and standard deviation of Q_i^τ, and series of x_is (length of τ) forms sequential snapshots X_i^τ.

In each window, Close-to-Close stock price return (in percentage) between τ and $\tau + 1$ is used as a label y_i for X_i^τ, and (X_i^τ, y_i)s are used as the inputs of the LSTM neural network.

3.4. Long short-term memory network

In order to retrieve sequential information within the sequential snapshots, we adopt an LSTM (Gers, Schmidhuber, & Cummins, 2000) network into the proposed workflow, since the LSTM has memory ability which could help improve prediction performances.

3.4.1. Long short-term memory unit

Different from general RNNs, LSTM has several gate structures controlling passages of information. The structure of a typical LSTM unit[2] is shown in Fig. 2. Different from the original structure, we adopt a *hard sigmoid* instead of *sigmoid* function as an activator of each gate layer in our LSTM neural network. It is a linear approximation of *sigmoid* which can accelerate the calculation of the network. The tanh layer is used to activate the input and construct candidate vectors for the next step.

The learning process of the LSTM unit can be illustrated by the following example.

At time step t, the inputs of the LSTM unit consist of a snapshot x_t as well as recurrent hidden state h_{t-1} and cell state C_{t-1} of time step $t-1$ respectively. The first step in LSTM unit is to decide which information is going to be discarded from the cell state, which is executed by a forget gate i.e., the *hard sigmoid* layer marked as (a) in Fig. 2. The layer outputs a vector v_t whose values are calculated based on x_t and h_{t-1},

$$v_t = \bar{\sigma}(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{8}$$

where W_f is a weighted matrix of the forget gate, b_f is a bias vector, and value of v_t is between 0 and 1, where 1 represents keeping previous information completely while 0 represents completely forgetting it. The second step is to calculate a candidate vector $\tilde{C}_t$ of new information by the tanh layer marked as (c),

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C), \tag{9}$$

where W_C is a weighted matrix of the tanh layer, and b_C is a bias vector. This process can be considered as that we add new information to the cell state to replace the old one. Input gate, the *hardsigmoid* layer marked as (b), determines how much new information in $\tilde{C}_t$ can be stored in the cell state,

$$i_t = \bar{\sigma}(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{10}$$

where W_i is a weighted matrix of the input gate, and b_i is a bias vector. The old cell state C_{t-1} can be then updated into the new cell state C_t by,

$$C_t = v_t * C_{t-1} + i_t * \tilde{C}_t. \tag{11}$$

[2] http://colah.github.io/posts/2015-08-Understanding-LSTMs/ .

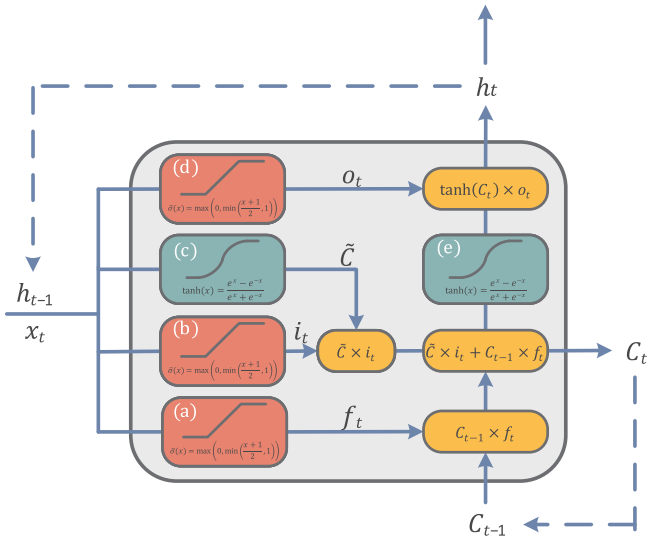


Fig. 2. Structure of an LSTM unit at time step *t*. The inputs of the LSTM unit are hidden states and cell states of time step $t-1$ as well as snapshot data of time step *t*. The outputs of the LSTM unit are cell states and hidden states of time step *t*. Dotted lines denote the transferring path of the outputs to the LSTM unit of the next time step. Layer (a) is a forget gate that decides how much information can be input to the LSTM unit. Layer (b) is an input gate that decides how much information can be learned. Layer (c) is a tanh activation layer that maps inputs into cell states. Layer (d) is an output gate that decides how much information can be output as a hidden state for the next time step. Layer (e) is a tanh activation layer that activates the information of the cell state and then adds it to the hidden state of the next time step.

During this process, the old information of prices and sentiments is actually discarded, and the new information is added. Finally, the LSTM unit needs to determine what to output, which is filtered by the new cell state C_t:

1. Output gate, the *hardsigmoid* layer marked as (d), decides which parts of the cell state are to be output,

$$o_t = \bar{\sigma}(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{12}$$

where W_o is a weighted matrix of the output gate, and b_o is a bias vector.

2. Put the new cell state through the tanh activation marked as (e) and multiply it by o_t. So that the LSTM unit only outputs the portion determined by itself for the next time step and the next layer by,

$$h_t = o_t * tanh(C_t). \tag{13}$$

The LSTM layers in Fig. 1 show the expanded form of an LSTM unit over time steps.

3.4.2. Dropout layer

Overfitting issue is manifested in the fact that models perform well with small prediction loss and high accuracy in training phases, while it is the opposite in validation or test phases. This issue is also serious in deep learning models since they need much more data for training, i.e., if a deep neural network becomes complex and there are few training samples, the model can easily overfit.

Dropout mechanism is a simple and effective regularization tool to avoid overfitting for neural networks. It sets some vectors to zero based on a given probability distribution in training. For a recurrent neural network, dropout can force the recurrent layer to perform their intermediate computations more robustly without erasing all information from the units.

Zaremba, Sutskever, and Vinyals (2014) show how to correctly apply dropout to LSTM. Their main idea is to apply the dropout operator only to the non-recurrent connections. Following their work, we add a dropout layer after each LSTM layer, as shown in Fig. 1. Except the first LSTM layer, the input of other LSTM layers becomes $dropout(h_t^{l-1})$ at time step *t*, where *l* is the layer number, $dropout(\cdot)$ is the dropout operator that sets the output hidden state to zero by a specified probability $1-p$, i.e.,

$$dropout(h_t^{l-1}) = \text{Bernoulli}(p) \cdot h_t^{l-1}, \tag{14}$$

where $\text{Bernoulli}(p)$ is a discrete probability distribution of a random variable which takes value 1 with probability p and value 0 with probability $1-p$.

3.4.3. Network structure

In this subsection, we introduce the proposed structure of the LSTM neural network, and details of each layer of the whole

network setting will be discussed in Section 4.

As shown in Fig. 1, there are two LSTM layers in our LSTM neural network. Each LSTM layer is connected by a dropout layer in order to avoid overfitting. At each time step, the first LSTM layer updates the hidden state within the layer and outputs the state values to the next layer. Note that, the second LSTM layer only outputs the last hidden state as the learning results for the next layer. After these recurrent layers, there is a fully connected layer (dense layer) containing three neurons. It is used to map the output hidden state of the LSTM layer to a category vector of length 3. Since our goal is to classify stock trends, a *softmax* function is employed as the activation function of the dense layer. So that the output of the dense layer is the probability of each category.

It is worth mentioning that we neither shuffle the input samples nor use a bidirectional LSTM layer in the network structure, in contrast, each (X_i, y_i) is inputted to the LSTM neural network in their sequential order. It is because that stock data are strictly time-ordered, which means that we cannot use future data to predict stock histories.

4. Experiments and discussions

In this section, we introduce the setup of baselines and the proposed approach and discuss the experimental results which compare models from different perspectives.

4.1. Datasets

4.1.1. Price and news data

We use stock data of Hong Kong Exchange (HKEx) to examine the effectiveness of our approach for stock predictions. The stock prices used in this experiment include daily prices from January 2003 to March 2008. Corresponding FINET news of the same period is also provided. Each news article is marked with a release time, which helps analyze the news by dates.

HKEx has thousands of stocks, but not all of them are traded actively in the market. Those inactive stocks usually have less trading volume and news reports. Our experiment mainly focuses on the Hang Seng Index (HSI) constituents, which includes stocks of bigger capitals comparing with others in the market. We select three representative stocks in each sector of HSI, i.e., Commerce, Finance, Properties, and Utilities. These stocks have higher market capitals and more news reports in their respective sectors, where detailed statistics are shown in Table 2.

4.1.2. Sentiment dictionary

In Section 2.3, we introduce some sentiment dictionaries built by manual and semi-automatic methods, and in Section 3.2 we introduce how the sentiment dictionaries are incorporated in the news analysis process. In the experiment, four sentiment dictionaries are used to calculate the sentiment vectors for the news articles, which are SenticNet 5, SentiWordNet 3.0, Vader and Loughran-McDonald Financial Dictionary 2018.

SenticNet 5 can perform fine-grained sentiment analysis, including sentiment polarity values and four dimensions of fine-grained sentiments: pleasantness, attention, sensitivity, and aptitude. Sentiment polarity values indicate positive, or, neutral, or negative sentiment orientation of words. When the value of sentiment polarity is greater than 0, it indicates that the sentiment orientation of the word is positive, when it is 0, it indicates neutral, and when it is less than 0, it indicates negative. In the experiment, we derive three sentiment polarity dimensions - positive, neutral, negative - from the original sentiment polarity. If the polarity of a word is positive, the value is added to the positive dimension; if the polarity of a word is negative, the value is added to the negative dimension; if the polarity is neutral, i.e. the value is 0, $\frac{1}{|W_i|}$ is added to the neutral dimension, where $|W_i|$ is the length of word vector of news N_i.

SentiWordNet 3.0 contains two polar sentiments of words: Positive and negative. In the sentiment analysis of news texts, we accumulate the positive and negative values of each word, and the word frequency of the neutral word is taken as the third sentiment dimension - neutral.

Vader contains four sentiment dimensions: positive, negative, neutral, and a composite score of sentiment polarity named

Table 2
Statistics on news articles of each stock.

Stock ID	Company name	Sector	Mean	Std	Max
0001.HK	CKH Holdings	Properties	3.46	4.20	56
0012.HK	Henderson Land	Properties	2.20	4.02	47
0016.HK	SHK PPT	Properties	3.03	3.54	30
0002.HK	CLP Holdings	Utilities	0.79	1.77	20
0003.HK	HK & China Gas	Utilities	0.70	1.90	32
0006.HK	Power Assert	Utilities	0.53	1.53	25
0005.HK	HSBC Holdings	Finance	**7.06**	5.45	44
0011.HK	Hang Seng Bank	Finance	1.29	2.71	29
2388.HK	BOC Hong Kong	Finance	1.42	2.75	35
0013.HK	Hutchison Whampoa	Commerce	5.26	**6.86**	70
0762.HK	China Unicom	Commerce	3.69	5.91	**82**
0941.HK	China Mobile	Commerce	6.25	5.93	61

compound. The Vader dictionary not only analyzes the sentiment of words but also provides a general interface for sentiment analysis on sentences, where it applies sentiment enhancement rules to improve sentiment analysis of sentences.

Loughran–McDonald Financial Dictionary 2018 (LMFinance) is manually made by Loughran and McDonald. It is a domain-specific sentiment dictionary for analyzing financial news, and contains 7 sentiment dimensions that are positive, negative, uncertainty, litigious, weak modal, strong modal and constraining. In the experiments, if words do not exist in those sentiments, we take their frequencies as the neutral sentiment.

4.2. Baseline setup

In the experiment, we adopt Support Vector Machine (SVM) and Multiple Kernel Learning (MKL) as the baselines.

The SVM is one of the most popular models in stock future trend classification. During the training and validation phases, parameters of SVM are tuned, and the best combination of the parameters is kept for the independent test phase. To be specific, polynomial kernel is used as the kernel of SVM and "one versus one" strategy is adopted for this multiple classification task. A grid-search and cross-validation process is adopted to optimize its performance, where the parameter grid consists of penalty $C = \{10^0, 10^{0.25}, ..., 10^3\}$ and polynomial degree $d = \{2, 3, 4, 5\}$. Accuracy is used as the metric of cross-validation.

The MKL enhances nonlinear learning performance of SVM. It combines several kernels with different characteristics to obtain the advantages of multiple kernels.

In the experiments, we adopt EasyMKL (Aiolli & Donini, 2015) as a baseline to predict stock future trends using prices, technical indicators and news sentiments. EasyMKL is done by learning a vector of combined coefficients $\boldsymbol{\eta}$ that forms the combined kernel $\mathbf{K}$ according to

$$\mathbf{K} = \sum_{r=1}^{R} \boldsymbol{\eta}_r \mathbf{K}_r, \ \boldsymbol{\eta}_r \geq 0, \tag{15}$$

where $\mathbf{K}_r$ is a basic kernel function. The problem of learning the combined kernel can be posed as a min-max problem over variables γ and $\boldsymbol{\eta}$, where γ is a probability distribution over the sets of positive and negative training samples. It maximizes the distance between positive and negative samples with a unitary norm vector $\boldsymbol{\eta}$ as the basic kernel combination vector.

$$\max_{\|\boldsymbol{\eta}\|=1} \min_{\gamma \in \Gamma} (1-\lambda) \boldsymbol{\gamma}^T \hat{\mathbf{Y}} \Big(\sum_{r}^{R} \boldsymbol{\eta}_r \hat{\mathbf{K}}_r \Big) \hat{\mathbf{Y}} \boldsymbol{\gamma} + \lambda \|\boldsymbol{\gamma}\|^2, \tag{16}$$

where $\hat{\mathbf{Y}}$ is the label vector of training set, $\hat{\mathbf{K}}_r$ is the r-th basic kernel matrix combined by training set. After maximizing the combined coefficients $\boldsymbol{\eta}$, the final optimization form of EasyMKL is denoted as,

$$\min_{\gamma \in \Gamma} (1-\lambda) \boldsymbol{\gamma}^T \hat{\mathbf{Y}} \Big(\sum_{r}^{R} \hat{\mathbf{K}}_r \Big) \hat{\mathbf{Y}} \boldsymbol{\gamma} + \lambda \|\boldsymbol{\gamma}\|^2. \tag{17}$$

In EasyMKL model, polynomial kernel is adopted as the basic kernel, which is presented as,

$$K(x_i, x_j) = (x_i \cdot x_j)^d, \quad d \in \mathbb{N}^+. \tag{18}$$

The combined kernel consists of a series of polynomial kernels, and the polynomial degree of each kernel are $d = \{2, 3, 4, 5\}$. In order to obtain optimal generalization, a cross-validation is performed on each regularization parameter $\lambda = \{0, 0.01, 0.1, 0.2, 0.5, 0.9, 1\}$. "One versus one" strategy is adopted for the multiple classification task and accuracy score is used as model cross-validation metric.

4.3. LSTM setup

In this part, we will describe the setup details of LSTM based stock prediction model.

Loss Categorical cross-entropy is adopted as a loss function to calculate prediction errors and optimize the parameters of the network during back propagation. The categorical cross-entropy is widely used in classification problems, and it is defined as,

$$L = -\frac{1}{n} \sum_{t=1}^{n} y_t \log p(\hat{y}_t), \tag{19}$$

where y_t is one-hot encoding label, $p(\hat{y}_t)$ is a vector indicating the probabilities of categorization in three classes respectively.

Optimizer Root mean square prop (RMSProp) (Tieleman, Hinton, Srivastava, & Swersky, 2012) optimizer is usually a good choice for recurrent neural networks. In our experiment, it is used as the optimizer of the LSTM neural network. The initial learning rate of the optimizer is set to 0.001, the same as the recommended default setting of the optimizer. Larger initial learning rate will speed up the optimization in the early stage of training and make the model closer to the optimal solution. However, it can cause the output of the loss function fluctuating around the minimum value, and it will be difficult to achieve optimality in the later stage. Therefore, we establish a learning rate reduction mechanism to automatically adjust the learning rate according to the change of loss. When the loss does not decrease in 5 consecutive iterations, the learning rate is reduced to one tenth of itself, so that the model can be closer to the optimal solution.

Batch size and epoch The batch size of the LSTM neural network is set to 32 and the max number of training epochs is 500. When

the training loss cannot be optimized or it is getting bigger and bigger after several rounds of iterations, the subsequent training of the model is not necessary. Early-stop mechanisms can stop the training process automatically and save the training time of neural networks. We apply an early-stop mechanism to the training which automatically stops the training when the loss no longer reduces in 10 epochs, and we save the model which has the least loss. In order to avoid the impact of the imbalanced class sizes on training, the loss weight of each class is automatically adjusted based on the class sizes.

Grid-search on hyperparameters In order to find the optimal hyperparameters of the LSTM neural network, we use grid-search together with a cross-validation method to search for the optimal hyperparameters on the parameter grid. The grid consists of 5 hyperparameters, each of which contains several candidate values of the hyperparameters:

- Neurons = {20, 50, 100, 200}: The dimension of hidden state.
- Forget bias = {True, False}: The bias of forget gate. If it is true, the initial bias vector is filled with ones, which means to remember all information first. If it is false, the initial bias vector is filled with zeros. According to Jozefowicz, Zaremba, and Sutskever (2015), this parameter can affect the training of the LSTM neural network and it is recommended to set it as true.
- Dropout rate = {0.2, 0.35, 0.5}: The dropout rate of dropout layers.
- Kernel initializer = {RandomNormal, RandomUniform, glorot uniform, glorot normal}: Method for initializing weight matrix of input features (except hidden state).
- Kernel regularizer = {None, l2}: Regular function applied to the weight matrix of kernel.

Early-stop mechanism Excessive training of neural networks will make model overfitting to in-sample data and have little generalization ability on out-of-sample data. In order to make the model have a good generalization ability. We add an early-stop mechanism to the training process. When the losses in the validation do not decrease in 10 consecutive epochs, the training is stopped and the best model is saved, and the neural network is expected to have the best generalization ability.

We select CKH Holdings (0001.HK) as an example to show the changes of the losses during training and validation. According to the training settings mentioned above, when the losses in the validation set no longer decrease, the learning process stops automatically, which is shown in Fig. 3.

4.4. Experiment setup

Datasets split Since calculations of technical indicators with different time lengths need different numbers of data points, values of some indicators at the beginning of the series are empty (which are set to null), and the snapshots with full values start from March 2003.

We divide the datasets into three parts by their dates. The data from March 2003 to March 2007 (about 80% of the total data) is used as the training set, and cross-validation is conducted on the training set to select optimal hyperparameters for models. For the LSTM model, the data from March 2007 to September 2007 (about 10% of the total data) is used as a set for early-stop, and the data from September 2007 to March 2008 (last 10% of the total data) is used as the test set to test the out-of-sample performances of the LSTM model. For the baselines, the data from March 2007 to March 2008 is completely used as the test set.

Cross-validation setup As the look-into-future concern stated in Section 3.4.3, Ratto et al. (2018) propose a time split cross-validation for stock prediction. We adopt 3-fold time split cross-validation to train and validate models on each node in the parameters grid. The time series split method uses the first k splits for training in each iteration, and the $k + 1$ split as the validation set.

Fig. 3. The training process of CKH Holdings (0001.HK) using SenticNet sentiment dictionary. The left sub-graph shows the loss (categorical cross-entropy) curves in the training set and the validation set, where the loss of the validation set reaches the minimum at about the 50th epoch but sharply increases in the subsequent training iterations. In order to get the best out-of-sample performance, the training process stops at the 50th epoch. The right sub-graph shows the accuracy curves in the training set and the validation set, where the accuracy of the validation set increases slightly in the first 30 epochs, but the accuracy gradually decreases after the 100th epoch.

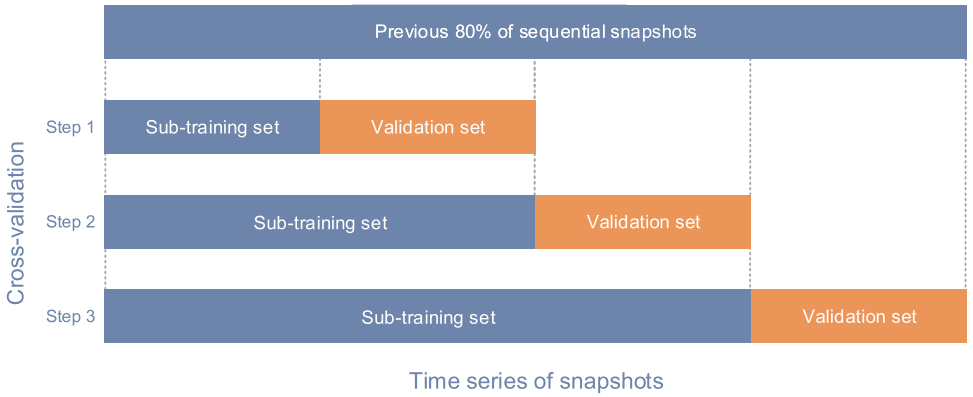


Fig. 4. Cross-validation process of 3-fold time series splitting method.

The cross-validation process is illustrated in Fig. 4.

Labels The purpose of our research is to use stock prices and news sentiments to predict Close-to-Close stock price return rate, which is defined as,

$$r_t = \frac{close(t) - close(t-1)}{close(t-1)} \times 100\%, \tag{20}$$

where t is the date of a trading day. The statistical characteristics of stock price return rates are shown in Table 3. The average return rate of these stocks is approximate to 0, and most of their price return distributions are lightly positive skewed (skewness $>$ 0) with fat tails (kurtosis $>$ 3).

In order to determine labels of each stock, we sort the close-to-close stock price return in an ascending order. Following by researches of Baltas, Jessop, Jones, and Lancetti (2014) and Holcroft et al. (2016), we use 25% quantile p_{25} and 75% quantile p_{75} of the close-to-close stock price return as thresholds for label determination. If the return is in the bottom 25%, the label is set as *fall* (class 0); if it is in the top 25%, the label is set as *rise* (class 2); the middle 50% is labeled as *horizontal* (class 1). For LSTM model, one-hot encoding is used to encode the labels, so that the labels can have the same distance and the classification error can be measured more accurately. To formulate, the labels of close-to-close stock price return of day t is defined as,

$$y_t = \begin{cases} [1, 0, 0], & r_t < p_{25}, \\ [0, 1, 0], & p_{25} \le r_t \le p_{75}, \\ [0, 0, 1], & r_t > p_{75}. \end{cases} \tag{21}$$

Metrics In the experiment, accuracy and weighted macro F1-score are adopted to evaluate the performances of each model. We let n denotes the total number of samples, N_c denotes the number of samples whose true label is c. These metrics are defined as follows,

$$accuracy = \sum_{c \in y} \frac{n_{cc}}{N}, \tag{22}$$

$$F1 = \sum_{c \in y} \frac{N_c}{N} \frac{2P_c R_c}{P_c + R_c}, \tag{23}$$

where $y = \{[1, 0, 0], [0, 1, 0], [0, 0, 1]\}$, N_c is the total number of samples in class c, N is the total number of samples, n_{ij} is the number

Table 3
Statistical characteristics of stock price return rate.

Stock	Mean	Standard deviation	Skewness	Kurtosis
0001.HK	0.074	1.843	0.299	2.650
0012.HK	0.088	1.976	0.302	3.414
0016.HK	0.093	1.878	0.394	2.915
0002.HK	0.061	0.983	0.262	4.323
0003.HK	0.076	1.383	0.077	5.369
0006.HK	0.047	1.081	-0.060	4.257
0005.HK	0.033	1.022	0.451	16.681
0011.HK	0.043	1.185	0.943	11.124
2388.HK	0.073	1.614	0.194	8.894
0013.HK	0.043	1.447	0.391	2.900
0762.HK	0.032	2.804	0.242	2.335
0941.HK	0.168	2.001	0.196	1.430

Table 4
Data split and metrics of training, cross-validation and test.

Model	Metrics of training and cross-validation	Metrics of test
LSTM	cross-entropy (early-stop), accuracy	cross-entropy (early-stop), accuracy, F1
MKL	accuracy	accuracy, F1
SVM	accuracy	accuracy, F1

of samples whose true label is i and predicted label is j. P_c and R_c in Eq. (23) are *precision* and *recall* presented as,

$$P_c = \frac{n_{cc}}{n_{cc} + \sum_{p \neq c} n_{pc}}, \tag{24}$$

$$R_c = \frac{n_{cc}}{n_{cc} + \sum_{p \neq c} n_{cp}}. \tag{25}$$

To summarize, the metrics of training, cross-validation and test phases are concluded in Table 4.

4.5. Experimental results

The accuracy scores of the baselines and the LSTM in cross-validation are shown in Table 5, where numbers in bold font represent the highest scores among different models for each stock using different sentiment dictionaries respectively. It can be observed that the LSTM based on different sentiment dictionaries outperforms the baselines on almost all of the stocks in cross-validation. SVM only gets the highest accuracy score in 2388.HK based on Vader, and MKL only gets the highest accuracy score in 0013.HK based on Vader.

The performances of the baselines and LSTM on test set are shown in Table 6 for LMFinance, Table 7 for SenticNet, Table 8 for SentiWordNet and Table 9 for Vader, respectively, where numbers in bold font represent the highest accuracy score, and numbers underlined represent the highest F1 score of each stock among different models.

Based on LMFinance (Table 6), the LSTM performs better than the baselines on 7 stocks in accuracy and 8 stocks in F1. The MKL performs the best on 6 stocks in accuracy and 1 stock in F1. The SVM only performs the best on 3 stocks in F1. To summarize, the LSTM performs the best, and the MKL outperforms the SVM (15:7:3). Based on SenticNet (Table 7), the LSTM performs better than the baselines on 6 stocks in accuracy and 5 stocks in F1. The MKL performs the best on 4 stocks in accuracy and 1 stock in F1. The SVM performs the best on 2 stocks in accuracy and 6 stocks in F1. To summarize, the LSTM performs the best, and the MKL underperforms the SVM (11:5:8). Based on SentiWordNet (Table 8), the LSTM performs better than the baselines on 6 stocks in accuracy and 7 stocks in F1. The MKL performs the best on 7 stocks in accuracy. The SVM only performs the best on 1 stock in accuracy and 5 stocks in F1. To summarize, the LSTM performs the best, and the MKL outperforms the SVM (13:7:6). Based on Vader (Table 9), the LSTM

Table 5
Accuracy of models on validation sets.

Stock	LMFinance			SenticNet			SentiWordNet			Vader		
	SVM	MKL	LSTM	SVM	MKL	LSTM	SVM	MKL	LSTM	SVM	MKL	LSTM
Properties												
0001.HK	0.706	0.798	**0.821**	0.423	0.512	**0.582**	0.406	0.473	**0.518**	0.432	0.454	**0.479**
0012.HK	0.782	0.799	**0.828**	0.410	0.522	**0.554**	0.410	0.489	**0.509**	0.409	0.454	**0.493**
0016.HK	0.777	0.781	**0.817**	0.426	0.463	**0.529**	0.428	0.443	**0.501**	0.403	0.428	**0.465**
Utilities												
0002.HK	0.713	0.767	**0.820**	0.420	0.448	**0.566**	0.431	0.442	**0.521**	0.426	0.407	**0.495**
0003.HK	0.767	0.783	**0.805**	0.400	0.478	**0.553**	0.437	0.435	**0.509**	0.402	0.403	**0.444**
0006.HK	0.786	0.784	**0.804**	0.439	0.479	**0.527**	0.447	0.463	**0.514**	0.459	0.431	**0.483**
Finance												
0005.HK	0.755	0.795	**0.825**	0.409	0.446	**0.506**	0.402	0.408	**0.473**	0.395	0.411	**0.434**
0011.HK	0.774	0.798	**0.812**	0.429	0.510	**0.561**	0.423	0.473	**0.517**	0.425	0.423	**0.499**
2388.HK	0.731	0.828	**0.835**	0.443	0.496	**0.521**	0.448	0.477	**0.493**	**0.499**	0.442	0.474
Commerce												
0013.HK	0.716	0.790	**0.822**	0.409	0.451	**0.541**	0.409	0.424	**0.489**	0.401	**0.438**	0.434
0762.HK	0.746	0.826	**0.838**	0.409	0.416	**0.513**	0.408	0.415	**0.453**	0.430	0.427	**0.441**
0941.HK	0.724	0.805	**0.820**	0.433	0.486	**0.518**	0.451	0.451	**0.505**	0.421	0.402	**0.455**

Table 6
Accuracy and F1 scores of LMFinance based models on test set.

Stock	SVM		MKL		LSTM	
	Accuracy	F1	Accuracy	F1	Accuracy	F1
Properties						
0001.HK	0.350	0.321	**0.388**	0.207	0.376	0.270
0012.HK	0.318	0.289	**0.561**	0.496	0.554	0.513
0016.HK	0.338	0.329	**0.557**	0.485	0.550	0.518
Utilities						
0002.HK	0.315	0.274	**0.336**	0.218	**0.336**	0.260
0003.HK	0.391	0.383	0.579	0.522	**0.584**	0.537
0006.HK	0.369	0.344	0.566	0.527	**0.568**	0.539
Finance						
0005.HK	0.400	0.381	**0.617**	0.550	0.601	0.579
0011.HK	0.249	0.188	0.501	0.431	**0.508**	0.455
2388.HK	0.271	0.267	0.371	0.224	**0.380**	0.279
Commerce						
0013.HK	0.355	0.325	0.434	0.332	**0.489**	0.327
0762.HK	0.333	0.320	0.465	0.339	**0.481**	0.370
0941.HK	0.369	0.344	**0.421**	0.256	0.419	0.304

Table 7
Accuracy and F1 scores of SenticNet based models on test set.

Stock	SVM		MKL		LSTM	
	Accuracy	F1	Accuracy	F1	Accuracy	F1
Properties						
0001.HK	**0.425**	0.422	0.422	0.264	0.422	0.298
0012.HK	0.274	0.273	**0.347**	0.198	0.339	0.268
0016.HK	0.296	0.284	**0.348**	0.269	0.336	0.180
Utilities						
0002.HK	**0.344**	0.331	0.296	0.173	0.312	0.190
0003.HK	0.232	0.132	**0.348**	0.200	0.328	0.257
0006.HK	0.328	0.312	0.348	0.224	**0.360**	0.282
Finance						
0005.HK	0.232	0.110	0.412	0.349	**0.424**	0.322
0011.HK	0.152	0.040	**0.248**	0.107	0.236	0.159
2388.HK	0.199	0.066	0.325	0.209	**0.347**	0.240
Commerce						
0013.HK	0.344	0.345	0.400	0.261	**0.408**	0.288
0762.HK	0.283	0.256	0.338	0.292	**0.418**	0.310
0941.HK	0.312	0.244	0.444	0.316	**0.448**	0.346

performs better than the baselines on 8 stocks in accuracy and 2 stocks in F1. The MKL performs the best on 4 stocks in accuracy and 5 in F1. The SVM performs the best on 5 stocks in F1. To summarize, the LSTM performs the best, and the MKL outperforms the SVM (10:9:5). From the performances of models based on different sentiment dictionaries, it can be concluded that the LSTM performs the best in most stocks, and the MKL plays the second best which is better than the SVM.

By comparing the results sector by sector, in most cases, the LSTM also performs better than the MKL and the SVM. Table 6 shows that the LSTM based on LMFinance performs better than the baselines in most stocks of Utilities (2 in F1 and 3 in accuracy), Finance (3 in F1 and 2 in accuracy) and Commerce (2 in F1 and 2 in accuracy), but performs almost the same as the MKL in Properties. Table 7 shows that the LSTM based on SenticNet performs better than the baselines in most stocks of Finance (2 in accuracy and 2 in F1) and

Table 8
Accuracy and F1 scores of SentiWordNet based models on test set.

Stock	SVM		MKL		LSTM	
	Accuracy	F1	Accuracy	F1	Accuracy	F1
Properties						
0001.HK	0.328	0.341	0.320	0.239	**0.340**	0.266
0012.HK	0.347	0.355	0.347	0.312	**0.379**	0.289
0016.HK	0.280	0.282	**0.356**	0.224	0.316	0.322
Utilities						
0002.HK	**0.336**	0.333	0.316	0.214	0.324	0.229
0003.HK	0.240	0.138	**0.360**	0.241	0.344	0.293
0006.HK	0.304	0.227	**0.344**	0.245	**0.344**	0.273
Finance						
0005.HK	0.288	0.284	**0.428**	0.254	0.380	0.378
0011.HK	0.152	0.040	**0.288**	0.145	0.252	0.255
2388.HK	0.257	0.263	**0.336**	0.260	0.332	0.279
Commerce						
0013.HK	0.376	0.366	0.376	0.306	**0.408**	0.308
0762.HK	0.369	0.354	0.375	0.335	**0.399**	0.341
0941.HK	0.344	0.291	**0.432**	0.341	**0.432**	0.374

Table 9
Accuracy and F1 scores of Vader based models on test set.

Stock	SVM		MKL		LSTM	
	Accuracy	F1	Accuracy	F1	Accuracy	F1
Properties						
0001.HK	0.264	0.249	**0.348**	0.331	0.344	0.329
0012.HK	0.290	0.225	0.331	0.309	**0.359**	0.300
0016.HK	0.328	0.339	0.332	0.314	**0.356**	0.318
Utilities						
0002.HK	0.336	0.314	0.316	0.283	**0.356**	0.254
0003.HK	0.304	0.322	**0.372**	0.291	0.352	0.334
0006.HK	0.272	0.227	0.284	0.297	**0.360**	0.227
Finance						
0005.HK	0.360	0.313	**0.372**	0.294	0.344	0.350
0011.HK	0.304	0.315	**0.324**	0.260	0.304	0.304
2388.HK	0.287	0.219	0.295	0.247	**0.339**	0.233
Commerce						
0013.HK	0.384	0.361	0.392	0.353	**0.412**	0.348
0762.HK	0.348	0.329	0.330	0.320	**0.370**	0.306
0941.HK	0.320	0.293	0.420	0.419	**0.448**	0.400

Commerce (3 in accuracy and 2 in F1), but the SVM and the MKL perform slightly better than the LSTM in Properties and Utilities. Table 8 shows that the LSTM based on SentiWordNet performs better than the baselines in most stocks of Properties (2 in accuracy and 1 in F1), Utilities (1 in accuracy and 2 in F1) and Commerce (3 in accuracy and 1 in F1), but performs almost the same as the MKL in Finance. Table 9 shows that the LSTM performs better than the baselines in most stocks of Utilities (2 in accuracy and 1 in F1) and Commerce (3 in accuracy), but performs almost the same as the MKL in Properties and Finance.

In order to compare the average performances of each model at sector level, we calculate average accuracy and F1 by sector and present them in Table 10. In the results based on LMFinance, the LSTM has the highest accuracy in 3 sectors and the highest F1 in 4 sectors. In the results based on SenticNet, the LSTM has the highest accuracy in 3 sectors and the highest F1 in 2 sectors. In the results

Table 10
Sector results of models on test set.

Sector	SVM		MKL		LSTM	
	Accuracy	F1	Accuracy	F1	Accuracy	F1
LMFinance						
Properties	0.335	0.313	**0.502**	0.396	0.493	0.434
Utilities	0.358	0.334	0.494	0.422	**0.496**	0.446
Finance	0.307	0.278	**0.496**	0.401	**0.496**	0.437
Commerce	0.352	0.330	0.440	0.309	**0.463**	0.334
SenticNet						
Properties	0.332	0.326	**0.372**	0.214	0.366	0.278
Utilities	0.301	0.259	0.331	0.199	**0.333**	0.243
Finance	0.194	0.072	0.328	0.222	**0.336**	0.240
Commerce	0.313	0.282	0.394	0.289	**0.425**	0.315
SentiWordNet						
Properties	0.318	0.326	0.341	0.258	**0.345**	0.292
Utilities	0.293	0.233	**0.340**	0.233	0.337	0.265
Finance	0.232	0.196	**0.351**	0.220	0.321	0.304
Commerce	0.363	0.337	0.394	0.327	**0.413**	0.341
Vader						
Properties	0.294	0.271	0.337	0.318	**0.353**	0.316
Utilities	0.304	0.288	0.324	0.291	**0.356**	0.272
Finance	0.317	0.282	**0.330**	0.267	0.329	0.295
Commerce	0.351	0.328	0.381	0.364	**0.410**	0.352

based on SentiWordNet, the LSTM has the highest accuracy in 2 sectors and the highest F1 in 3 sectors, while the MKL gets the highest accuracy in 2 sectors. In the results based on Vader, the LSTM has the highest accuracy in 3 sectors and the highest F1 in 1 sector, while the MKL gets the highest F1 in 3 sectors. In general, it can be concluded that based on different sentiment dictionaries, the LSTM outperforms the baselines in most sectors.

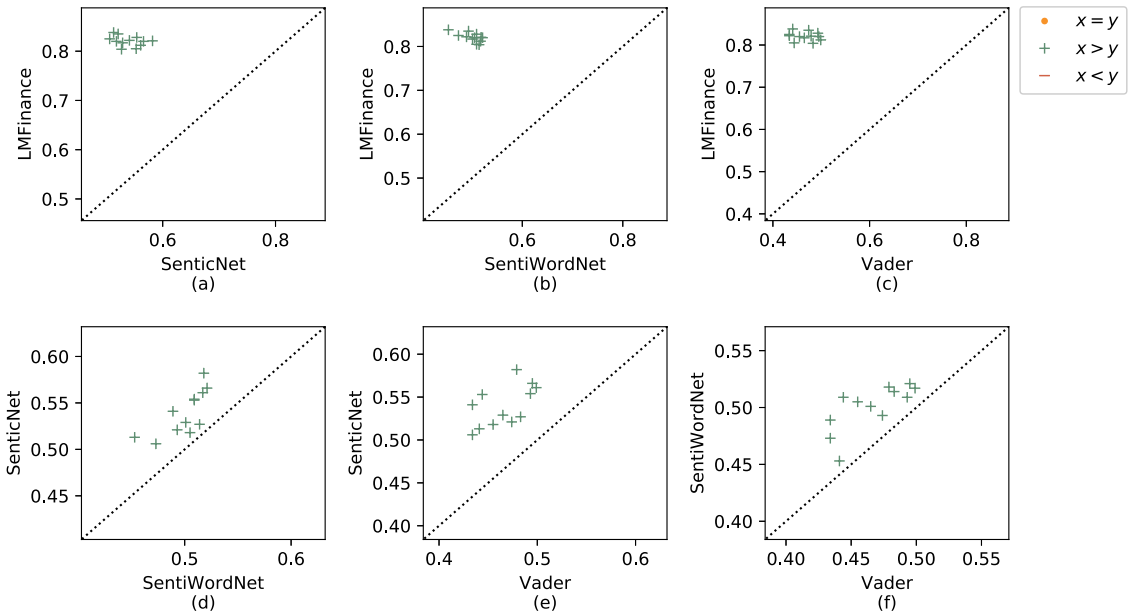


Fig. 5. Accuracy of LSTM based on different sentiment dictionaries on validation set. x (or y) axis denotes the accuracy score of the LSTM based on x (or y) dictionary. Green '+'s above the line mean that the LSTM based on y dictionary performs better than the one based on x dictionary; Orange '.'s on the black dotted line mean that the LSTM performs equally based on the two dictionaries; Red '-'s below the line mean that the LSTM based on x dictionary performs better than the one on y dictionary.

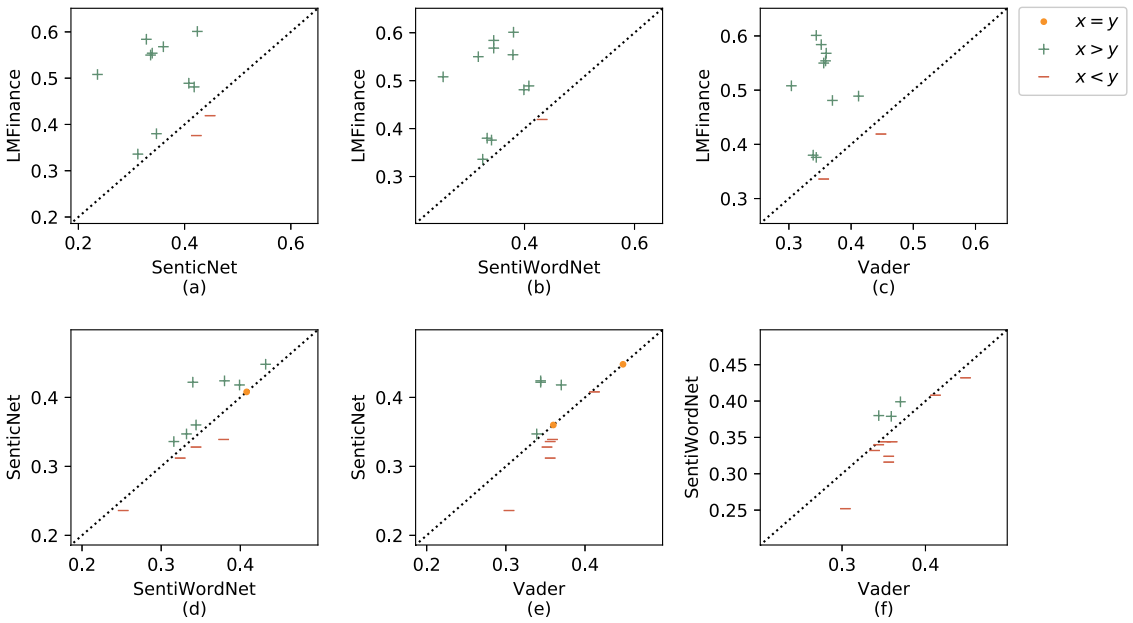


Fig. 6. Accuracy of LSTM based on different sentiment dictionaries on test set. x (or y) axis denotes the accuracy score of the LSTM based on x (or y) dictionary. Green '+'s above the line mean that the LSTM based on y dictionary performs better than the one based on x dictionary; Orange '.'s on the black dotted line mean that the LSTM performs equally based on the two dictionaries; Red '-'s below the line mean that the LSTM based on x dictionary performs better than the one on y dictionary.

In order to compare the performances of each sentiment dictionary, we visualize the accuracy scores of the LSTM based on different sentiment dictionaries. The accuracy on validation is presented in Fig. 5. The sub-graphs (a), (b) and (c) show that the LMFinance performs better than the SenticNet, the SentiWordNet and the Vader on all stocks, and sub-graphs (d) and (e) show that the SenticNet outperforms the SentiWordNet and the Vader, and sub-graph (f) shows that the SentiWordNet outperforms the Vader.

Accuracy of the test set is presented in Fig. 6. The sub-graphs (a), (b) and (c) show that the LMFinance performs better than the SenticNet, the SentiWordNet and the Vader on most stocks, and sub-graph (d) shows that the SenticNet performs better than the SentiWordNet in 7 stocks, and sub-graph (e) shows that the Vader outperforms the SenticNet in 6 stocks, and sub-graph (f) shows that the Vader outperforms SentiWordNet in 9 stocks.

4.6. Discussions on experimental results

4.6.1. The effectiveness of news sentiments

To investigate how much improvement can the news information bring to the stock predictions and what are the differences among the sentiment dictionaries while they improve the prediction accuracy, we use Δ_{news} to represent the relative improvement of accuracy between the results of using both price data and news sentiments and using only the price data. Δ_{news} is calculated by,

$$\Delta_{news} = \frac{acc_{p,s} - acc_p}{acc_p}, \tag{26}$$

where $acc_{p,s}$ is the accuracy with both information sources and acc_p the accuracy with only price information. The improvements of different sentiment dictionaries are presented in Fig. 7. It can be observed that using both price data and news sentiments to predict stocks outperforms the one only using the price data. If we further compare the improvements between different dictionaries, the improvements brought by general dictionaries (i.e. SenticNet, SentiWordNet, Vader) are almost the same, where most of the improvements are between 10% to 40%. Domain-specific dictionary (i.e. LMFinance) can improve the prediction accuracy much more, where the Δ_{news} of the LMFinance is significantly higher than other dictionaries and some of the improvements are over 80%.

4.6.2. The effectiveness of price data

In order to analyze whether models with multiple information sources are better than the models only with news data in the stock prediction, we adopt Δ_{price} to evaluate the relative improvements of accuracy between the LSTM model using both prices and news sentiments and the LSTM model only using news sentiments on the test set. Δ_{price} is calculated by,

$$\Delta_{price} = \frac{acc_{p,s} - acc_s}{acc_s}, \tag{27}$$

where acc_s is the accuracy with only news information. The results of Δ_{price} are show in Table 11. It can be observed that using

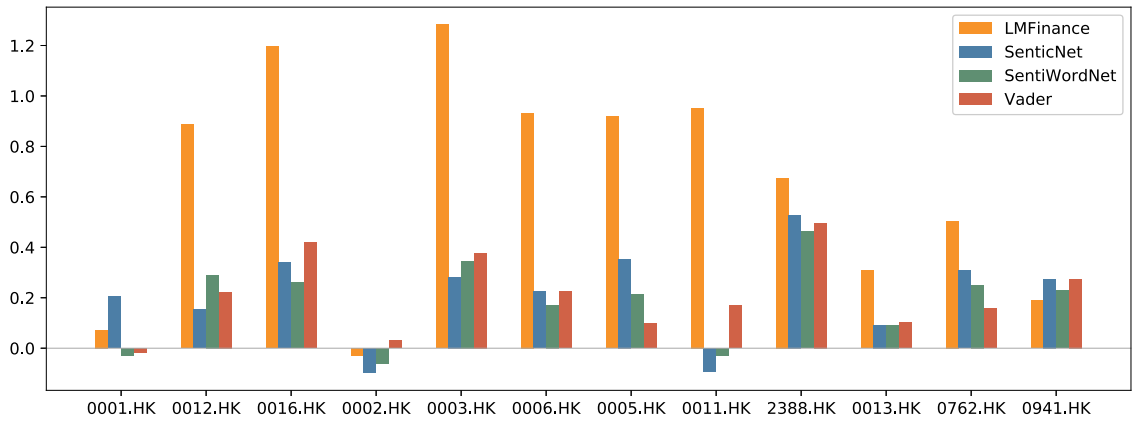


Fig. 7. The Δ_{news} of each stock, where x axis represents different stocks.

Table 11
The prediction accuracy of the LSTM neural network based on different data sources.

Stock	LMFinance			SenticNet			SentiWordNet			Vader		
	News	All	Δ_{price}	News	All	Δ_{price}	News	All	Δ_{price}	News	All	Δ_{price}
Properties												
0001.HK	0.325	0.376	0.157	0.487	0.422	-0.133	0.208	0.340	0.635	0.264	0.344	0.303
0012.HK	0.332	0.554	0.669	0.202	0.339	0.678	0.250	0.379	0.516	0.202	0.359	0.777
0016.HK	0.298	0.550	0.846	0.288	0.336	0.167	0.208	0.316	0.519	0.208	0.356	0.712
Utilities												
0002.HK	0.345	0.336	-0.026	0.336	0.312	-0.071	0.336	0.324	-0.036	0.312	0.356	0.141
0003.HK	0.329	0.584	0.775	0.264	0.328	0.242	0.288	0.344	0.194	0.288	0.352	0.222
0006.HK	0.373	0.568	0.523	0.288	0.360	0.250	0.288	0.344	0.194	0.288	0.360	0.250
Finance												
0005.HK	0.360	0.601	0.669	0.328	0.424	0.293	0.344	0.380	0.105	0.224	0.344	0.536
0011.HK	0.316	0.508	0.608	0.200	0.236	0.180	0.152	0.252	0.658	0.152	0.304	1.000
2388.HK	0.330	0.380	0.152	0.199	0.347	0.744	0.199	0.332	0.668	0.199	0.339	0.704
Commerce												
0013.HK	0.320	0.489	0.528	0.296	0.408	0.378	0.296	0.408	0.378	0.296	0.412	0.392
0762.HK	0.329	0.481	0.462	0.342	0.418	0.222	0.316	0.399	0.263	0.367	0.370	0.008
0941.HK	0.298	0.419	0.406	0.416	0.448	0.077	0.304	0.432	0.421	0.344	0.448	0.302

multiple information sources is better than using only news information source for almost all the sentiment dictionaries.

5. Conclusions

We study an attractive research topic in financial market, which is how to combine technical indicators from stock prices and news sentiments from textual news articles, and make prediction models be able to learn time series sequential information in an intelligent way. To address the problem, we build up a stock prediction system and propose an approach that 1) converts historical prices into technical indicators that summarize aspects of the price information, and models news sentiments by using different sentiment dictionaries and represents textual news articles by sentiment vectors, 2) constructs a two-layer LSTM neural network to learn the sequential information within market snapshots series, 3) constructs a fully connected neural network to make stock predictions. Experiments have been conducted on more than five years of real Hong Kong stock market data using four different sentiment dictionaries. Two baseline models, i.e., MKL and SVM, are employed as benchmarks to compare the performances of our proposed approach. It is found from the results that,

1. Based on both information sources, the LSTM outperforms the MKL and the SVM in both prediction accuracy and F1 score.
2. The LSTM incorporating both information sources outperforms the models that only use either technical indicators or news sentiments, in both individual stock level and sector level.

3. Among the four sentiment dictionaries, finance domain-specific sentiment dictionary (Loughran–McDonald Financial Dictionary) models the new sentiments better, which brings at most 120% prediction performance improvement, compared with the other three dictionaries (at most 50%).

In the future work, we need to change full-text sentiment analysis into event-based sentiment analysis, which extracts events that have impacts on stocks from massive news articles and identifies the sentiments within those events in order to make more accurate predictions, and the algorithms of how to select key sentences, extract events and identify event sentiments in this circumstance worth more research work.

CRediT authorship contribution statement

Xiaodong Li: Conceptualization, Methodology, Investigation, Project administration, Supervision. **Pangjing Wu:** Formal analysis, Software, Validation, Visualization, Writing - original draft. **Wenpeng Wang:** Writing - review & editing.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under grant 61602149, and in part by the Fundamental Research Funds for the Central Universities under grant 2019B15514.

References

Aiolli, F., & Donini, M. (2015). EasyMKL: A scalable multiple kernel learning algorithm. *Neurocomputing, 169*, 215–224.

Akita, R., Yoshihara, A., Matsubara, T., & Uehara, K. (2016). *Deep learning for stock prediction using numerical and textual information. 2016IEEE/ACIS 15th international conference on computer and information science*. IEEE1–6. https://doi.org/10.1109/ICIS.2016.7550882.

Baccianella, S., Esuli, A., & Sebastiani, F. (2010). *Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. Lrec10. Lrec* 2200–2204.

Baltas, N., Jessop, D., Jones, C., & Lancetti, S. (2014). *Investment strategies and textual analysis signalsTechnical Report*. UBS.

Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of Computational Science, 2*(1), 1–8.

Cambria, E., Poria, S., Hazarika, D., & Kwok, K. (2018). *SenticNet 5: Discovering conceptual primitives for sentiment analysis by means of context embeddings. Thirty-second AAAI conference on artificial intelligence*. AAAI1795–1802.

Chen, K., Zhou, Y., & Dai, F. (2015). *A LSTM-based method for stock returns prediction: A case study of China stock market. 2015 IEEE international conference on big data*. IEEE2823–2824. https://doi.org/10.1109/BigData.2015.7364089.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv:1406.1078.

Deng, S., Mitsubuchi, T., Shioda, K., Shimada, T., & Sakurai, A. (2011). *Combining technical analysis with sentiment analysis for stock price prediction. 2011 IEEE ninth international conference on dependable, autonomic and secure computing*. IEEE800–807.

Ding, X., Zhang, Y., Liu, T., & Duan, J. (2015). *Deep learning for event-driven stock prediction. Proceedings of the 24th international conference on artificial intelligenceIJCAI'15*AAAI Press2327–2333.

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research, 270*(2), 654–669. https://doi.org/10.1016/j.ejor.2017.11.054.

Gers, F., Schmidhuber, J., & Cummins, F. (2000). Learning to Forget: Continual Prediction with LSTM. *Neural Computation, 12*, 2451–2471.

Hochreiter, S., & Schmidhuber, J. (1997). *Long short-term memory. 9*. MIT Press.

Holcroft, J., Winter, P., Jessop, D., Antrobus, O., Wu, S., Baltas, N., ... Gerken, J. (2016). *R advice getting started with random forestsTechnical Report*. UBS.

Hopfield, J. J. (1982). *Neural networks and physical systems with emergent collective computational abilities. 79*, National Acad Sciences2554–2558.

Hu, Z., Liu, W., Bian, J., Liu, X., & Liu, T.-Y. (2018). *Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. Proceedings of the eleventh ACM international conference on web search and data miningWSDM '18*New York, NY, USA: ACM261–269. https://doi.org/10.1145/3159652.3159690.

Hutto, C. J., & Gilbert, E. (2014). *Vader: A parsimonious rule-based model for sentiment analysis of social media text. Eighth international AAAI conference on weblogs and social media*.

Stone, J. P., Dunphy, C. D., Smith, M., & Ogilvie, M. D. (1966). *The general inquirer: A computer approach to content analysis. 4*MIT Presshttps://doi.org/10.2307/1161774.

Jozefowicz, R., Zaremba, W., & Sutskever, I. (2015). *An empirical exploration of recurrent network architectures. International conference on machine learning*2342–2350.

Junqué De Fortuny, E., De Smedt, T., Martens, D., & Daelemans, W. (2014). Evaluating and understanding text-based stock price prediction models. *Information Processing & Management, 50*(2), 426–441. https://doi.org/10.1016/j.ipm.2013.12.002.

Li, W., & Liao, J. (2018). *A comparative study on trend forecasting approach for stock price time series. Proceedings of the international conference on anti-counterfeiting, security and identification, asid2017-Octob. Proceedings of the international conference on anti-counterfeiting, security and identification, asid* 74–78. https://doi.org/10.1109/ICASID.2017.8285747.

Li, X., Cao, J., & Pan, Z. (2019). Market impact analysis via deep learned architectures. *Neural Computing and Applications, 31*, 5989–6000.

Li, X., Huang, X., Deng, X., & Zhu, S. (Huang, Deng, Zhu, 2014a). Enhancing quantitative intra-day stock return prediction by integrating both market news and stock prices information. *Neurocomputing, 142*, 228–238.

Li, X., Wang, C., Dong, J., Wang, F., Deng, X., & Zhu, S. (2011). *Improving stock market prediction by integrating both market news and stock prices. International conference on database and expert systems applications*. Springer279–293.

Li, X., Xie, H., Chen, L., Wang, J., & Deng, X. (Xie, Chen, Wang, Deng, 2014b). News impact on stock price return via sentiment analysis. *Knowledge-Based Systems, 69*, 14–23.

Li, X., Xie, H., Lau, R. Y., Wong, T. L., & Wang, F. L. (2018). Stock prediction via sentimental transfer learning. *IEEE Access, 6*, 73110–73118. https://doi.org/10.1109/ACCESS.2018.2881689.

Li, X., Xie, H., Song, Y., Zhu, S., Li, Q., & Wang, F. L. (2015). Does summarization help stock prediction? A news impact analysis. *IEEE Intelligent Systems, 30*(3), 26–34.

Li, X., Xie, H., Wang, R., Cai, Y., Cao, J., Wang, F., ... Deng, X. (2016). Empirical analysis: stock market prediction via extreme learning machine. *Neural Computing and Applications, 27*, 67–78.

Loughran, T., & McDonald, B. (2011). When is a liability not a liability? Textual analysis, dictionaries, and 10-Ks. *The Journal of Finance, 66*(1), 35–65.

Malandri, L., Xing, F. Z., Orsenigo, C., Vercellis, C., & Cambria, E. (2018). Public mood–driven asset allocation: The importance of financial sentiment in portfolio management. *Cognitive Computation, 10*(6), 1167–1176.

Malkiel, B. G., & Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The Journal of Finance, 25*(2), 383–417. https://doi.org/10.1111/j.1540-6261.1970.tb00518.x.

Nelson, D. M., Pereira, A. C., & De Oliveira, R. A. (2017). *Stock market's price movement prediction with LSTM neural networks. Proceedings of the international joint*

conference on neural networks2017-May. *Proceedings of the international joint conference on neural networks* IEEE1419–1426. https://doi.org/10.1109/IJCNN.2017.7966019.

Nguyen, T. H., Shirai, K., & Velcin, J. (2015). Sentiment analysis on social media for stock movement prediction. *Expert Systems with Applications, 42*(24), 9603–9611.

Ratto, A. P., Merello, S., Oneto, L., Ma, Y., Malandri, L., & Cambria, E. (2018). *Ensemble of technical analysis and machine learning for market trend prediction. 2018 IEEE symposium series on computational intelligence (ssci)*. IEEE2090–2096.

Schumaker, R. P., & Chen, H. (2009). A quantitative stock prediction system based on financial news. *Information Processing & Management, 45*(5), 571–583.

Sehgal, V., & Song, C. (2007). *Sops: stock prediction using web sentiment. Seventh IEEE international conference on data mining workshops (ICDMW 2007)*. IEEE21–26.

Shi, L., Member, S., Teng, Z., Wang, L., Zhang, Y., & Binder, A. (2019). DeepClue : Visual interpretation of text-based deep stock prediction. *IEEE Transactions on Knowledge and Data Engineering, 31*, 1094–1108. https://doi.org/10.1109/TKDE.2018.2854193.

Tetlock, P. C. (2007). Giving content to investor sentiment: The role of media in the stock market. *The Journal of Finance, 62*(3), 1139–1168.

Tetlock, P. C., Saar-Tsechansky, M., & Macskassy, S. (2008). More than words: Quantifying language to measure firms' fundamentals. *The Journal of Finance, 63*(3), 1437–1467.

Tieleman, T., Hinton, G. E., Srivastava, N., & Swersky, K. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*.

Wu, J.-L., Su, C.-C., Yu, L.-C., & Chang, P.-C. (2012). *Stock price predication using combinational features from sentimental analysis of stock news and technical analysis of trading information. International proceedings of economics development and research*.

Xiong, R., Nichols, E. P., & Shen, Y. (2015). Deep learning stock volatility with Google domestic trends. *2*, 0–5 arXiv:1512.04916.

Yu, J. H., Kang, J., & Park, S. (2019). Information availability and return volatility in the bitcoin market: Analyzing differences of user opinion and interest. *Information Processing & Management, 56*(3), 721–732. https://doi.org/10.1016/j.ipm.2018.12.002.

Zaremba, W., Sutskever, I., & Vinyals, O. (2014). Recurrent neural network regularization. arXiv:1409.2329.

Zhang, G., Xu, L., & Xue, Y. (2017). Model and forecast stock market behavior integrating investor sentiment analysis and transaction data. *Cluster Computing, 20*, 789–803. https://doi.org/10.1007/s10586-017-0803-x.

Zhang, X., & Tan, Y. (2018). *Deep stock ranker: A LSTM neural network model for stock selection. International conference on data mining and big data*. Springer614–623.