The Author-Topic Model on Tweets for Bitcoin Price

HO Ngok Chao(A0181923A) CHEN Zhiwei(A0181419A) Dong Xiaohan(A0181382E)

LIAN Zirui(A0181993N) DU Yiyun(A0181347A)

1. Introduction

In our experiment, Author-Topic model was applied to tweets to find topics. Based on the model, Topic trends were generated across time and were used as input features for BP Neural Network, Elman Neural Network and LSTM for Bitcoin price prediction.

2. Literature Review

Author-Topic Model is an extension of Topic model. Topic, which is discovered unsupervised, is defined as a distribution over words. Author's interest is defined as a distribution over topics. The generative mechanism is described below.

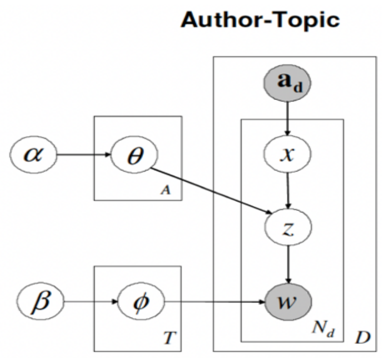


Figure 2.1: Author-Topic model

First, an author x is randomly chosen from the author list a_d . Secondly, x is substituted into Topic by Author distribution θ which used Dirichlet Distribution α as its prior, for sampling for topic z. Then, topic z is substituted into Word by Topic distribution ϕ, which used Dirichlet Distribution β as its prior, for the sampling of word w.

In order to estimate ϕ and θ. The referenced paper [1]used Gibbs sampling. Initialization would be assigning words to random topic and author pair. Next, apply Equation (2.1) to every word and make new assignments accordingly. Then, repeat this step 1000 times and save ϕ and θ at the end according to (2.2) and (2.3).

$$P(z_i = j, x_i = k | w_i = m, \mathbf{z}_{-i}, \mathbf{x}_{-i}, \mathbf{w}_{-i}, \mathbf{a}_d) \propto \frac{C_{mj}^{WT} + \beta}{\sum_{m'} C_{m'j}^{WT} + V\beta} \frac{C_{kj}^{AT} + \alpha}{\sum_{j'} C_{kj'}^{AT} + T\alpha} \quad (2.1)$$

$$\phi_{mj} = \frac{C_{mj}^{WT} + \beta}{\sum_{m'} C_{m'j}^{WT} + V\beta} \quad (2.2) \qquad \theta_{kj} = \frac{C_{kj}^{AT} + \alpha}{\sum_{j'} C_{kj'}^{AT} + T\alpha} \quad (2.3)$$

For left-hand side of Equation (2.1), $z_i = j$ and $x_i = k$ represent the assignments of the ith word in a document to topic j and author k respectively, $w_i = m$ represents the observation that the ith word is the mth word in the lexicon (unique word list), z_{-i} and x_{-i} represent all topic assignments and word assignments not including the ith word

[1] Michal RosenZvi, Thomas Griffiths, Mark Steyvers, Padhraic Smyth (2018). The Author-Topic Model for Authors and Documents

respectively. w_{-i} represents all words' position in lexicon not including the ith word and a_d represents author list. For right-hand side of (2.1), C_{kj}^{AT} is the number of times author k is assigned to topic j, not including the current instance. C_{mj}^{WT} is the number of times word m is assigned to topic j. For the denominator, m' represents all words in the lexicon and j' represents all topics. Equation (2.2) and (2.3) shared the same notation as (2.1). V is the number of unique words and T is the number of topics. ϕ_{mj} is the probability of using the word m given topic j is used while θ_{kj} is the probability of using topic j given the author is k.

The model was originally applied to NIPS collection (academic papers) in the reference paper. In this experiment, the subject is changed to tweets. The idea of Topic trends came from another paper[2] with the intention to find topic popularity in the research literature. The definition of topic trends in this experiment is changed to be based on the absolute frequency and are believed to be able to reflect the market mood explained later.

3. Implementation

To reduce the effect of meaningless words, URL, most punctuations and stop words were removed. Moreover, all letters were converted to lowercase and extra whitespace was trimmed. One version of the text kept all emoji, and another removes all emoji.

In this experiment, Number of documents D = 50811, Number of author A = 9. The number of topics is chosen to be T = 100, and ϕ and θ were saved the program after 1000 iterations. a_d only has 1 author. Number of unique words is denoted by V = 65160. For Equation (2.1), (2.2) and (2.3), hyperparameters β is set to be 0.01 and α is set to be 0.25.

Therefore, size of ϕ is V*T and that of θ is A*T; C++ was chosen to implement this model due to large scale computation and ϕ and θ are saved in hash tables, because if V becomes large ϕ would become a sparse matrix; Equation (2.1) was applied to each word which new Author-Topic assignments were sampled from it.

[2] Steyvers, M., Smyth, P., Rosen-Zvi, M., & Griffiths, T. (2004). Probabilistic author-topic models for information discovery.

After 20 hours (measured by package Boost) on 2.3Ghz CPU, ϕ and θ results were obtained. Sample output is shown below. To test which contains more information for Bitcoin price prediction, the process was repeated twice for text with and without emoji.

Table 4.1: Top3 Word by Topic 13 Distribution (left) & Topic by Author Distribution (right)

Topic ϕ	Word ϕ	Prob ϕ	Author θ	Topic θ	Prob θ
13	rt	0.0265172	anondran	0	0.103402
13	twice	0.00736854	anondran	45	0.0278108
13	support	0.00736854	anondran	90	0.0211368

Topic trends were built as below and used as input for bitcoin prediction based on the assumption that one topic reflects one type of market mood. The assumption behind is, for these 9 authors, when they describe one type of market mood, they always use the same set of words; therefore, the author-topic model explains their occurrences by grouping them into the same topic. While the sentiment of a single word is ignored, topic constituted by the probability of appearance of a set of words describes the mood which can be seen for topic 13. Unrelated topics should be filtered by Granger Causality Test.

4. Topic Trends

Each word has a different probability of occurrence in 100 topics (pre-division). The topic the word belongs to is determined to be the topic the word has the highest probability to be used. After that, frequencies of topic assignments of words are counted through different time points. In Figure 4.1, f_{11} represents the frequency of the words assigned to topic 1 at time point 1.

Since topics are word frequency using the same unit, they are not standardized (such as z-score). 100 topic trends were constructed by obtaining frequency from time point 1 to time point n.

$$\begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1n} \\ f_{21} & f_{22} & \cdots & f_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ f_{1001} & f_{1002} & \cdots & f_{100n} \end{bmatrix}$$

Figure 4.1: Topic Trend Distribution

5. Empirical Test

Because the price of Bitcoin fluctuates greatly, we temporarily took the difference

(DPRICE), as well as the difference in the logarithmic form (DIF_LOG_PRICE) to ensure passing the ADF test. Correspondingly, the difference of frequency of a topic was also taken as a measure for changes. The difference of topic frequency and difference of bitcoin price can be explained as the movements or changes in direction (Up or Down). After linear regression, R^2 was too little to show a linear relationship. Then, Granger Causality Test is performed to find the factors with significant influence according to the study[3]. Granger Causality analysis can be used to investigate the hypothesis that market mood states are predictive of changes[4] in bitcoin price.

Table 5.1: Granger Causality Test Result

		Topics
DIF_LOG_PRICE	<0.05	D2/D6/D8/D9/D13/D22/D33/D29/D36/D37/D54/D55/D58/D63/D66/D67/D68/D85/D93
	<0.1	(Include above)D16/D17/D19/D26/D33/D65/D78/D86
DPRICE	<0.05	D13/D14/D16/D19/D2/D22/D23/D29/D32/D33/D35/D36/D39/D49/D58/D59/D6/D66/D67/D69/D77/D79/D8/D99/D100
	<0.1	(Include above)D1/D17/D41/D62/D78/D93/D94

The common factors are:

D2/D13/D22/D23/D29/D36/D58/D6/D66/D67/D69/D8/D17/D93.

Because the number of endogenous variables in the VAR model is limited, in order to reduce the numbers of factors, common factors (both reject the Null at 95% confidence level) are used for the VAR model. These common factors are all significant at different price formation, therefore they are used for bitcoin price prediction. After plot two AR graphs, both of the two forms are found stationary based on the VAR model.

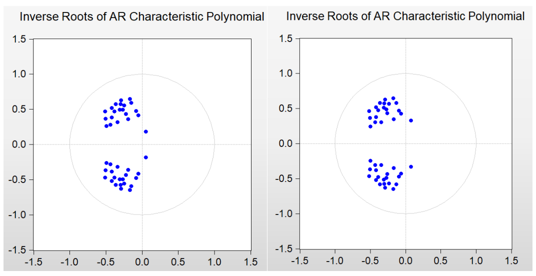


Figure 5.1: AR Graph

According to AIC, the lag order of DPRICE equals to 5, while the lag order of DIF_LOG_PRICE is more than 15, hence DPRICE is preferred to show the significant

[3] Chen, Wenhao & Cai, Yi & Lai, Kin Keung & Xie, Haoran. (2016). A topic-based sentiment analysis model to predict stock market price movement using Weibo mood.

[4] Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market.

lag-period influence. Meanwhile, DPRICE passes the cointegration test, indicating that DPRICE series and selected factors have a cointegrating relationship.

After doing PCA, the result is not good enough. 10 PCs can only cover 85% of the total variance. Then, from Figure 5.2, we can clearly distinguish the fluctuation direction in each period. Finally, the fluctuation converges to zero within ten periods.

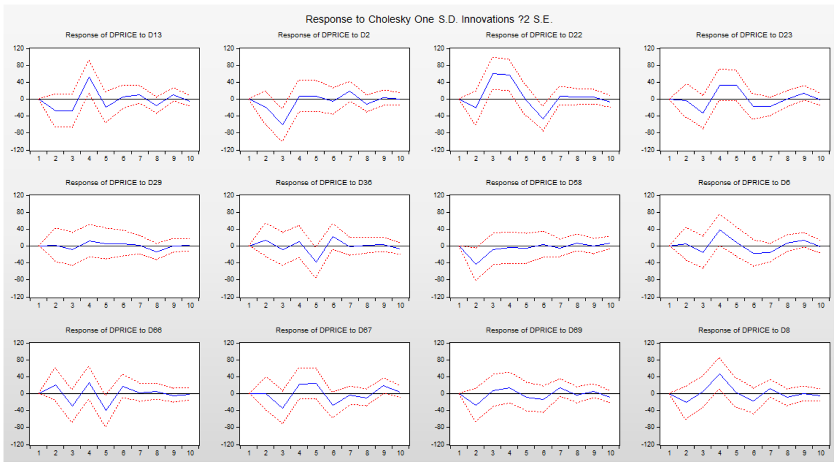
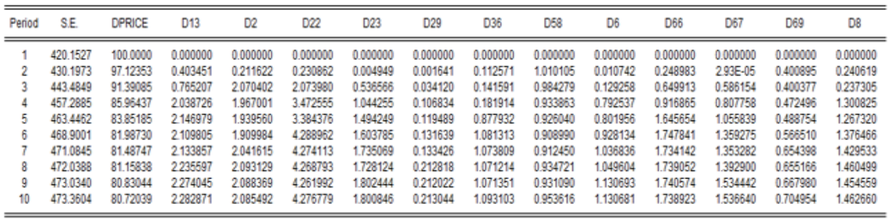


Figure 5.2: Impulse Response Graph

From Table 5.2, most of the variance is caused by the price fluctuation itself. Until 10 periods, D22 causes 4.27%, D13 and D2 cause 2% respectively.

Table 5.2: Variance Decomposition



Period	S.E.	DPRICE	D13	D2	D22	D23	D29	D36	D58	D6	D66	D67	D69	D8
1	420.1527	100.0000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2	430.1973	97.12353	0.403451	0.211622	0.230862	0.004949	0.001641	0.112571	1.010105	0.010742	0.248983	2.93E-05	0.400895	0.240619
3	443.4849	91.39085	0.765207	2.070402	2.073980	0.536566	0.034120	0.141591	0.984279	0.129258	0.649913	0.586154	0.400377	0.237305
4	457.2885	85.96437	2.038726	1.967001	3.472555	1.044255	0.106834	0.181914	0.933863	0.792537	0.916865	0.807758	0.472496	1.300825
5	463.4462	83.85185	2.146979	1.939560	3.384376	1.494249	0.119489	0.877932	0.926040	0.801956	1.645654	1.055839	0.488754	1.267320
6	468.9001	81.98730	2.109805	1.909984	4.288962	1.603785	0.131639	1.081313	0.908990	0.928134	1.747841	1.359275	0.566510	1.376466
7	471.0845	81.48747	2.133857	2.041615	4.274113	1.735069	0.133426	1.073809	0.912450	1.036836	1.734142	1.353282	0.654398	1.429533
8	472.0388	81.15838	2.235597	2.093129	4.268793	1.728124	0.212818	1.071214	0.934721	1.049604	1.739052	1.392900	0.655166	1.460499
9	473.0340	80.83044	2.274045	2.088369	4.261992	1.802444	0.212022	1.071351	0.931090	1.130693	1.740574	1.534442	0.667980	1.454559
10	473.3604	80.72039	2.282871	2.085492	4.276779	1.800846	0.213044	1.093103	0.953616	1.130681	1.738923	1.536640	0.704954	1.462660

Lastly, we use the data with emoji, re-run and re-test. New common factors obtained are: D13/D14/D2/D22/D23/D29/D36/D54/D58/D6/D66/D67/D69/D8/D85/D9

6. Prediction on Bitcoin Price

6.1 Experiment setting for all methods

(1) Training without common factors of the topics is performed. Namely, the only input is the moving average of Bitcoin price in the past 5 days (t-1, t-2….t-5).

(2) Then training with common factors is conducted. The inputs include both the past 5 days' moving average of Bitcoin price and the individual frequency of each common

factors in the previous day (t-1). Besides, the topic information we used is also divided into two categories: with emojis and without emojis.

6.2 BP Neural Network

6.2.1 Introduction of BP Neural Network

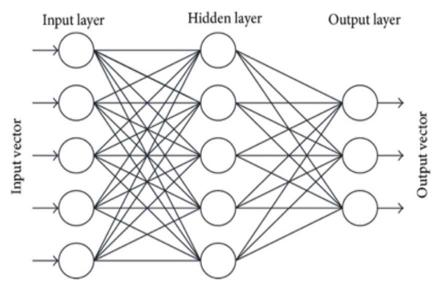


Figure 6.1: BP Neural Network

Any supervised learning algorithm is aimed at finding a function best mapping a set of inputs to correct output. BP trains a multi-layered neural network which can learn appropriate internal representations and learn any arbitrary mapping of input to output. It works by approximating the non-linear relationship between input and output by adjusting the weight values internally. The number of hidden layers is determined by the empirical formula (6.1).

$$\text{the number of hidden layers} = \sqrt{\text{input layers} + \text{output layers}} + a, a \in [1,10] \qquad (6.1)$$

The error function is $J = \frac{1}{2}(\vec{p} - \vec{a})^2$, where vector p is predicted prices, and vector a is our actual prices.

6.2.2 Parameters

• Layers: 5 hidden layers • Goal of train: 0.05 • Epochs: 10000 • Learning rate: 0.000001

6.2.3 Results

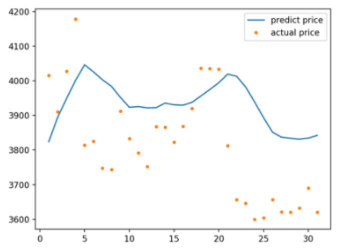


Figure 6.2: Prediction results without topics

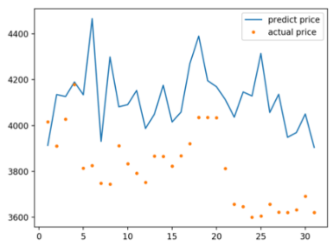


Figure 6.3: Prediction results with topics

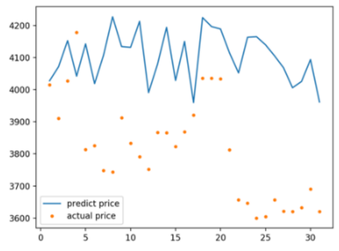


Figure 6.4: Prediction Results without emoji

Function *net.newelm* with *train_rprop* as training option from package *neurolab* in Python is used. For comparison, according to the experiment setting (1), we got the predicted price in the next month as shown in Figure 6.2.

As can seen, the model with only MA didn't predict accurately. Therefore, we proceed to experiment setting (2). With topics included, better predictions are obtained as shown in Figure 6.3. Furthermore, the model is trained with tweets with emoji, which believed to be able to reflect the authors' emotional feelings. The "emoji" prediction result is shown in Figure 6.4.

Both Figure 6.3 and Figure 6.4 show a better performance than the prediction without topics. And the prediction with emoji seems to be more stable. For strict comparisons, MSE, MAPE, precision, recall and F1 score are calculated.

Table 6.1: BP Prediction metrics

	Without topics	With topics	Topics with emojis
MSE	42784.8824	123482.8374	110075.8657
MAPE	0.04633	0.0832	0.08
Precision	0.4286	0.5625	0.4667
Recall	0.4615	0.6923	0.5385
F1 score	0.4444	0.6209	0.5
Accuracy	0.4838	0.5484	0.5161

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (6.2) \qquad Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (6.3)$$

$$F1\ score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (6.4) \qquad Accuracy = \frac{True\ Positive + True\ Negative}{Sample} \quad (6.5)$$

According to accuracy, the second BP neural network model has gotten the most accurate result. However, the model including emoji has smaller MSE than that excluding emoji, which shows less volatility. Hence, the information from tweets is helpful in Bitcoin price prediction, but the emoji doesn't have a significant impact under BP model.

6.3 Elman neural network

6.3.1 Introduction of Elman neural network

Unlike BP neural network, Elman neural network is a kind of feedback neural network[5]. Figure 6.5 shows the structure of Elman which includes input layer, hidden layer, connection layer, and output layer. The connection layer is composed of connection units, which has a delaying effect on previous samples data. The function of connection layer mainly centers on memorability, which reflects a mixed influence coming from the previous state of hidden layer and the current input by the use of linking neurons and adjusting weight between the hidden layer and connection layer. Due to the connection

[5] Binghui Wu, Tingting Duan; A Performance Comparison of Neural Networks in Forecasting Stock Price Trend; International Journal of Computational Intelligence Systems, Vol. 10 (2017) 336–346

layer, the network system can adapt to the time-varying dynamic characteristics and has strong global stability.

Figure 6.6 shows the whole running process of Elman neural network. After a series of circulation process, the errors between final values in the output layer and expected values are computed. The error function is $e_{Elman} = \frac{1}{2}[y(r)^k - y(k)]^T[y(r)^k - y(k)]$, where $y(k)$ is the actual prices and $y(r)^k$ is the predicted prices. If the error becomes acceptable, the running process will be terminated.

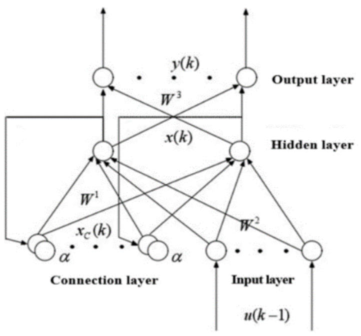
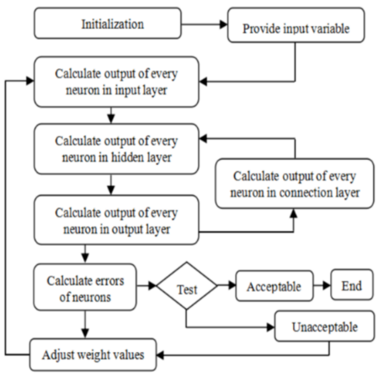


Figure 6.5: Structure of Elman Neural Network.



Initialization
Provide input variable
Calculate output of every neuron in input layer
Calculate output of every neuron in hidden layer
Calculate output of every neuron in connection layer
Calculate output of every neuron in output layer
Calculate errors of neurons
Test
Acceptable
End
Unacceptable
Adjust weight values

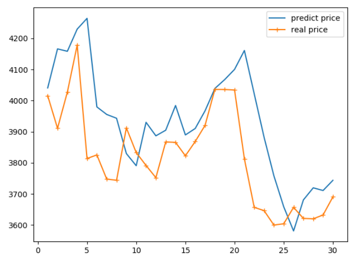
Figure 6.6: Running Process of Elman Neural Network

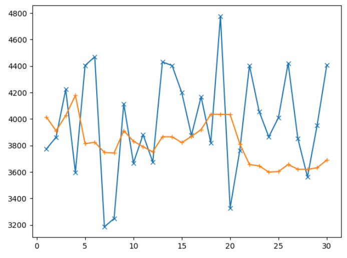
6.3.2 Model Settings

Similarly to BP, *net.newelm* with *train_rprop* from package *neurolab* are used.

- Layers: 2 layers including output layer. The hidden layer has 5 neurons. The output layer has 1 neuron.
- Epochs: 10000
- Goal of train: 0.0001
- Learning rate: 0.000001

6.3.3 Result





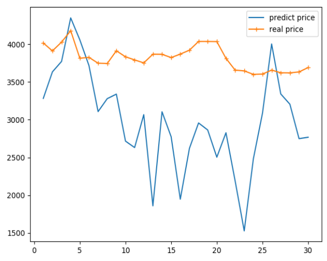


Figure 6.7: Prediction results without topics Figure 6.8: Prediction results with topics Figure 6.9: Prediction results without emoji.

Table 6.2: Elman neural network prediction metrics

	Without Topic Distribution	With Topic Distribution	With Topic Distribution(with emoji)
MSE	27925.15023	199364.1081	1063614.1017
MAPE	0.0319	0.0934	0.3547
Precision	0.4375	0.5625	0.6

Recall	0.5385	0.6923	0.6923
F1 score	0.4828	0.6207	0.6429
Accuracy	0.5	0.5333	0.6667

From figure 6.7 using experiment setting (1), the predicted price shows obvious lagging from the real price. When topic trends were added into input, this situation got improved. Although MSE and MAPE may increase when topic distribution was added, F1 score and accuracy were higher. The model using twitter topic distribution including emoji has highest F1 score and accuracy, showing the best performance in forecasting the change direction of bitcoin price. Elman Neural Network performs best among three models.

6.4 LSTM neural network

6.4.1 Introduction of LSTM neural network

Long short-term memory (LSTM) neural network is an extension for standard Recurrent Neural Networks (RNN) including Elman. LSTM enables RNN to remember their inputs over a long period of time. The problematic issue of vanishing gradients of RNN is solved through LSTM because it keeps the gradients steep enough and therefore the training is relatively short and with high accuracy.

As can be seen in Figure 6.10, the repeating module in LSTM contains four interacting layers. The repeating module (cell) is detailed in Figure 6.11. Every cell contains three gates: Forget gate, Updating gate and Output gate. Forget gate decides how much information from the old subject should be forgotten. Updating gate decides what new information we're going to store in the cell state. Finally, the Output gate decides what we're going to output. $\frac{1}{m}\sum_{i=1}^{m}|(y_i - \hat{y}_i)|$ where y_i is the actual price and $\hat{y}_i$ is the predicted price.

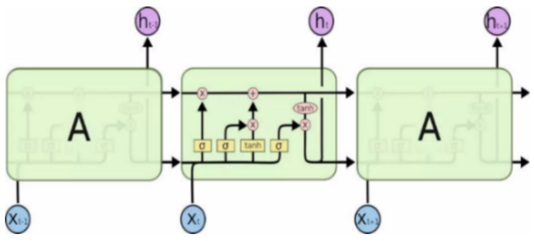


Figure 6.10: LSTM Neural network

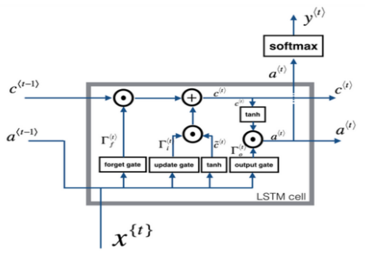


Figure 6.11: Repeating module

6.4.2 Parameters

Package keras is used to train the model.

• Hidden layers: 1 • Units: 50 • Epochs: 100 • Batch size: 72

6.4.3 Results

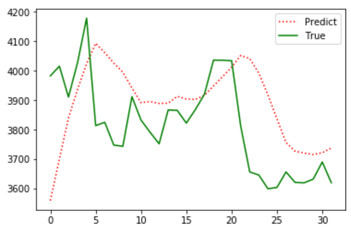


Figure 6.12: Prediction results without topics

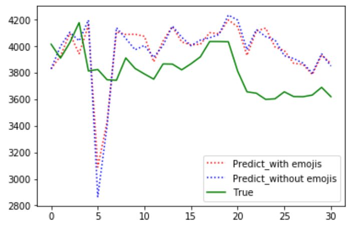


Figure 6.13: Prediction results with topics

From the figures, it can be seen that when the neural network was trained without topics, there's obvious price lag influence. In contrast, with topics included, the price lag influence is eased partially. The prediction metrics of LSTM neural network is below.

Table 6.3: LSTM Prediction metrics

	Without topics	With topics (with emojis)	With topics (without emojis)
MSE	37814.1345	86929.8808	97814.6009
MAPE	4.0833	6.778	6.88131
Precision	0.4667	0.3846	0.3571
Recall	0.5385	0.4167	0.4167
F1 score	0.5	0.4	0.3846
Accuracy	0.5313	0.3923	0.3871

However, according to the table above, LSTM neural network with common factors inputs performed worse than the model without topics as inputs. Topics information with emojis and without emojis does not have major difference under LSTM neural network. The reason why LSTM does not perform well may be that it put weights on previous long-term information, but Bitcoin price more relies on the latest information.

7. Conclusion

In general, topic information generated from author-topic model helps to predict the Bitcoin price especially when emoji is included, although LSTM does not perform well. In the future, experiments focus on using NLP to classify words into each topic. Other improvement factors will be tested in future research such as dynamic author-topic model, NARX neural network.