Sentiment Analysis of Stocktwits Data with Word Vector and Gated Recurrent Unit

Oscar¹, Hilman F. Pardede¹²

¹Graduate School in Computer Science, STMIK Nusa Mandiri, Jakarta, Indonesia
²Research Center for Informatics, Indonesian Institute of Sciences, Indonesia

¹14002331@nusamandiri.ac.id
²hilman@nusamandiri.ac.id

Abstract—Prediction of stock movements is important in the business world for knowing the movement of stock both for buying and selling goods. Stock is a financial product characterized by high risk, high return and flexible trading, which is favored by many investors. Investors can get abundant returns by accurately estimating stock price trend. Historical price is often used to predict the stockprice, it can only estimate the periodical trends of the stockprice. However, there could be a particular event that may affect the price. So it cannot capture sudden unexpected events. Social media texts like tweets can have huge impacts on the stock market. By analysing the sentiments of social media information, unexpected behaviour of the price trend could be detected. In this study, we propose to use Gated Recurrent Unit (GRU) for predicting the sentiment of tweets related to stockprice. We implement word vector, in particular word2vec, as features for GRU. Our experiments show that the proposed method is better than other deep learning based sentiment analysis such as BERT (Bidirectional Encoder Representations from Transformers) and BiLSTM (Bidirectional Long Short Term Memory).

Keywords—Stocktwits, GRU, Word2vec, stockprice, sentiment analysis

I. INTRODUCTION

Stock is a financial product characterized by high risk, high return and flexible trading, which is favored by many investors. Investors can get abundant returns by accurately estimating stock price trends (Ji, Wang, and Yan 2021). Prediction of stock movements is important in the business world for knowing the movement of stock both for buying and selling goods. Stock prices have an intrinsically volatile and non-stationary nature, making their rise and fall hard to forecast (Adam, Marcet, and Nicolini 2016). Uninformed trading decisions can leave traders and investors prone to financial risk and experience monetary losses. On the contrary, careful investment choices can maximize profits (de Souza et al. 2018).

Stock market prediction methods are divided into two main categories: technical and fundamental analysis. Technical analysis focuses on analyzing historical stock prices to predict future stock values (i.e. it focuses on the direction of prices). On the other hand, fundamental analysis relies mostly on analyzing unstructured textual information like financial news and earning reports (Subhi Alzazah and Cheng 2021). Many researchers believe that technical analysis approaches can predict the stock market movement (Chung and Shin 2018), (Long, Lu, and Cui 2019), (Xu and Cohen 2018). In general, these researches did not get high prediction results as they depend heavily on structured data neglecting an important source of information that is the online financial news and social media sentiments.

Usually, historical data is used to predict the stock price. However, historical data alone fail to capture market surprises and impacts of sudden unexpected events. Social media texts like tweets can have huge impacts on the stock market (Sawhney et al. 2020). For example, US President Donald Trump shared tweets expressing negative sentiments against Lockheed Martin, which led to a loss of around $5.8 Billion to the company’s market capitalization (Phil McParlane 2018). Therefore, text mining system to predict sentiments from social media related to event that may affect the stock price is needed.

There have been several studies that use text mining for predicting sentiments for events related to stock price prediction. In (Jaggi et al. 2021). Actually, text sources are also divided into two parts. One is focused on the direct feedback of customer and market reception for a new product to estimate the popularity and price of a product, even a stock (Xie 2017). It is mainly based on identifying positive and negative words and processing text with the purpose of classifying its emotional stance as positive or negative, widely used in predict turning point or tendency of a specific stock (Nguyen 2018), (Kim and Jeong 2019), (Shen Li et al. 2017). The other one tries to deal with passive influence from external text sources. Such sources are usually from mass-media, like main websites, newspapers and magazines (Malik, Sheoran, and Dhand, n.d.), (Suganya and Vijayarani 2020), (Prameswari et al. 2017).

In this paper, for FAANG dataset we propose an architecture of GRU for this tasks with Word2Vec and
SkipGram with parameters are the same as previous studies. By comparing the prediction accuracy with the previous studies, we aim to assess whether the inclusion of sentiment variables based on news articles and twitter sentiment can enhance the accuracy of the stock price prediction process.

II. METHOD

For implementing the GRU model using Word2Vec embeddings, the Google news pretrained Word2Vec model was used. This model has been trained on more than 100 billion words extracted from Google news data and includes word vectors for approximately 3 million phrases and words (Mikolov et al. 2013).

A. Data Preprocessing

Data preprocessing is a data mining technique which is used to transform the raw data in a useful and efficient format. Steps Involved in Data Preprocessing for this study are:

1) Data Cleaning: Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled.

2) Tokenizer: Tokenizer is the process of tokenizing or splitting a string, text into a list of tokens. One can think of token as parts like a word is a token in a sentence, and a sentence is a token in a paragraph. In this study tokenizer use is TweetTokenizer.

3) Padding: Since every sentence in the text has not the same number of words, we can also define maximum number 160 words for each sentence, if a sentence is longer then we can drop some words.

4) Converting to categorical: a numpy array (or) a vector which has integers that represent different categories, can be converted into a numpy array (or) a matrix which has binary values and has columns equal to the number of categories in the data.

B. Word vector

For this research we use Word2Vec. Word2Vec has two models, namely Continuous Bag-of-Word (CBOW) and Skip-Gram. skip-gram model’s aim is to predict the context from the target word, the model typically inverts the contexts and targets, and tries to predict each context word from its target word. For this research we use skip-gram.

C. Modeling

For this research we use Gated Recurrent Network (GRU) for modelling.

Figure 1. system overview

Figure 2. example skip-gram with 2 window

Skip-gram architecture works with three layers, the first layer is the input layer, the second layer is hidden layer, and the third layer is the output layer. In this study we only use 1 layer, for vector size we use 100, for maximum distance between the current and predicted word within a sentence we use 5 windows, to ignores all words with total frequency lower we use 5, for workers we use all cpu count from google colab.

Figure 3. example GRU architecture

Gated Recurrent Network (GRU) is also a variant of RNN and but simpler than LSTM, it also performs good in NLP area. GRU has 2 gates which are an reset gate and update gate compared with 3 of LSTM (Saihan Li and Gong 2021).
For making GRU model we use this step:

1) Embedding Layer: The embedding layer here is for becomes the input layer, where the amount of data will be inputted is the total words that exist in dataset. And for the output dimension are 160.

2) GRU layer: In this study use only 1 layer with 100 units of neurons.

3) Output Layer: In output layer use 4 neuron units and the activation function use softmax. Because the research is a categorical classification, the loss function categorical_crossentropy is used and the optimization function uses 'RMSprop'. The batch size used is 32 with 2 epochs and the model evaluation parameter is 'accuracy'.

III. EXPERIMENTAL SETUP

dataset use in this study FAANG StockTwits pre-trained dataset with 3 percentage change labeling, this dataset was taken from previous study. The dataset accessible from their drive on google drive with link: https://drive.google.com/file/d/1RtUEwOBWltS1JaV6onuahpNPFLJX3lsb/view?usp=sharing. The data obtained in this study amounted to 2,566,826 data with 8 attributes that will be processed to sentiment analysis of Stocktwits data. Some of the attributes contained in the dataset are:

<table>
<thead>
<tr>
<th>No</th>
<th>Attribute name</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>symbol</td>
<td>object</td>
</tr>
<tr>
<td>2</td>
<td>message</td>
<td>object</td>
</tr>
<tr>
<td>3</td>
<td>datetime</td>
<td>object</td>
</tr>
<tr>
<td>4</td>
<td>user</td>
<td>int64</td>
</tr>
<tr>
<td>5</td>
<td>message_id</td>
<td>int64</td>
</tr>
<tr>
<td>6</td>
<td>Date</td>
<td>object</td>
</tr>
<tr>
<td>7</td>
<td>Time</td>
<td>object</td>
</tr>
<tr>
<td>8</td>
<td>label</td>
<td>int64</td>
</tr>
</tbody>
</table>

IV. RESULTS AND DISCUSSION

We performed various experimentations on the StockTwits FAANG dataset. With more than 6.4 million data samples, the baseline models, traditional machine learning models, and proposed models were trained for all set of possibilities to extract results. All the experimentations were done for three labelling techniques.

All the experimentations were done on FAANG data with 1 and 2 years of the subset. Table I shows all the results for Percentage change with three labels (Positive, Neutral, and Negative) with 1 and 2 years of data.

In traditional machine learning, several tests have been carried out with several models are Naive Bayes with TF-IDF Vectorizer ngram range (2,2) and smoothing parameter 0.1. Gradient Boosting with CountVectorizer, GradientBoostingRegressor, GridSearchCV, learning rate 0.025, max depth 5, estimators 50 and cross validation 5. Logistic Regression with CountVectorizer, max iter 5000, solver liblinear, GridSearchCV, vector range [1,2],[2,2], C [0.01,0.1,1], penalty l1 and l2, cross validation 5. Random Forest with CountVectorizer, TfidfTransformer, use idf True, max features sqrt, max depth 5, estimators 200, class weight balanced.

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc 1 Year</th>
<th>Acc 2 Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive Bayes</td>
<td>49%</td>
<td>49%</td>
</tr>
<tr>
<td>Gradient Boosting</td>
<td>20%</td>
<td>19%</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>48%</td>
<td>47%</td>
</tr>
<tr>
<td>Random Forest</td>
<td>33%</td>
<td>34%</td>
</tr>
</tbody>
</table>

Based on table 2, it can be seen for machine learning traditional result that Naive bayes get better accuracy for 1 year data and 2 year data with a value of accuracy 49%, while naive gets the lowest accuracy with a value of 20% for 1 year data and 19% for 2 year data.

In deep learning several tests have been carried out with several models are BERT with Changing Text to the format required for BERT model training, add special tokens, max length 160, truncation True, pad to max length True, return attention mask True, return tensors pt. Add the encoded sentence to the list. And its attention mask (simply differentiates padding from non-padding). Converting the lists into tensors, Creating TensorDataset by combining the training inputs 90:10. Dividing the training and validation dataset by selecting samples randomly. DataLoader dataset, sampler RandomSampler(dataset), batch size 32, Loading BertForSequenceClassification model for training, BertForSequenceClassification from pretrained, bert base uncased, number labels 3, output attentions False, output hidden states False, Assigning the model on the GPU, optimizer AdamW, lr 1e-5, eps 1e-8, weight decay 1e-5, get linear schedule with warmup, epochs 2.

FinBERT with max seq length 64, train batch size 16, eval batch size 32, learning rate 5e-5, number train epochs 10.0, warm up proportion 0.1, no cuda False, do lower case True, seed 42, local rank -1, gradient accumulation steps 1, fp16 False, output mode classification, discriminate True, gradual unfreeze True, encoder no 12, optimizer BertAdam.

FinBERT with Load the pre-trained FinBERT Model, encoded dict tokenizer encode plus, add special tokens True, max length 160, pad to max length True, return attention mask True, return_tensors pt, truncation True, train dataloader, sampler RandomSampler(train_dataset), batch size 32, optimizer AdamW, lr 1e-5, eps 1e-8, weight decay 1e-2, epochs 2, Evaluating Performance on Test dataset, add special tokens True, max length 160, padding to max length True, return attention mask True, return_tensors pt.

CNN with Word2vec use EMBEDDING DIM 300, MAX VOCAB SIZE 175303, MAX SEQUENCE LENGTH 160, batch size 64, number epochs 2.
RegexpTokenizer, add a 1D convnet with global maxpooling, Dropout 0.5, activation sigmoid, optimizer adam.

CNN with BERT embeddings use bert-base-uncased, number labels 3, output attentions False, output hidden states False, MAX SEQUENCE LENGTH 160, MAX VOCAB SIZE 30,522, EMBEDDING DIM 768, VALIDATION_SPLIT 0.1. Add one dimensional convolutional network with global maxpooling, filters 128, kernel size 3, activation relu, Dropout(0.5), output layer activation sigmoid, optimizer adam.

Word2Vec with BiLSTM and attention use word2vec skip-gram, size 100, window 5, min count 5, workers 4, sg 1, MAX SEQUENCE LENGTH 160, MAX NB WORDS 200000, EMBEDDING DIM 100, VALIDATION_SPLIT 0.1, activation softmax, optimizer rmsprop, epochs 2, batch size 32, Attention layer GRU, activation softmax, optimizer rmsprop, epochs 2, batch size 32.

Fasttext with BiLSTM and attention use FastText, size 100, window 5, min count 5, workers 4, sg 1, MAX SEQUENCE LENGTH 160, MAX NB WORDS 200000, EMBEDDING DIM 100, VALIDATION_SPLIT 0.1, activation softmax, optimizer rmsprop, epochs 2, batch size 32, Attention layer GRU, activation softmax, optimizer rmsprop, epochs 2, batch size 32.

Word2Vec With GRU use word2vec, skip-gram, size 100, window 5, min count 5, workers cpu_count(), sg 1, MAX SEQUENCE LENGTH 160, MAX NB WORDS 200000, EMBEDDING DIM 100, VALIDATION_SPLIT 0.1, activation softmax, optimizer rmsprop, epochs 2, batch size 32.

Based on table 3, it can be seen for deep learning models result that Word2Vec With GRU get better accuracy with a value of accuracy 80% for 1 year data and 76% for 2 year data , while CNN with BERT embeddings gets the lowest accuracy with a value of accuracy 23% for 1 year data and Word2Vec with BiLSTM and attention with a value of accuracy 20% for 2 year data.

V. CONCLUSION

In conclusion, the labelling technique used to classify the messages is based on the changes in stock prices and not on sentiments. Currently, not many experiments are performed on this labelling technique. Based on experiment result from this study, it can be conclude that Gated Recurrent Unit with word2vec feature better as compared to the other models on the table for prediction, the two years of data for FAANG company.

Compared to traditional models, the transformer-based model’s training time was too high, and there was no significant improvement in model performance.

REFERENCES


