Intraday Market Movement Prediction Using Sentiment Analysis and Historical Data

ELEC-E5550 - Statistical Natural Language Processing

Bruce Nguyen
School of Science
Aalto University
Espoo, Finland
bruce.nguyen@aalto.fi

Laurine Burgard-Lotz
School of Electrical Engineering
Aalto University
Espoo, Finland
laurine.burgard-lotz@aalto.fi

I. Introduction

Market prediction is a very challenging problem. In fact, the Efficient Market Hypothesis (EMH) stated that assets price reflect all the available information, thus no one can forecast the market with more than 50% accuracy [5]. Nevertheless, there has been numerous attempts to solve the problem using from historical stock data (technical analysis) to economic indicators (fundamental analysis). Recently, social media platforms have brought about an explosion of real-time textual data containing rich information about the market. Among such types of information, public sentiment has been shown to have predictive power [2].

In this project, we aim to predict the S&P 500 Index using such information as well as the historical data. The sentiments are collected by performing sentiment analysis using the state-of-the-art language model RoBERTa on user posts, as known as Tweets, on the social media website Twitter. Once all the needed features are acquired, the market movement will be predicted using a Gated Recurrent Units (GRUs) recurrent neural network.

II. Methods

A. Data

1) Dataset

The initial plan was to collect Tweets thanks to the Twitter API or the Stocktwits one. However, to do that, we have to generate keys that are returned by creating a Developer account, which turned out to be too complicated. We finally decided to find a dataset of raws collected tweets in english, and to preprocess it by ourselves. Another problem we faced was the lack of recent publicly available data sets. Furthermore, we thought that the data in Stocktwit's tweets might be too precise for BERT (a language specific to traders, whereas BERT is pre-trained on an everyday language).

We found several interesting datasets:

- *Sentiment140 dataset* : it contains 1,600,000 tweets extracted using the twitter API, which are already annotated with 0 for negative and 4 for positive. The advantage of this dataset is that we can use it to train our algorithms and/or measure its accuracy, since it is already labeled. However, the data is really sparse in term of publication date, and thus would probably not have been effective for predicting stock market movements.
- *Cheng-Caverlee-Lee dataset* [3] : it is a collection of public tweets, which is way more complete than the previous dataset, as we can see in Figure 1. However, the tweets are not labelled. That is why we decided to use this dataset as our **testing dataset for the prediction of stock movement**.
- *Data used for the International Workshop on Semantic Evaluation (SemEval)* [1] : this dataset contains testing and training data for *Message Polarity Classification*, in the form of comma-separated values files with ID, Label (either positive, neutral or negative), Tweets and their date of publication, for the years 2013, 2014, 2015 and 2016. The date are most of the time missing, so this dataset can't be used for the stock movement prediction. However, after preprocessing, this **dataset would be ideal to train our models to label tweets**.

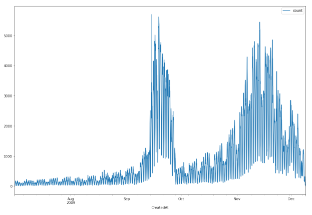

Fig. 1. Publication by date in the Cheng-Caverlee-Lee dataset

2) Preprocessing

The preprocessing of the Tweets has to be different depending on the model we plan to use. We will see later in the report that BERT use its own tokenizer and has its own particular type of inputs.

[1]https://www.dropbox.com/s/byzr8yoda6bua1b/2017_English_final.zip?

In any case, tweets are a special type of data, since they contain a very casual language, with the addition of emojis, "@", "#", or URLs, which constitute noise for our sentiment analysis task. So we have to process these elements, to create a dataset which can be easily learned by a classifier. We decided to remove them, by using regular expressions (thanks to the `re` module from Python). As an example, to remove '@' from our tweets, we can apply `tweet = re.sub(r'(@.*?)[\s]', '', ' '.join(tweet))`

Another important thing when we deal with tweets is to convert 2 or more letter repetitions to 2 letters. Indeed, some people add multiple characters in a word to emphasize their thoughts, but this would affect the model.

Of course we also did general preprocessing on our tweets such as lowercasing the tweets, and removing stopwords, by using the Natural Language Toolkit (nltk) which contains a list of stopwords in english.

We could have proceeded without removing the stopwords, as this may affect the performance of the sentiment classification task in the wrong way, according to this study from Saif et al. [10], which evaluate how stopwords removal affect the performance of Twitter sentiment classifiers. However, we decided to remove them anyway for the baseline method, because it is based on word counting from lexicon and thus consider stopwords as noise. We don't need to remove stopwords for the BERT model, because it was pre-trained on entire sentences.

We did not apply stemming to the tweets either, because the lexicons used by the baseline alogirithm are based on complete words and not stem of words and BERT doesn't need stemming.

B. Sentiment Analysis

1) Baseline

As a baseline, we decided to use a simple positive and negative word counting method. For this purpose, we used the opinion lexicon [2] of Hu and Liu (2004), containing lexicon of positive and negative words in English. In order to classify a given tweet and assign it a positive, negative or neutral sentiment, we counted if the number of positive words in the tweet was respectively superior, inferior or equal to the number of negative words.

The advantage of this method is that we don't need to train it and it is really easy to implement when we already have the lexicons. However, it is really slow and the accuracy is not good at all. When labelling the testing set from the SemEval dataset, we get an accuracy of 54.53%. The Figure 2 plot the confusion matrix for this baseline algorithm :

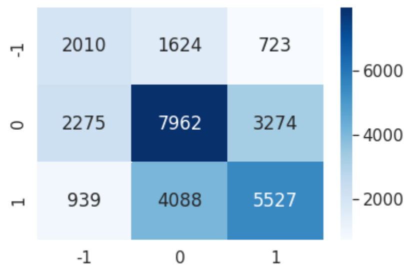

Fig. 2. Confusion matrix over the testing set (-1 for negative, 0 for neutral and 1 for positive sentiment) for the word counting

Since this model proved to be a bit too simple, we also implemented a decision tree thanks to the scikit-learn library. For this purpose, we first preprocessed the training and testing set from the SemEval dataset, then we initialized `X_train, y_train, X_test, y_test` as follows: `X_train` and `X_test` are `numpy arrays` containing the Tweets (`string`) from the training and testing set respectively while `y_train` and `y_test` are `numpy arrays` containing the corresponding labels (`int` -1,0 or 1). After that, to extract features from our textual data, we used TF-IDF.

```
from sklearn.feature_extraction.text import
    TfidfVectorizer
# Calculate TF-IDF
tf_idf = TfidfVectorizer()
X_train_tfidf = tf_idf.fit_transform(X_train)
X_test_tfidf = tf_idf.transform(X_test)
```

We are now able to apply the decision tree model using these features to predict the sentiment. The resulting accuracy is 57.54%, after tuning the hyperparameters to avoid overfitting and increase the accuracy.

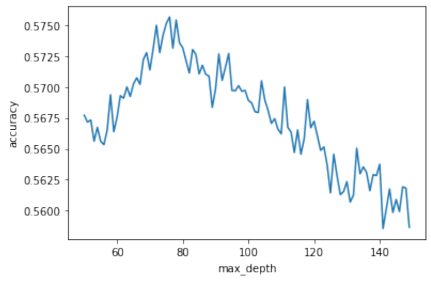

Fig. 3. Accuracy based on the maximum depth of the decision tree

The confusion matrix is shown in Figure 4

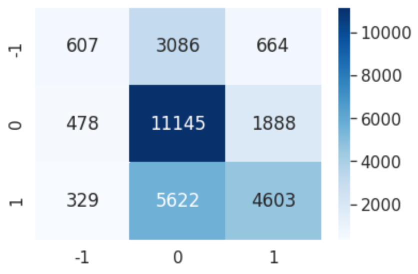

Fig. 4. Confusion matrix over the testing set (-1 for negative, 0 for neutral and 1 for positive sentiment) for the Decision Tree

[2]https://www.cs.uic.edu/ liub/FBS/opinion-lexicon-English.rar

We can see that the accuracy is actually a poor choice to see the efficiency of the model, since the class distribution is unbalanced. It thus gives high scores to models predicting the most frequent class, ie. *neutral (0)* here. To know how well the decision tree is performing we could rather compute the precision, recall and f1-score. We can see the result in Table 1.

TABLE I
CLASSIFICATION REPORT FROM SCIKIT-LEARN METRICS FOR THE DECISION TREE

	Precision	**Recall**	**F1 score**
-1	0,38	0,1	0,15
0	0,55	0,84	0,66
1	0,65	0,42	0,51
Accuracy			0,57
Macro average	0,53	0,45	0,44
Weighted average	0,56	0,57	0,53

2) BERT Model

For our main model, we wanted to fine-tune a pre-trained BERT model[3] (Bidirectional Encoder Representations from Transformers) from the **Huggingface** PyTorch library, using the concept of *transfer learning*. As we said previously, the theory consists in adding an untrained layer of neurons at the end of the pre-trained BERT model, and training the new model for our sentiment analysis task.

To implement that, we first had to tokenize the data, using the `BertTokenizer`, provided by the library. Actually, BERT can take 512 tokens at a time, need the start, end, [SEP] and [CLS] special tokens, and an attention mask. For this purpose we used the `tokenizer.encode_plus()` method, which allowed us to split our tweets into tokens, pad or truncate the tweets to the max length, add the special tokens,... We can get the attention mask using the `.get('attention_mask')` method, and the encoded integers sequences using `.get('input_ids')`. The next step was to create an iterator by using the torch `DataLoader` class. This allowed us to create batches for training from the tweets samples.

After that, we can instantiate a BERT model and implement the fine-tuning: we decided to add a hidden linear layer, followed by ReLU and a final linear layer which has 3 outputs for the 3 possible labels. To make the prediction, we'll pass the result through a softmax function which generate a class probability vector.

There are different ways to train the model. One of the methods consists in training the entire model, and thus back-propagate the error through the entire architecture to update the different weights. However, this method is extremely time-consuming. Another way to proceed is to train only the last layer, while freezing the pre-trained one (so we don't update their weights). This can be done using `.requires_grad = False` for the parameters of the pre-trained BERT model.

[3]It's a bidirectional transformer pretrained using a combination of masked language modeling objective and next sentence prediction on a large corpus

The hyper-parameters recommended are a batch size of size 16 or 32, few epochs (3/4), the `AdamW` optimizer with a learning rate of $3e-5$. Due to a lack of time, we were unfortunately unable to wait for our model to train. Even using the GPU from Google Colab, 1 epoch was taking more than 6 hours to be finished, and we needed the result to continue our work and proceed to the stock prediction thanks to the sentiment analysis. That is why we finally decided to use an already fine-tuned model, based on **RoBERTa** [4]. It was trained on approximately 58M tweets and finetuned for sentiment analysis with the *TweetEval benchmark* [1].
In Figure 5, the confusion matrix for BERT model. Compared to our two previous models, the accuracy is definitely higher : we obtain 75% over the test set.

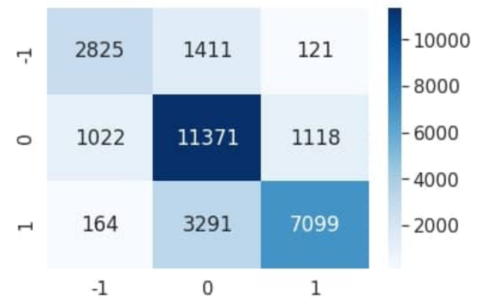

Fig. 5. Confusion matrix over the testing set (-1 for negative, 0 for neutral and 1 for positive sentiment) for BERT

C. Market Prediction

1) Setup

We made predictions on the movement of the opening value of the S&P 500 index from September 1st to December 31nd in 2009 at an hourly frequency. The label of the movement at a given time t is 1 if the index value goes up or stays the same, 0 if it decreases. The features used are both the sentiment information and the historical movement. The train, validation, test set size ratios are 70%, 15% and 15%, respectively.

Using the fine-tuned RoBERTa model, we obtained the labels of the tweets from the *Cheng-Caver-Lee* dataset. However, to use the labels for the prediction, we need to create an index that represent the public sentiment at time t. To achieve this, we start by assigning sentiment numerical values -1, 0, 1 to the *negative*, *neutral*, *positive* tweets, respectively. It can be assumed that the sentiment value of a tweet coming from an individual at time t corresponds to the true sentiment value of said individual. In other words, the tweet sentiment reflects the sentiment of the author. Consider n independent random variables $X^1_{[t-\delta,t)}, X^2_{[t-\delta,t)}, ..., X^n_{[t-\delta,t)}$, each corresponds to the sentiment value of an individual $i = 1, 2, ..., n$ who tweet at a time interval $[t-\delta, t)$. We can estimate the public sentiment at a time interval $[t-\delta, t)$ by the sentiment index $SI_{[t-\delta,t)}$ as

$$SI_{[t-\delta,t)} = \bar{x}_{[t-\delta,t)} = \frac{1}{n}\sum_{i}^{n} X^i_{[t-\delta,t)}.$$

[4]https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment

To get statistically significant estimates, we need to determine an appropriate minimum sample size using the formula

$$n = \frac{1}{\frac{d^2}{z_{\alpha/2}^2 \cdot \sigma^2} + \frac{1}{N}},$$

where d is the desired margin of error, $z_{\alpha/2}^2$ is the critical value of the Normal distribution at $\alpha/2$, σ^2 is the population variance, N is the population size. After having investigated the distribution of the tweet sentiments, we have determined the desirable values for parameters d and σ^2 as 0.02 and 0.50, respectively. Since we are looking at the sentiment of the public in the US, N will naturally be its population size. Finally, we get the minimum sample size of tweets needed to estimate the public sentiment at a 90% confidence level as $n = 1689$.

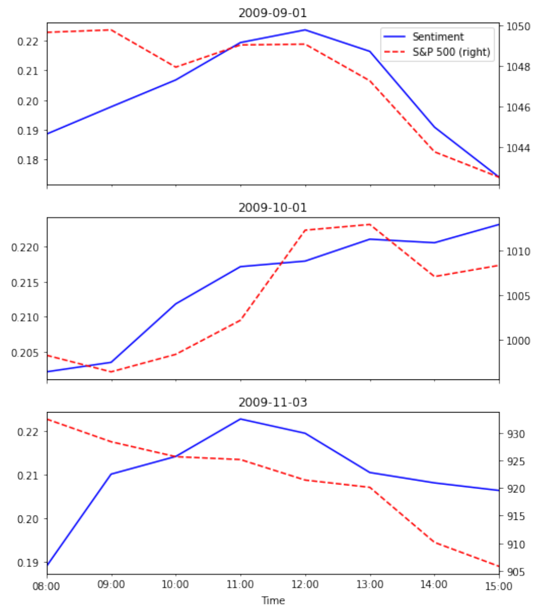

Fig. 6. Visualizing the sentiment and the S&P 500 index on different trading days. The y-axis on the right and the red dashed line represent the sentiment value, while that the left y-axis and blue line show the S&P 500 index value.

Unfortunately, the counts of tweets in the *Cheng-Caver-Lee* dataset at 1-hour intervals in general do not meet this requirement, thus one-to-one mapping from the tweets and their sentiments to the S&P 500 value was not possible. Thus, we have chosen $\delta = 8$ to resolve the problem. In other words, to predict the S&P 500 movement at time t, we use the aggregated sentiment index for an interval of 8 hours before t. This approach allows for the accounting of the new sentiment information at time t, and the long interval can also make the index more robust against outliers. In the end, we arrived at an array with 2 columns representing sentiment index $SI_{[t-8,t)}$ and the S&P 500 index value movement at time t. To make better predictions, we use a window of previous index value movement and sentiment index from time t to $t - l$ to make a prediction for the index value movement at time t. In our experiment, we choose the parameter $l = 10$.

2) *Baseline*

In the experiment, we adopt a Support Vector Machine (SVM) as the baseline model. SVMs are demonstrated as one of the most effective classical machine learning models empirically while also is simple to implement [6]. In our experiment, the SVM implementation comes from the `scikit-learn` library, in particular the `SVC` classifier. Mathematically, the goal is to find $w \in \mathbb{R}^p$ and $b \in \mathbb{R}$ such that the prediction given by $\text{sign}(w^T\phi(x) + b)$ is correct for the maximum number of examples, given training vectors $x_i \in \mathbb{R}^p$, in two classes, and a vector $y \in \{1, -1\}^n$. To achieve that, `SVC` tries to solve the following optimization problem:

$$\begin{aligned} \min \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i \\ \text{s.t. } y_i(w^T\phi(x_i) + b) \leq 1 - \zeta_i \\ \zeta_i \leq 0, i = 1, ..., n \end{aligned}$$

where ζ_i is the slack variable and C is the penalty term [9]. The most important parameters available for tuning is the kernel type, which we defaulted to `rbf` after a process of iterating over different sets of parameters.

3) *GRU*

GRUs, or Gated Recurrent Units, is a popular type of Recurrent Neural Network for machine learning problems with a temporal dimension. We chose it for this problem not only because we are working with time series data, but also due to the fact that we have a dataset too small for the use of more complex ones such as LSTMs, which might result in overfitting. Nevertheless, GRUs have been demonstrated to exhibit performance similar to LSTMs.

Conceptually, the models uses an update gate and a reset gate [4] to decide what information to keep and discard from the current input, respectively. The 4 computations made at one time step t given input x_t and the hidden state h_{t-1} by GRU are:

$$\begin{aligned} z_t &= \sigma_g(W_z x_t + U_z h_{t-1} + b_z), \\ r_t &= \sigma_g(W_r x_t + U_r h_{t-1} + b_r), \\ \hat{h}_t &= \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h), \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t, \end{aligned}$$

where $\hat{h}_t$ is the candidate activation vector, W, U and b are the parameter matrices and vector, σ_g is the sigmoid function, and ϕ_h is the tanh function. To obtain the label of index movement as predictions, we first flatten the output from GRU and then feed it to a linear layer and finally a sigmoid function.

In the modelling process, we adopted the binary cross-entropy loss function and the Stochastic Gradients Descent algorithm for optimizing the parameters of the model after numerous experiments. The hyperparameters are chosen as follows: learning rate is 0.005, hidden size of GRU is 5, number of GRU layers is 1.

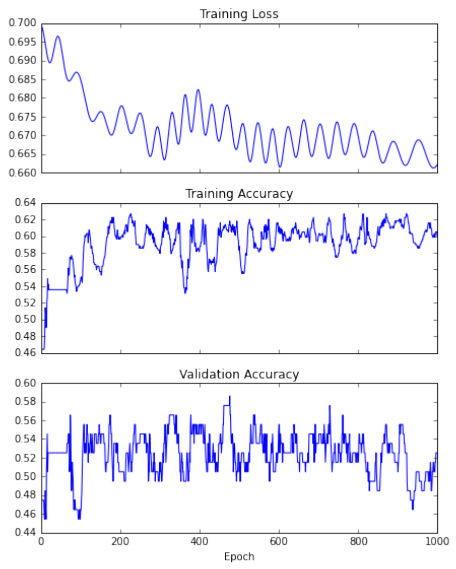

Fig. 7. The GRU training process

The training loss and accuracy in the training process suggests our model is learning. However, we can see a very stochastic behavior of the validation accuracy, which can be a result of the validation set being too small. In addition, the chance of the model overfitting the training set is very high, which can lead to high variance.

D. Results

	SVM	LSTM
Validation	51.12	58.58
Test	49.18	53.53

TABLE II
VALIDATION AND TEST ACCURACY

In general, stock prediction is a very challenging task, and an accuracy of roughly 56% being satisfactory for binary stock movement prediction [8]. With regards to our results, the SVM did not perform well, with the accuracy being merely random guess. The GRU model, on the other hand, demonstrated an improvement of 3% over the result of the SVM, which is significant for this task. In fact, its true accuracy can even be better, since the size of the test set can pose a problem for accurately measuring the true performance of the models.

Other state-of-the-art sophisticated models include TSLDA [8], HAN [7], Stocknet with accuracy levels of 54.07, 57.64, 58.23, respectively [11]. Even though such numbers is not directly comparable to our study because of the different datasets, the methodology is very much similar, and we can see the general landscape of the field. Holistically, even after seeing the impressive results from the mentioned models, we can still see that GRU performed very well, given that it is much simpler and thus less computationally intensive - a big advantage for this application.

III. Conclusion

The project used sentiment analysis and historical prices to predict intraday stock prices movement. Our contribution is we are the first to use these methods on intraday market data, as far as we know. Specifically, we predicted the hourly values of the S&P 500 index using public mood estimated from Twitter and historical values of the index. To do so, we have collected and processed tweets, then feed them to a BERT model fine-tuned for sentiment analysis. BERT is a state-of-the-art machine learning model used for NLP tasks, which clearly surpasses the baselines we have implemented (basic word count and decision tree).

Once we had the sentiments associated with the tweets, we were able to use them in an GRU neural network, as a complement of the historical stock movement, in order to predict the future stock movement. The final accuracy achieved by GRU was 53.35%, which was surprisingly significant considering those of the more sophisticated models. To improve the results, the size of the data should be increased much further, with the prediction interval extended to preferably a year.

IV. Miscellaneous

A. Division of labor

- Bruce Nguyen: I. Introduction, II.A. Event Embeddings in Methods (in Literature Review), III.B.2) BERT (coding), III.C. Market Prediction

- Laurine Burgard-Lotz: II.B. Sentiment Analysis, III.A. Data, III.B Sentiment Analysis, IV. Conclusion

B. Appendix

The following drive contains the different notebooks we used for the project :

https://drive.google.com/file/d/1Ezy4KUCPEHGvi70Q3q75sNuwr9USRaVv/view?usp=sharing

References

[1] Francesco Barbieri et al. "TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification". In: *arXiv preprint arXiv:2010.12421* (2020).

[2] Johan Bollen, Huina Mao, and Xiaojun Zeng. "Twitter mood predicts the stock market". In: *Journal of Computational Science* 2.1 (2011), pp. 1–8. ISSN: 1877-7503. DOI: https://doi.org/10.1016/j.jocs.2010.12.007. URL: https://www.sciencedirect.com/science/article/pii/S187775031100007X.

[3] Zhiyuan Cheng, James Caverlee, and Kyumin Lee. "You are where you tweet: a content-based approach to geo-locating twitter users". In: *Proceedings of the 19th ACM international conference on Information and knowledge management*. 2010, pp. 759–768.

[4] Kyunghyun Cho et al. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179. URL: https://www.aclweb.org/anthology/D14-1179.

[5] Eugene F. Fama. "Efficient Capital Markets: A Review of Theory and Empirical Work". In: *The Journal of Finance* 25.2 (May 1970), pp. 383–417. DOI: 10.2307/2325486.

[6] Manuel Fernández-Delgado et al. "Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?" In: *Journal of Machine Learning Research* 15.90 (2014), pp. 3133–3181. URL: http://jmlr.org/papers/v15/delgado14a.html.

[7] Ziniu Hu et al. "Listening to Chaotic Whispers: A Deep Learning Framework for News-Oriented Stock Trend Prediction". In: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. WSDM '18. Marina Del Rey, CA, USA: Association for Computing Machinery, 2018, pp. 261–269. ISBN: 9781450355810. DOI: 10.1145/3159652.3159690. URL: https://doi.org/10.1145/3159652.3159690.

[8] Thien Hai Nguyen and Kiyoaki Shirai. "Topic Modeling based Sentiment Analysis on Social Media for Stock Market Prediction". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, July 2015, pp. 1354–1364. DOI: 10.3115/v1/P15-1131. URL: https://www.aclweb.org/anthology/P15-1131.

[9] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[10] Hassan Saif et al. "On stopwords, filtering and data sparsity for sentiment analysis of twitter". In: (2014).

[11] Yumo Xu and Shay B. Cohen. "Stock Movement Prediction from Tweets and Historical Prices". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 1970–1979. DOI: 10.18653/v1/P18-1183. URL: https://www.aclweb.org/anthology/P18-1183.