BELT: A Pipeline for Stock Price Prediction Using News

Yingzhe Dong
School of Business Administration
Northeastern University
Shenyang, Liaoning, China
dongyingzhe99@sina.com

Da Yan
Department of Computer Science
The University of Alabama at Birmingham
Birmingham, AL, USA
yanda@uab.edu

Abdullateef Ibrahim Almudaifer
Department of Computer Science
The University of Alabama at Birmingham
Birmingham, AL, USA
lateef11@uab.edu

Sibo Yan
Freddie Mac
Washington, D.C., USA
sibo_yan@freddiemac.com

Zhe Jiang
Department of Computer Science
The University of Alabama
Tuscaloosa, AL, USA
zjiang@cs.ua.edu

Yang Zhou
Department of Computer Science and Software Engineering
Auburn University
Auburn, AL, USA
yangzhou@auburn.edu

2020 IEEE International Conference on Big Data (Big Data) | 978-1-7281-6251-5/20/$31.00 ©2020 IEEE | DOI: 10.1109/BigData50022.2020.9378345

***Abstract*—Stock investment is a vehicle for many people to grow their wealth. However, market downturns can cause huge losses and need to be predicted for a timely sell. In fact, with effective prediction, stocks are a good investment even during periods of market volatility as many stocks are "on sale".**

News is an important source of signal for stock price movement. However, stock analysts usually adjust their analysis according to the news in a subject manner, and wrong judgments can cause investors huge losses.

Twitter is a great source for breaking news, and provides a timely stream of signals on stock trends. News on Twitter also tends to have a great impact on the market due to the large number of Twitter users. This paper proposes a data-driven pipeline to timely incorporate Twitter news about a company into a time series prediction model on the company's stock price. Our approach, called BERT-LSTM (BELT), extracts informative features on stock price direction from Twitter news using the state-of-the-art natural language processing (NLP) model BERT, which are then used as covariates to a many-to-many stacked LSTM model that also utilizes historical stock prices to predict the direction of future stock price. Utilizing a carefully curated stock news dataset, we fine-tune BERT to effectively identify those news tweets that are relevant, and to extract NLP features that are indicative of price rises and falls. All model parameters are trained end-to-end to provide a data-driven and objective pipeline to incorporate news signals so as to avoid subjective analysis. Extensive experiments on real stock prices and Twitter news show that BELT is able to predict stock prices more accurately utilizing news information than if historical price data are used alone for prediction, and beats StockNet which is the current state of the art for news-based stock movement prediction.

***Index Terms*—stock, news, prediction, BERT, LSTM**

I. Introduction

Stocks are an equity investment that represents part ownership in a corporation. Investing in stocks can be tricky and wrong judgments can cause huge losses. For example, the stock market crash of 2008 occurred in September 2008, with Lehman Brothers declared bankruptcy on September 15, and money market funds lost $196 billion in the following days; on September 29 alone, the Dow Jones Industrial Average fell 777.68 points in intraday trading. Until the stock market crash of 2020, the 2008 crash was the largest point drop in history, and it took years for the financial crisis to recover.

Unfortunate, we are now at a financial crisis crossroad again with the COVID-19 pandemic that has caused over 5.6 million cases and 175,000 deaths in the US, and the Black Lives Matter protests across the US and around the world trigged by George Floyd's death caused by police violence. In fact, the coronavirus fear has caused five circuit breakers in March, with the Dow Jones Industrial Average declined by 7.8%, 10.0% and 12.9%, respectively, on March 9, 12 and 16. Despite the quick bound-back in recent months activated by the CARES Act (a $2.2 trillion economic stimulus bill passed in March 18, 2020), the market volatility brings a lot of uncertainty down the road given that the COVID-19 pandemic is becoming severer frustrating the reopening efforts. Therefore, it is more important than ever to have a prediction model that can indicate the future stock price directions with a reasonably high confidence to avoid losses and to make profits.

A lot of stock plunges can be inferred from (or are triggered by) news, such as the layoffs by Uber, Lyft and Airbnb, and the reported security flaws of Zoom. These news are timely posted on the various credible Twitter news accounts, and following them in time is important to project price directions. However, stock analysts usually adjust their judgment based on the recent news in a subjective manner, and different analysts can often provide conflicting recommendations. For example, COVID-19 pandemic leads to a skyrocketing of Zoom's user base and most analysts will predict Zoom's stock price to be increasing in the long run as it has been since the beginning of 2020; but the security issues exposed by the skyrocketing of Zoom users caused stock price plunge once, and different analysts may have different interpretation of the duration and significance of the security scandal's impact.

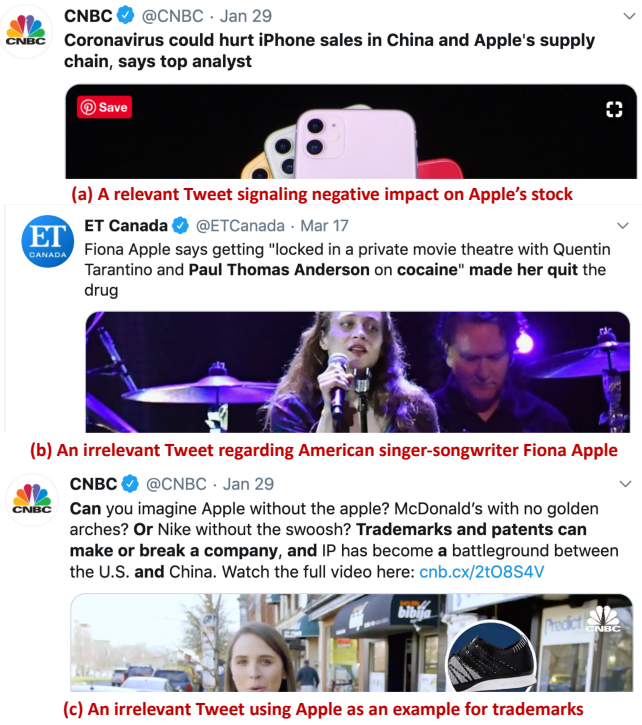

Fig. 1. Examples of News Tweet Relevance about "Apple"

We propose to adopt an objective data-driven approach to integrate news information into the stock price prediction. This is made possible by the recent advancement in deep learning, which has revolutionized how computer vision (CV) and natural language processing (NLP) tasks are carried out. Deep learning models extract informative semantic features from raw data (e.g., words) for effective prediction. Large pre-trained models such as BERT [9] and GPT-3 [6] has demonstrated performance on par with (and sometimes surpassing) human-level accuracy on NLP tasks, and they allow for transfer learning that only requires a few thousand of problem-specific annotated data to achieve state-of-the-art performance for a particular problem domain.

In this paper, we build a deep learning pipeline to continuously extract informative market signals from Twitter news, which are integrated into an online stock price prediction model to guide investment decisions. We consider 2 market signals: (i) historical price of a stock, (ii) a stream of news from Twitter about the stock. Specifically, our workflow contains a pipeline of four modules:

- **Twitter News Crawler.** This module continuously crawls those news tweets that contain the stock names from high-quality Twitter accounts, including credible News accounts and those of top investing experts actively offering market or economic commentary. Besides high tweet quality, those accounts have a large number of followers and are themselves impacting the stock market.
- **Relevance-Based Tweet Extractor.** Despite our quality control by only crawling news from credible Twitter accounts, we can still obtain tweets irrelevant to the stock

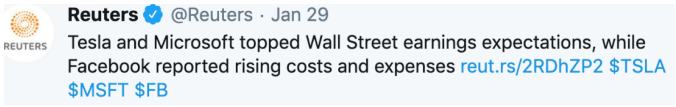

Fig. 2. An Example Tweet Where Subject Matters for the Sentiment

names. To illustrate, consider the 3 examples shown in Figure 1. Figure 1(a) is about a potential drop in Sales of Apple and is relevant to Apple's stock. Figure 1(b) is about a singer whose name is "Fiona Apple", and the tweet is irrelevant due to name ambiguity of "Apple". The "Apple" word in Figure 1(c) is referring to Apple Inc. but the tweet is still irrelevant since it is talking about trademarks and not about the market performance/expectation on Apple. Therefore, our crawled tweets will first go through a relevance judging module, and those that are judged as irrelevant are filtered in time.

- **Sentiment-Based Tweet Feature Extractor.** Sentiment analysis is the interpretation and classification of emotions (positive, negative and neutral) within text data. In our context, we care about if a tweet holds a positive or negative view about a stock, which will impact its price positively or negatively, respectively. We extract sentiment-based features from only those tweets that pass our relevance filter, by building a sentiment classification model on the tweet text. Since sentiments are intuitive for human to annotate, a deep-learning based classifier can be easily trained by transfer learning, and the last-layer semantics features can be used as sentiment features to be used by our next module for price prediction.
- **Price Prediction Network.** Once our sentiment feature extractor is trained, it can be used to extract stock-price related semantics features from each news tweet. These semantics features, along with historical prices, can be fed into a sequence-to-price prediction model to predict the future prices of a stock.

A recent breakthrough in NLP is the invention of *Transformer* [24], which relies entirely on self-attention to compute text representations without using a recurrent network structure, enabling better parallelization and a higher learning quality. Pioneered by Google's BERT [9], a series of Transformer-based language models have been proposed [7], [8], [16], [17], [21], [26], which allows users to fine-tune a pre-trained NLP model for their target NLP applications. We build our relevance-based and sentiment-based tweet classifiers on top of BERT, gluing together semantic features of an entire tweet along with those of the stock subject for effective learning. The need of integrating stock-subject specific features is illustrated by the news tweet shown in Figure 2, where we can see that if the subject is Tesla or Microsoft, the sentiment is positive; while if the subject is Facebook, the sentiment is negative.

The relevance and sentiment of a news tweet is intuitive to human, making annotation a viable task. Using sentiment annotations as supervision, we can train our BERT-based tweet sentiment classifier end-to-end to build a feature extractor that returns the last-layer sentiment features given a tweet text as

the input. These features are the input to our final LSTM [14]-based regression model that predicts future stock prices from historical price data along with the extracted sentiment news features. Since this LSTM model is also trained end-to-end, our solution is fully data-driven and avoids the subjectivity of stock analysts when they digest news for price prediction.

We call our workflow pipeline as BELT since it is a combination of BERT and LSTM models. The main contributions of this work are summarized as follows:

- **Data Curation.** We contributed a set of 5,237 carefully annotated tweet news based on their relevance and sentiments, which can be used to fine-tune any pre-trained lanugage model for classification.
- **Quality Control.** Pre-trained transformer-based language models usually do not work well with average-quality tweets which contain colloquial language and slangs that deviate from formal text. Therefore, we only collect tweets from high-quality accounts. We also use a BERT-based relevance classifier to filter irrelevant tweets.
- **Objective Integration of News Signals.** While analysts usually take news into consideration in a subject manner, we advocate a data-driven approach to learn how news impacts stock prices through end-to-end training. Via human annotated tweet sentiments, we train a BERT-based sentiment classifier whose last-layer features serve as semantic signals to our prediction model, which is also trained end-to-end using real stock prices to learn how these signals weigh in determining the future price trend.
- **Feature integration by Autoregressive LSTM.** We adopt an autoregressive LSTM network to predict future prices of a stock from its prices in previous time steps, expanded with covariates of news sentiment features. Due to the autoregressive nature of our model, the error adjustment in each time step backpropagates to tune the LSTM parameters so that the whole price sequence is fully utilized in supervised price regression learning.

We remark that BELT is general and can be used to predict any price-based investment besides stocks, such as futures and options, where news are also important signals for buys and sells. BELT also works with other curated news sources other than quality-controlled tweets; for example, one may use news from Factiva (https://professional.dowjones.com/factiva/) or other paid service to further improve the data quality.

The rest of this paper is organized as follows. Section II briefly reviews the related work on stock price prediction. Section III presents the preliminaries including a brief review of BERT, and our problem definition and dataset description. Section IV then describes our BELT workflow including the design of its four modules. Section V reports our experimental results on real stock data that verify the effectiveness of BELT. Finally, we conclude this paper in Section VI.

II. Related Work

Autoregressive LSTM networks have been used for inventory prediction by Amazon in their DeepAR model [11] where instead of directly predicting a predicted value y_t, LSTM output predicts the parameters of a negative binomial distribution from which y_t is sampled as a positive (inventory) count. DeepAR also proposes to let LSTM output predict the parameters of a Gaussian distribution from which to sample a continuous value for y_t (e.g., a stock price), and the Gaussian distribution can account for the market microstructure noises. However, the additional distribution layer before y_t slows down training and prediction, and when generating future price sequences for prediction, DeepAR has to sample a price at each time step leading to nondeterministic outputs so that many sampling passes are needed to estimate a stable prediction. Recently, we proposed another autoregressive LSTM model [13] for stock price prediction that takes technical indicators as covariates to capture longer-term stock features, which demonstrated a superior performance compared with other competing predictive models. Unlike DeepAR, our model does not use an additional distribution layer before y_t. To cope with market microstructure noises, we cleanse and sparsify the high-frequency trade data using well-established finance practices before training our LSTM network, which we also follow in the current work. DeepAR has also been improved by replacing the distribution that generates y_t with spline quantile functions and trained with Continuous Ranked Probability Score (CRPS) as the loss [12]; and by fusing state space models (e.g., ARIMA) to enforce temporal smoothness [22].

Besides using covariates extracted from historical prices as we did in [13], a number of works have explored the extraction of other features when other data sources are available, such as the technical indicators of other stocks in the same sector [5], temporal stock correlations [19], correlation between social media and financial data [23], tweet mood scores [18], limit order books [28], etc. However, all these works model the prediction as a binary classification problem rather than a direct price regression, which has been found to be suboptimal in our work of [13]. Among other works, [15] extracts image features from candlestick stock charts using convnets to enhance the temporal features learned by LSTM, while [4] decomposes the stock price time series by wavelet transforms to eliminate noise, followed by extracting deep high-level features using stacked autoencoders, and the features are then fed into LSTM for price forecast. We remark that our model is compatible with all above sources, as they can be fed into our model as covariates (just like our extracted sentiment features).

In our domain of tweet-based stock prediction, StockNet [25] is the state of the art which adopts an encoder-decoder architecture to digest corpus embeddings and historical prices, uses attention weights to account for tweet quality, uses neural variational inference to address the intractable posterior inference, and uses attentive temporal auxiliary to integrate temporal loss. GRU is used in replace of LSTM in the component design. We have compared our model with StockNet and found that our model is superior since we directly adopt a relevance-based tweet filter to eliminate low-quality tweets, and our pipeline design is able to utilize the more powerful Transformer model to extract tweet features.

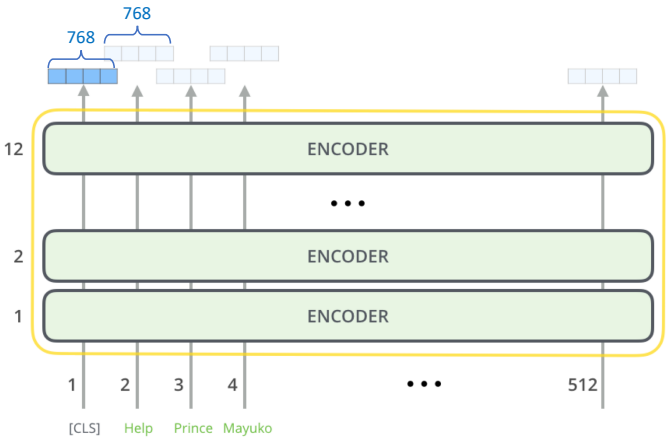

Fig. 3. BERT Architecture

III. Preliminaries

Since BERT is a basic building block in our BELT pipeline, this section first provides a minimally necessary introduction to BERT, before we formally define our problem and describe the types of data that BELT takes as its input.

A. A Brief Review of BERT

BERT [9] has two pre-trained models, BERT-base and BERT-large. BERT-base was explicitly created to compare with OpenAI GPT [20] and thus has exactly the same parameters as GPT: 12 layers of stacked transformer-based encoders and an embedding-vector length of 768 (see Figure 3); while BERT-large is even bigger with 24 layers of encoders and an embedding length of 1024. BERT-large beats BERT-base in all NLP tasks and achieves the state-of-the-art performance, but without loss of generality, this paper assumes that BERT-base is used throughout due to its better efficiency.

Each word in a sentence starts with its embedding representation from the embedding layer (which is pre-trained). As Figure 3 shows, every layer does some multi-headed attention computation on the word representation of the previous layer to create a new intermediate representation. All these intermediate representations are of the same size (i.e., 768).

The final embedding that corresponds to [CLS] (i.e., the first blue vector in Figure 3) is called a "pooled output" which encodes the semantic of the entire input sentence, since it is connected to the loss of a classification task called Next Sentence Prediction (NSP) during pre-training. NSP receives pairs of sentences as input and learns to predict if the second sentence in the pair is right next to the first sentence in the original document. While BERT admits two sentences as input, in an actual classification task such as in our relevance and sentiment classification the input only uses the first sentence.

In Figure 3, the output also contains one embedding vector (in light blue color) for each input word, which represents the semantic features of that word. This word embedding captures its context in the sentence since the transforms utilize the self-attention mechanism. In fact, each word embedding vector is connected to the loss of a task called Masked Language Modeling (MLM) which masks the current word with a token [MASK] and attempts to predict the masked word based on the context provided by the other, non-masked, words in the sentence. The sequence of all these output word embeddings are collectively called as the "sequence output" of BERT.

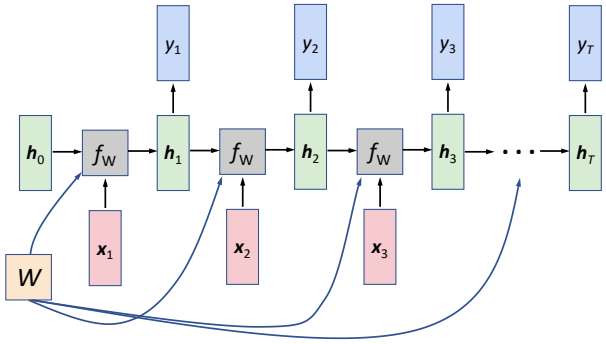

Fig. 4. A Many-to-Many Recurrent Neural Network

When pre-training the BERT model, MLM and NSP are trained together, with the goal of minimizing the combined loss function of the two strategies.

We remark that BERT is a perfect tool to extract the semantic features of a news tweet (i.e., pooled output) as well as the semantic features of the stock subject mentioned in the tweet (i.e., sequence output elements), both of which are important in determining the final classification label as we have illustrated using Figure 2.

B. Our Problem Formulation

This paper studies the problem of predicting the future prices of a stock based on its historical prices and news tweets about this stock. We adopt a many-to-many recurrent neural network (RNN) architecture for stock price prediction.

Many-to-Many RNN. Figure 4 shows an RNN which takes an input sequence $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$ and predicts an output sequence $(y_1, y_2, \ldots, y_T)$. Here, each $\mathbf{x}_t$ is a vector of input features at time step t,, and each y_t is the predicted output at time t. For regression, y_t is a scalar (e.g., stock price in the next time step); let r_t be the actual next stock price, then we use a loss function $\ell_t = (y_t - r_t)^2$ to measure the error at time t. The overall loss of the training/validation data sequence $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$ and $(y_1, y_2, \ldots, y_T)$ is computed as the average loss $\mathcal{L} = \frac{1}{T}\sum_t \ell_t$, and the loss of a set of training mini-batch or a validation set is computed as the average value of $\mathcal{L}$ in the set.

In Figure 4, a recurrence formula $\mathbf{h}_t = f_W(\mathbf{h}_{t-1}, \mathbf{x}_t)$ is applied at every time t, where $\mathbf{h}_t$ is the hidden state at time t which captures all the information up to time t as it sees $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t)$. Bidirectional RNN is common in NLP but we do not consider it here since price prediction is one directional. In a vanilla RNN, we have $f_W = \tanh(W_{hh}\mathbf{h}_{t-1} + W_{xh}\mathbf{x}_t)$, and $\mathbf{y}_t$ is read out from the hidden state $\mathbf{h}_t$ as $\mathbf{y}_t = W_{hy}\mathbf{h}_t$. Note that $W = (W_{hh}, W_{xh})$, since $\mathbf{h}_t = f_W = \tanh(W\mathbf{v}_t)$ where $\mathbf{v}_t = (\mathbf{h}_t, \mathbf{x}_t)^T$. In reality, LSTM [14] is used in replace of vanilla RNN to mitigate the vanish gradient problem.

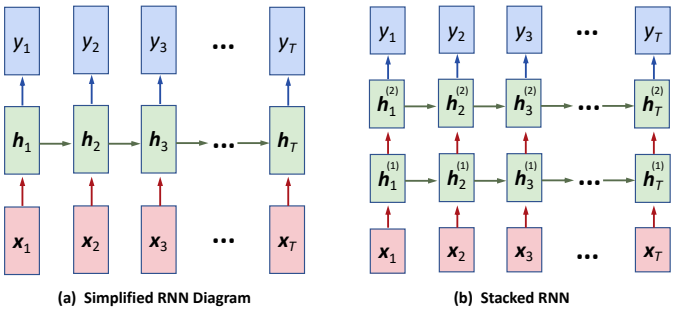

Fig. 5. Simplified RNN Diagram

We call the RNN in Figure 4 as a many-to-many RNN, since it emits an output $\mathbf{y}_t$ at every time t. Since all the recurrence steps share the same recurrence parameter W and the same read-out parameter W_{hy}, the loss ℓ_t at every time t will adjust the values of W and W_{hy} during back-propagation. Such a strong supervision may not be possible in some applications like sentiment classification of a sentence, where only one output (positive/negative) is available after reading the entire sequence of words in a sentence. In such a case, a many-to-one RNN is used where the output is only y_T and the loss function is defined only over y_T. However, since we have the entire price sequence, we should adopt a many-to-many architecture to fully utilize the available supervision for training. In fact, many existing work on stock price prediction still uses a many-to-one model rather than an autoregressive one, as we reviewed in Section II, which is a weakness though easy to fix.

For ease of presentation, we will simplify the RNN in Figure 4 into the RNN diagram in Figure 5(a), where $\mathbf{h}_0$ is omitted since it is usually initialized as a zero state.

If we treat the output $(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T)$ as intermediate features denoted as $(\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T)$, we can stack another layer of RNN on top that takes $(\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T)$ as the input sequence, which gives the stacked RNN in Figure 5(b) with 2 hidden layers that can be depicted by:

$$\begin{aligned} \mathbf{h}_t^{(1)} &= f_{W^{(1)}}(\mathbf{h}_{t-1}^{(1)}, \mathbf{x}_t), & \mathbf{o}_t &= W_{h^{(1)}o}\mathbf{h}_t^{(1)}, \\ \mathbf{h}_t^{(2)} &= f_{W^{(2)}}(\mathbf{h}_{t-1}^{(2)}, \mathbf{o}_t), & \mathbf{y}_t &= W_{h^{(2)}y}\mathbf{h}_t^{(2)}. \end{aligned}$$

More LSTM layers can be stacked on top to extract higher-level features before we use them to generate the output.

Time-Step Choice & Median-Share Price. High-frequency price data are now readily available, such as the Trade and Quote (TAQ) database available at the Wharton Research Data Services (WRDS) website that provides stock data including price, number of shares, and time of each transaction to the nearest second. On each weekday, the market opens at 9:30 AM ET and ends at 4:00 PM ET, and the market data in every second is recorded.

However, market microstructure noises are a major hurdle towards the price analysis when the sampling frequency is high since the realized volatility will not be stable. As a well-established conclusion, [27] shows that a sampling frequency of 5 minutes or longer is a must for stable analysis.

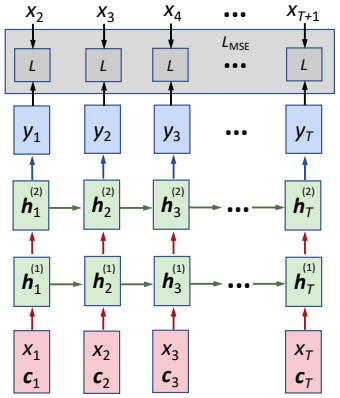

Fig. 6. Our autoregressive RNN Model

Instead of subsampling, we adopt the approach by [10] which better captures average price information in each time block. Specifically, we use the *median share-price* as a representative price for each time block, which is the median price per share treating each share traded as a separate observation.

An Autoregressive LSTM Architecture. Without loss of generality, this paper studies the problem of predicting the median share-price of a future day from the information of a sequence of T previous days. In other words, we use one day as the basic unit of a time block. This is because we can have sufficient news tweets on each day to extract sentiment features, as we only collect tweets from the most credible accounts for quality control. Our model can be easily used for a smaller time block size if the news sources are richer (e.g., with access to Factiva), or we allow some time steps to have no news which can be done by setting news feature values to be fed as LSTM input to 0.

In fact, most investors will not consider buying and selling their stocks on the same day or too frequently due to transaction fees, let alone some markets like that of China that do not allow T+0. Our model provides predicted stock prices for future days so that investors can get an intuitive feeling of how likely and how much the stock price will rise or drop to guide their investment decisions. Note that predicting median share-price means that the price is at least the predicted value for approximately 50% of the time of the day of prediction.

Figure 6 shows our autoregressive many-to-many LSTM network for price prediction. Specifically, y_t is the predicted median share-price on day $(t+1)$, while x_{t+1} is the actual median share-price on that day. Note that having a representative price for each time step is the key to our price regression model, while prior works only model the price information in each time step as a feature vector (e.g., OHLCV) and only predict price directions as a classification problem (rather than regression), providing less supervision than allowed by the actual price since there is no penalty as long as the price direction is correct and the loss makes no effort to further close the gap between y_t and x_{t+1}. At each step t, the input contains the median share-price x_t along with a vector of covariates $\mathbf{c}_t$ that contains the sentiment features of news tweets on the considered stock posted in day t so far.

TABLE I
ILLUSTRATION OF NEWS SOURCES ON TWITTER

Top-10 News Accounts	Top-10 Investor Accounts
CNN Breaking News — @cnnbrk (55.4M followers)	Downtown Josh Brown@ReformedBroker
The New York Times — @nytimes (43.6M followers)	Joe Weisenthal@TheStalwart
CNN — @cnn (42.1M followers)	Vitalik Buterin@VitalikButerin
BBC Breaking News — @bbcbreaking (40M followers)	Charlie Lee@satoshilite
BBC World — @bbcworld (25.3M followers)	Barry Ritholtz@ritholtz
The Economist — @theeconomist (23.8M followers)	Wu-Tang Financial@Wu_Tang_Finance
Reuters Top News — @reuters (20.5M followers)	StockCats@StockCats
The Wall Street Journal — @wsj (16.7M followers)	Rudy Havenstein@RudyHavenstein
Time — @time (16.1M followers)	Liz Ann Sonders@LizAnnSonders
ABC News — @abc (14.4M followers)	Ivan the K@IvanTheK

IV. THE BELT PIPELINE

BELT is consisted of a sequential pipeline of four modules: (1) a Twitter news crawler that continuously collects news tweets, (2) a relevance-based tweet filter that removes irrelevant tweets, (3) a tweet feature extractor that extracts sentiment features from the remaining tweets that are relevant, and (4) our autoregressive LSTM network for price prediction that integrates the tweet sentiment features as covariates.

We remark that modules 2 and 3 are both BERT-based classifiers following the same network architecture but with different labels: module 2 emits a label "relevant" or "irrelevant" for a tweet, while module 3 emits "positive", "neutral" or "negative". Only the last-layer sentiment features of module 3 are useful and are fed as covariates into module 4.

Also, both modules 1 and 2 conduct quality control over tweets to ensure that only high-quality tweets pass them and make it to module 3 for feature extraction. This is important for two reasons: (i) most tweets are informal text with colloquial language and slangs, which is quite different from what BERT's pre-trained tokenizer can effectively recognize; (ii) since anyone can comment on stocks in Twitter, an average tweet containing stock keywords is likely not credible and only brings noise to the prediction in module 4.

We next introduce our four modules one by one in the following four subsections.

A. Twitter News Crawler

This module continuously collects news tweets from credible Twitter accounts. There is a tradeoff when selecting the number of sources on Twitter: the more sources we select, the noisier the tweet quality is which compromises prediction quality, but meanwhile the more abundant news tweets we will obtain which allows for for frequent collection and thus a finer time-block unit. Our strategy in this paper is to only collect tweets from the most credible sources, which not only guarantees source quality but also tends to collect tweets with more formal language that better fits BERT's pre-trained tokenizer that we have to use in order to use BERT-base.

Specifically, we collected those tweets containing stock name keywords from the 30 must-follow Twitter accounts for news in 2019 listed in [3]. Since the number of news tweets from them could be very limited for stocks that are not very popular, we also enrich the sources with the 50 most important people for investors to follow listed in [2]. Table I shows the top-10 from both sources as an illustration.

In our BELT pipeline, an online stream of news tweets can be easily collected from the timelines of the above Twitter accounts using Python's Tweepy library that wraps Twitter Official API. However, for training purpose using historical prices of stocks, we also need to collect their old news tweets. This is made possible by Python's GetOldTweets3 library that "hacks" Twitter Search to bypass Twitter API's time constraints (i.e., you cannot get tweets older than a week).

An alternative solution is to use Twitter's streaming API to get those stocks that contain stock names as keywords. We actually adopted this solution initially but almost all our randomly sampled tweets for scrutiny are irrelevant and of a low quality. Features extracted from them cannot help modules 2 and 3 learn anything, and thus we abandoned this solution. We remark that quality control on tweets is a key to allow effective identification of correct market signals.

In order to train our two text classifiers in modules 2 and 3, we curated a training dataset of 5,237 tweets collected during the duration of January 1st, 2020 to April 30th, 2020 for the 100 most popular stocks listed in [1]. To ensure quality, 5 Computer Science PhD students from the University of Alabama at Birmingham labeled these tweets ensuring that each tweet is labeled by 2 different students.

Our final curated news tweet dataset has 4,380 relevant tweets out of the totally 5,237 thanks to our quality control over the news sources. Among the 4,380 relevant tweets, only 2,830 have consistent sentiment labels by the two students who labeled them: 955 are positive, 982 are neutral, and 893 are negative. All 5,237 tweets are used to fine-tune our relevance classifier in module 2, while only the 2,830 tweets with consistent sentiment labels are used to fine-tune our sentiment classifier in module 3 to ensure the quality of training data. Our curated dataset is released on https://github.com/sdsz20142087/stock_sentiment_data.

B. Relevance-Based Tweet Extractor

Each crawled tweet needs to pass through an additional step of quality control before being passed for feature extraction, and this is achieved by our relevance classifier. Only those that are classified as relevant will be used. Both our relevance classifier and the sentiment classifier are built by fine-tuning

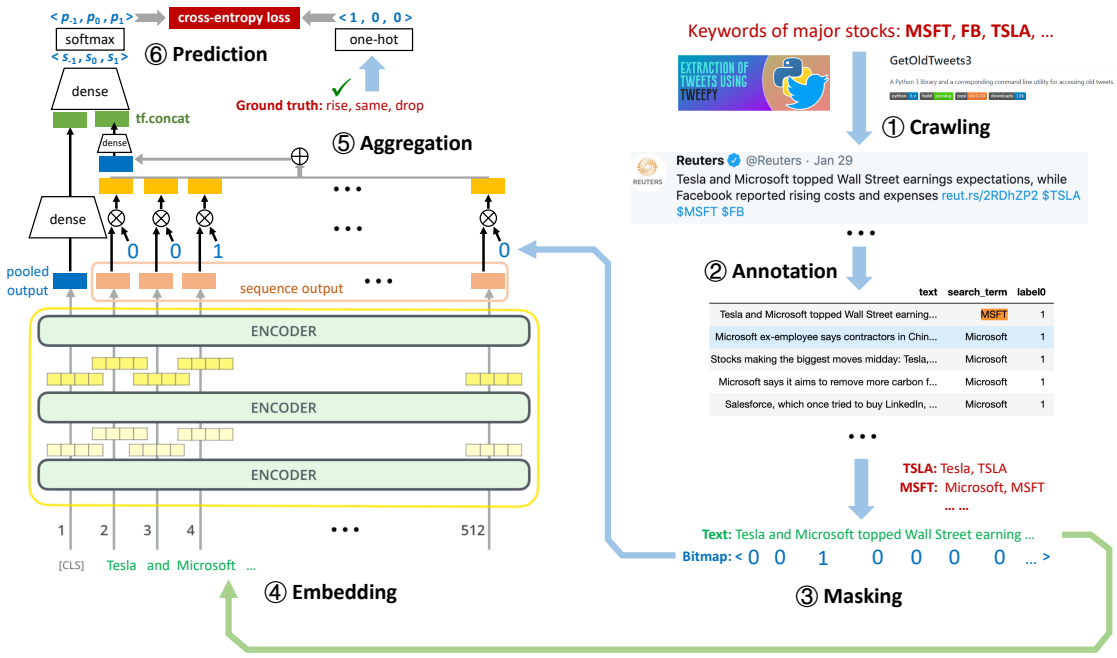

Fig. 7. The BERT-based Classifier Architecture

BERT-base over our curated news tweet dataset, the network architecture of which is shown in Figure 7. The difference is that our relevance classifier is a binary classifier judging whether a news tweet is relevant to the stock keyword that is used to crawl that tweet or not, while our sentiment classifier classifies each tweet into one of three labels: positive, neutral or negative, based on their sentiment on the mentioned stock.

As shown in Figure 7, the training workflow can be summarized as 6 steps, which we introduce next.

In the **first step**, we crawl tweets using stock keywords. For each stock, we use both its stock name and its stock code. For example, 'TSLA' and 'Tesla' are both used to crawl Tesla-related news tweets, while 'MSFT' and 'Microsoft' are both used to crawl Microsoft-related news tweets. In the **second step**, we annotate these tweets. Note that in Figure 7, each annotated tweet has its text, label, and the search term that is used to crawl it. We denote a tweet by t, denote its text by $text(t)$, its label by $label(t)$, and its search term by $term(t)$.

In the **third step**, we prepare out model input. Specifically, we first remove words that are not part of the sentence. For example, for the tweet shown in Figure 7, we remove the URL and "$TSLA $MSFT $FB" at the end. We also remove other non-word terms such as hashtags. Then, each cleansed tweet t will be tokenized into a word sequence $text(t)$ using BERT's tokenizer. We also prepare a masking bitmap for t by scanning through each word w in $text(t)$: if w matches the search term $term(t)$ or its equivalent form (e.g., 'TSLA' and 'Tesla'), the corresponding bit is set to 1; otherwise, the bit is set to 0. We denote the bitmap of t by $bitmap(t)$.

Now that we have both $text(t)$ and $bitmap(t)$ for each tweet t in the training dataset, The **fourth step** takes $text(t)$ as input to extract the sentence embedding (i.e., pooled output) and the word embedding sequence (i.e., sequence output).

These outputs are then integrated with the other model input $bitmap(t)$ in the **fifth step** to obtain the semantic features for classification. Specifically, we align the embedding of each word with the word's bit in $bitmap(t)$, and compute the average as follows:

$$subject(t) = \sum_i \Big\{ bitmap(t)[i] \cdot sequence\_output(t)[i] \Big\},$$

where $subject(t)$ essentially computes the average of all context-aware embeddings of words that match $term(t)$ in $text(t)$, and thus well represents the subject stock in the tweet. Meanwhile, we also have $pooled\_output(t)$ which well represents the entire tweet. These two embeddings are shown as blue rectangles in Figure 7.

So far, we yet have no fine-tuning using our training labels. To incorporate the impact of training labels, we transform the two BERT feature vectors $pooled\_output(t)$ and $subject(t)$ by a dense layer to obtain two relevance-specific (or sentiment-specific) semantic features. These two transformed feature vectors are shown as green rectangles in Figure 7. Finally, a dense layer takes a concatenation of both the transformed feature vectors, and converts them into a score vector whose length equals the number of classes, which can be used in the **sixth step** to choose the highest-scored class as the label.

For training purpose, both vectors are transformed into a probability vector using softmax, and then connected to cross-entropy loss to penalize its difference from the ground-truth one-hot label encoding.

Note that we actually have other options before the final dense layer besides using both the transformed feature vectors. For example, we can use the transformed features of only $pooled\_output(t)$ or $subject(t)$ alone, but our tests show that using both delivers the highest performance. In

fact, using $subject(t)$ alone often reduces the prediction accuracy by around 10%. Another option is to directly wire $pooled\_output(t)$ and/or $subject(t)$ to the final dense layer without another dense layer of feature transformation, but our test show that this method reduces the prediction accuracy.

C. Sentiment-Based Tweet Feature Extractor

Unlike our relevance classifier which is trained to emit relevance-labels for filtering purpose, the sentiment classifier is just using our annotated labels to fine-tune the model to extract the semantic features. In our final deployment, we only need the concatenated vector of the two transformed features, i.e. the green vectors shown in Figure 7, which serve as the extracted tweet features to be input as covariates to module 4.

We remark that here, the two dense layers that transforms $pooled\_output(t)$ and $subject(t)$, respectively, are important as otherwise, we will be using $pooled\_output(t)$ or $subject(t)$ directly (i.e., those blue embeddings shown in Figure 7) which can be derived from the pretrained BERT-base without any fine-tuning using our curated dataset. We find the latter solution gives lower predication accuracy which is no wonder since the features are not transformed to expose sentiments.

Once the sentiment classifier is trained, it serves as a feature extractor from those relevant tweets that pass through module 2 during our deployment of the BELT pipeline.

D. Feature integration by Autoregressive LSTM

In fact, modules 2 and 3 are both general to any stock, and when we train them, we actually replace the stock subject with [MASK] since we just want to learn if the context indicates if the mentioned stock is positive, neutral or negative, and we do not care which stock it is. In other words, we are learning a general language model that is not stock-specific.

However, our price prediction model should be stock-specific, since the property of different stocks can be very different. In other words, the price data should be trained separately on each stock for its prediction, even though the fine-tuned modules 2 and 3 are shared to extract news features.

As we have discussed in Section III-B, our price prediction is achieved by a many-to-many autoregressive LSTM network where the input to each LSTM unit includes the price on day t, i.e. x_t, along with a covariate feature vector $\mathbf{c}_t$ for day t. Here, $\mathbf{c}_t$ is given by the average vector of the transformed sentiment feature vectors extracted from all relevant tweets on day t.

We remark that while existing works on stock prediction often predicts price directions rather than the actual price, our LSTM network architecture is superior as explained in Section II. Note that after predicting the price $\hat{x}_T$, we can still get the predicted price direction $sign(\hat{x}_T - x_{T-1})$ and compare it with the actual direction $sign(x_T - x_{T-1})$. We also remark that we can easily expand $\mathbf{c}_t$ to include additional features from other data sources to further improve our performance.

V. Experiments

This section reports our experiments, which were conducted using a machine equipped with NVIDIA Tesla P100 GPUs. Every experiment is repeated for 3 times and all reported results are averaged over the 3 runs to cancel out the randomness caused by the stochastic nature of parameter initialization and mini-batch selection.

Pre-training Modules 2 and 3. We trained modules 2 and 3 over our curated news tweet dataset collected for the duration of January 1st, 2020 to April 30th, 2020, by extensively testing different model parameters, especially the number of hidden units in each feature vector transformed from $pooled\_output(t)$ or $subject(t)$.

During the training of both classifiers, we use a batch size of 32, a learning rate of 2×10^{-5}, and run for 4 epochs with a warmup proportion of 10%. The maximum sequence length of BERT is set as 128 (i.e., long tweets are truncated while zero-padding is used for short tweets).

For module 2, the relevance classifier, we use all our 5,237 tweets to find the best setting. Specifically, 30% of the tweets (i.e., 1,572 tweets) are used for testing, and among the remaining 3,665 tweets, we have 3,073 relevant (i.e., positive) tweets and only 592 irrelevant (i.e., negative) tweets. To address the class imbalance problem while utilizing all our annotation efforts, we upsample a pool of 3,073 negative tweets from the 592 irrelevant tweets to allow duplication. Training is then conducted with 1/3 of the tweets used as a validation set to avoid overfitting. We find that when the transformed feature vectors from $pooled\_output(t)$ and $subject(t)$ have 256 units each, the test accuracy is the highest, 94.7%.

The sentiment classifier in module 3 is trained similarly and we find that the best test accuracy of 84.5% is achieved when the transformed feature vectors from $pooled\_output(t)$ and $subject(t)$ have 32 units each. The smaller number of feature vector units and lower accuracy than our relevance classifier is because we dropped part of the tweets with inconsistent annotations to ensure quality but the training set size is reduced, and our sentiment classification has 3 classes while our relevance classifier only considers 2 classes. We expect the accuracy to improve further if more annotated tweets are available or richer news sources are used such as Factiva. Note that since each transformed feature vector from $pooled\_output(t)$ or $subject(t)$ has 32 units, we obtain a concatenated vector of 64 sentiment features for each news tweet which is fed into our module 4 for price prediction.

Models for Stock-Specific Prediction in Module 4. Unlike modules 2 and 3 that are general, we train a price prediction model for each stock separately since different stocks expose different properties which are reflected in their stock price curve. Albeit not the topic of this paper, there might be correlations between stocks in the same sector that can be further utilized (e.g., using graph convolutions over a stock correlation network) to improve accuracy.

Besides our autoregressive LSTM model for next-day price regression, we also consider the problem of next-day price direction prediction (i.e., classification) that are popularly studied by existing works. For this purpose, we compare with the baseline approach that transforms output features by a

dense layer into a score for the positive (i.e., price-rising) class, which is then converted to the class probability using Sigmoid and given to a binary cross-entropy loss function to close the gap with the actual price direction in the next day. This baseline utilizes the price direction $sign(x_{T+1}-x_T)$ as a classification label which is a weaker kind of supervision than regression using the actual magnitude of x_{T+1} to close the gap between it and its prediction $\hat{x}_{T+1}$. We denote our model by *REG*, and the baseline model by *CLS*. Note that REG can be used to generate price direction prediction as $sign(\hat{x}_{T+1}-x_T)$, and we expect the direction prediction to be more accurate due to the strong supervision in *REG* compared wth *CLS*.

For both *REG* and *CLS*, we further consider 2 versions, one where LSTM only takes the median share-price as input which serves as a baseline, and the other where LSTM takes both the median share-price and a covariate of sentiment features extracted from news tweets on that day. This gives rise to 4 algorithms versions to consider for price direction prediction: *REG-Price*, *REG-Sent*, *CLS-Price*, *CLS-Sent*. Here, for example, *REG-Sent* denotes the algorithm *REG* that utilizes both median share-prices and news sentiment features. For next-day price regression, only two models, *REG-Price* and *REG-Sent*, are applicable.

Data for Stock Prediction. Our stock data are purchased from kibot.com containing the price and trade volume of every stock in every minute till the current moment. We consider the duration of January 1st, 2019 to May 31st, 2020 for training and testing our model. To avoid the impact by different quarters of a year, for each quarter in 2019, the first 2 months are included in our training data, and the third month is included for testing. The same applies to 2020 except that in Q2, April is used for training and May is used for testing. We collected all the stock-related tweets from the credible sources that we introduced in Section IV-A. We choose the recent duration because there are abundant news tweets during this duration, and thus our experiments can properly reflect the impact of news tweets. This is in contrast to, say, 2008 where we found that the number of news tweets is very limited.

LSTM Model Configuration. We tested all four models, *REG-Price*, *REG-Sent*, *CLS-Price*, and *CLS-Sent*, using various configurations, and found that the following configuration consistently delivers good performance over all our tested stocks, and is thus adopted in the experiments reported hereafter. Specifically, we use a stacked LSTM architecture with 2 LSTM layers. Right after the first LSTM layer, we add batch normalization plus ReLU before connecting with the second LSTM layer. Batch normalization is used since we find it to significantly speed up convergence. The hidden unit size for the first LSTM layer is 64, and that for the second LSTM layer is 16. The test error continues to decrease as we increase the LSTM sequence length all the way up to 15 (or, 3 weeks), and thus $T = 15$ is used as the sequence length for prediction.

We set the learning rate as 0.01 and runs for 150 epochs. This setting is sufficient for all our tested models to converge.

One important preprocessing is to normalize the price data by dividing with the maximum stock price in the duration. This is because, for example, the price of AAPL (or Apple) is over $300, but our extracted sentiment features from BERT all have values either between 0 and 1, or not much larger than 1. So, for example, if we concatenate the median share-price with the 64 sentiment features extracted by module 3 to create the input vector to an LSTM unit, the input features are not of the same scale which impacts training effectiveness.

TABLE II
ROOT MEAN SQUARE ERROR

Stocks	REG-Price	REG-Sent
BA	0.051	0.041
GM	0.054	0.026
GE	0.133	0.021
AAPL	0.110	0.027
XOM	0.085	0.021
MCD	0.066	0.028
DOW	0.049	0.037

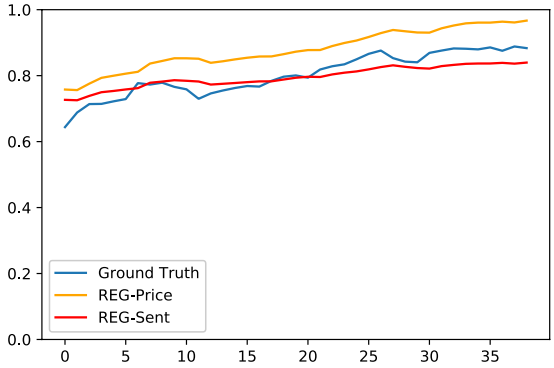

Fig. 8. Price Prediction for Apple, Inc.

Experiments on Next-Day Price Prediction. We tested the price prediction performance of our regression models *REG-Price* and *REG-Sent* over 7 popular stocks plus DOW (Dow Jones Industrial Average), and the results are shown in Table II where stocks are shown with their codes. In Table II, the errors are reported as the RMSE of price prediction on our test data, and recall that our price values have been normalized between 0 and 1. We can see that *REG-Sent* beats *REG-Price* in all cases thanks to the integration of news information for prediction. In fact, in most cases the RMSE of *REG-Sent* is many times smaller than *REG-Price*.

To visualize how both models perform, we run them to predict the next-day price for the last 40 days in our duration and plot their predicted prices in Figure 8. We can see that relies on price alone, *REG-Price* tends to predict the next-day prices higher than the actual prices. This is because the overall price trend is upward and the long sequence of upward prices makes REG-Price over confident in prediction. In fact, there are two clear price drops on day 11 and day 28 meaning that there are negative factors limiting the price growth, and these signals are captured by *REG-Sent* via the stream of news tweets. REG-Sent actually adjusted from overestimating the price towards underestimating the price (possibly because

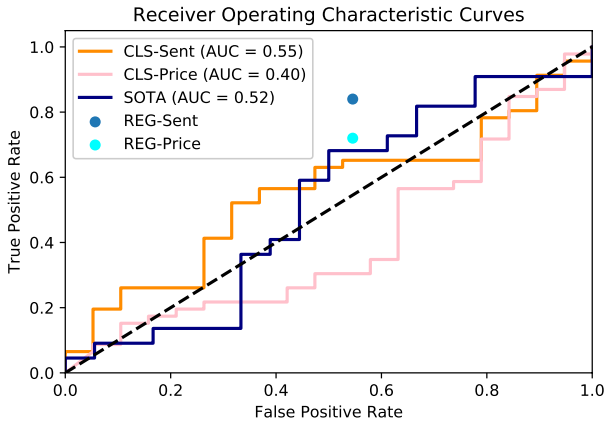

Fig. 9. ROC Curves of Different Models

the top of the price curve is going to be reached due to negative signals which can be captured by news), but error is consistently smaller than *REG-Price* in all days.

Experiments on Next-Day Price Direction Prediction. We remark that even when using *REG* models for the binary classification problem of the next-day price direction prediction, it is more accurate than the existing classification scheme as we emulated using our baseline *CLS* models. Figure 9 shows the ROC curves of *CLS-Sent*, *CLS-Price*, and *SOTA* which is StockNet [25]'s fully-equipped and best version called HedgeFundAnalyst. Since all of them obtains positive class probability using sigmoid, the ROC curves are obtained by varying the probability threshold to be positive (i.e., price-rise). *CLS-Sent* (AUC = 0.55) is better than *SOTA* since Transformers are more powerful in extracting semantic features, and StockNet's attention weights are not as effective as our relevance filter in reducing noise from irrelevant tweets. *CLS-Price* (AUC = 0.40) does not utilize news sentiments an thus perform poorly. *REG-Sent* and *REG-Price* are points since we compare $sign(\hat{x}_{T+1} - x_T)$ with $sign(x_{T+1} - x_T)$ which is deterministic. Both points are above the ROC curves showing that regression models are more effective in price movement prediction, though *REG-Sent* is much better as it considers tweets sentiment.

VI. Conclusion

This paper proposed an objective way of integrating news signals streamed from Twitter into an autoregressive LSTM model that predicts the stock price for end-to-end training. The end-product is a predictive pipeline called BELT which is effectiveness of BELT in stock price prediction.

Acknowledgements. This work is partially supported by NSF DGE-1723250.

References

[1] A Hundred Most Popular Stocks. https://robinhood.com/collections/100-most-popular.

[2] Fifty Most Important People for Investors to Follow. https://www.marketwatch.com/story/finance-twitter-the-50-most-important-people-for-investors-to-follow-2018-12-13.

[3] Thirty Must-Follow Twitter News Accounts in 2019. https://medium.com/@intellfusionmarketing/30-must-follow-twitter-accounts-for-news-in-2019-e981595759a5.

[4] W. Bao, J. Yue, and Y. Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS one*, 12(7):e0180944, 2017.

[5] S. Borovkova and I. Tsiamas. An ensemble of lstm neural networks for high-frequency stock market classification. *Journal of Forecasting*, 2019.

[6] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

[7] A. Conneau and G. Lample. Cross-lingual language model pretraining. In *NeurIPS*, pages 7057–7067, 2019.

[8] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *ACL*, pages 2978–2988, 2019.

[9] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186, 2019.

[10] B. Ellickson, M. Sun, D. Whang, and S. Yan. Estimating a local heston model. *SSRN 3108822*, 2018.

[11] V. Flunkert, D. Salinas, and J. Gasthaus. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *CoRR*, abs/1704.04110, 2017.

[12] J. Gasthaus, K. Benidis, Y. Wang, S. S. Rangapuram, D. Salinas, V. Flunkert, and T. Januschowski. Probabilistic forecasting with spline quantile function rnns. In *AISTATS*, pages 1901–1910, 2019.

[13] Y. Gu, D. Yan, S. Yan, and Z. Jiang. Price forecast with high-frequency finance data: An autoregressive recurrent neural network model with technical indicators. In *CIKM*, 2020.

[14] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[15] T. Kim and H. Y. Kim. Forecasting stock prices with a feature fusion lstm-cnn model using different representations of the same data. *PloS one*, 14(2):e0212320, 2019.

[16] N. Kitaev, L. Kaiser, and A. Levskaya. Reformer: The efficient transformer. In *ICLR*, 2020.

[17] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *ICLR*, 2020.

[18] A. Mittal and A. Goel. Stock prediction using twitter sentiment analysis. *Standford University, CS229*, 15, 2012.

[19] Z. Qi, Z. Bu, X. Xiong, H. Sun, J. Cao, and C. Zhang. A stock index prediction framework: Integrating technical and topological mesoscale indicators. In *IRI*, pages 23–30, 2019.

[20] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. 2018.

[21] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

[22] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski. Deep state space models for time series forecasting. In *NeurIPS*, pages 7796–7805, 2018.

[23] T. T. Souza and T. Aste. Predicting future stock market structure by combining social and financial network information. *Physica A: Statistical Mechanics and its Applications*, 535:122343, 2019.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017.

[25] Y. Xu and S. B. Cohen. Stock movement prediction from tweets and historical prices. In I. Gurevych and Y. Miyao, editors, *ACL*, pages 1970–1979. Association for Computational Linguistics, 2018.

[26] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, pages 5754–5764, 2019.

[27] L. Zhang, P. A. Mykland, and Y. Aït-Sahalia. A tale of two time scales: Determining integrated volatility with noisy high-frequency data. *Journal of the American Statistical Association*, 100(472):1394–1411, 2005.

[28] Z. Zhang, S. Zohren, and S. J. Roberts. Deeplob: Deep convolutional neural networks for limit order books. *IEEE Trans. Signal Processing*, 67(11):3001–3012, 2019.