CS230

Predicting Stock Price Movement with Event-Driven Deep Learning Approach

Yuan Fang yuanfy@stanford.edu
Shelly Wang shelly29@stanford.edu
Yitian Zeng ytzeng1@stanford.edu

Abstract

This project uses NLP methods to forecast stock movements with financial news headlines, with the focus on input format. Whereas the majority of the work in this area resorts to simple text features, we use structured events representations and other mature NLP embedding methods to preprocess our input data. We also try a few CNN- and RNN-based model architectures. With Smooth Inverse Frequency embedding and LSTM network, we achieved test accuracy roughly comparable with Ding et al. [2015], the paper we set out to replicate and expand upon.

1 Introduction

Predicting stock prices is of central interest to investment professionals as well as academic researchers. Predicting stock movements, however, is no easy task. Stock prices are incredibly noisy, and a multitude of information sources can affect their trajectories.

One central tenet in financial theory—the Efficient Market Hypothesis (Fama [1965])—states that, at any given point in time, stock prices reflect all available information; prices only change whenever new information becomes available, and they do so immediately. The Efficient Market Hypothesis thus offers theoretical support for the value of news as a potential source of signals, with recent technological developments only further fueling the attention towards financial news.

In this project, we aim to build a non-traditional (with regard to input format) neural network NLP model that predicts the directions of stock price movements with financial news titles.

2 Related work

Traditional NLP approaches rely on simple features such as bag-of-words, noun phrases, and named entities from text documents to predict stock-related quantities (Kogan et al. [2009]; Schumaker and Chen [2009]). However, such techniques fail to capture the structured relations within texts, as initiators of actions are not differentiated from their subjects.

In this project, we intend to capture structured relations in news, building upon the work of Ding et al. [2015], who found the usage of structured events representations (i.e. word embeddings) to significantly improve the performance of predictive models. We hope to replicate and expand upon the work done in this paper, but instead of moving to event embeddings like the authors, we experiment with other mature NLP embedding methods.

3 Dataset and Features

We used financial news from Reuters and Bloomberg from October 2006 to November 2013, released by Ding et al. [2014]. We focus on new headlines because Radinsky et al. [2012] shows they are

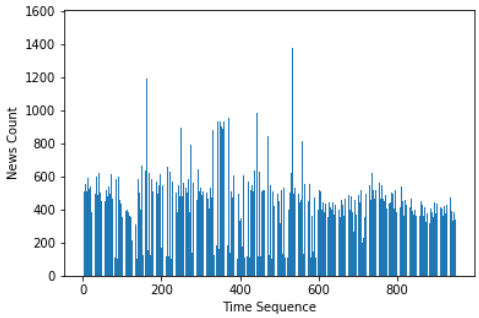

Figure 1: Amount of news headlines from time period 2010-04-23 to 2013-11-25.

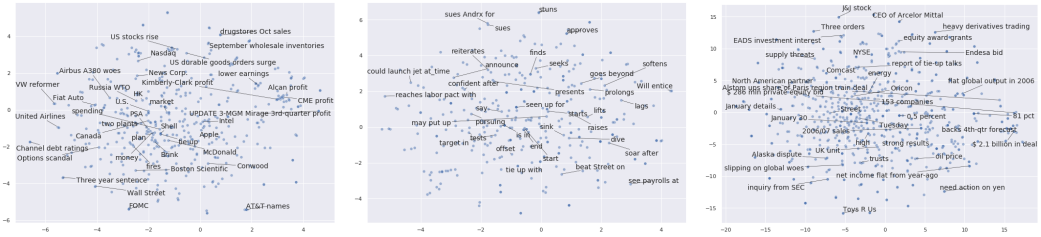

Figure 2: Word2Vec Embeddings in 2D. Left is the t-distributed Stochastic Neighbor Embedding (TSNE) of the subject (actor). Middle is the relation embedding (action) and on the right is the object embedding.

more helpful for prediction tasks. A subset of the distribution of headline counts is shown in Figure 1. The average number of headlines published during each business day is around 500, while that for an average weekend day is around 20. Since RNN requires datasets to have the same number of time steps in each batch, and there was no corresponding stock price during weekends, headlines published on weekends were not used in RNN models. The price trajectories of the Standard Poor's 500 index from the same period, from which we extract the Y input of the deep learning process, were obtained from the quantmod R package by Ryan and Ulrich [2019].

4 Methods

4.1 Headline Embedding

Embedding sentences into vectors while still preserving useful information is the core of NLP. In this work we used open source pre-trained models such as Word2Vec, Bidirectional Encoder Representations from Transformers (BERT), Smooth Inverse Frequency (SIF) and Universal Sentence Embedding (USE) for quick deployment and implementation.

4.1.1 Word2Vec

We first used open information extraction (Open IE) implemented in the Stanford CoreNLP framework [Manning et al., 2014] to obtain structured events representations, formulated as tuples of (Actor, Action, Object, Timestamp), that can represent more accurate and complex information.

A pre-trained 300 parameters Word2Vec model based on Google News (Mikolov et al. [2013]) was then used to encode the event tuples into vectors. We used t-distributed Stochastic Neighbor Embedding (TSNE) to convert the 300-dimensional vectors into 2D for visualization. The top 1000 results are plotted in Figure 2. We can observe some desirable behaviors: 'rise' and 'surge', 'Canada' and 'U.S', 'dive' and 'sink' are quite close to each other. It shows that a simple Word2Vec can preserve certain features of the word and reduce the sparsity of the input data.

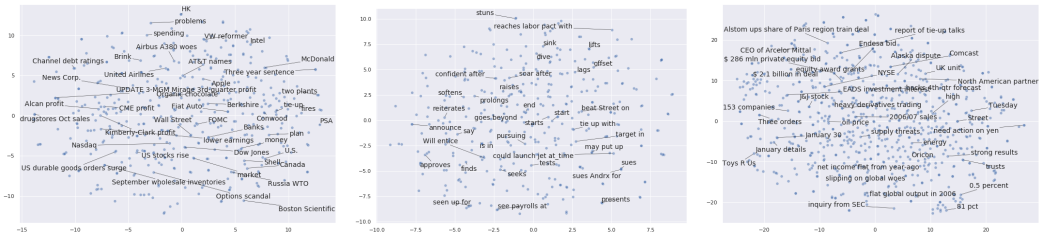

Figure 3: USE Embeddings in 2D. Left is the t-distributed Stochastic Neighbor Embedding (TSNE) of the subject (actor). Middle is the relation embedding (action) and on the right is the object embedding.

4.1.2 BERT

BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. BERT is a deeply bidirectional, unsupervised language representation. For deployment, we used open source bert-as-service Devlin et al. [2018] python implementation to map a variable-length sentence to a vector with 768 features.

4.1.3 SIF

SIF, developed by Arora et al. [2016], involves computing weighted average of word vectors in the sentence based on word frequency and removing projections of the average vectors on their first singular vector. For this project we use the implementation of SIF in the *fse* package developed by Borchers [2019], which maps each headline to a 300-dimensional vector.

4.1.4 USE

USE, developed by Cer et al. [2018], is an embedding method that targets transfer learning to other NLP tasks. This project utilizes the implementation of USE in tensorflow hub to map each headline to a 512-dimensional vector. The same visualizing exercise previously conducted with Word2Vec is repeated with USE; the result is shown in Figure 3.

4.2 Baseline Method

With the averaged word embeddings as input X with 300 features (i.e. length of vectors produced by Word2Vec) and the corresponding stock price trend as Y, we implemented a simple logistic regression as our baseline model. This model was trained over 1800 samples and 100 iterations. We achieved a training accuracy of 0.69 and test accuracy of 0.58. Overall, it shows that there are certain relations between financial news and stock price and therefore we can do better than random guessing.

4.3 Convolutional Neural Network

Before feeding our inputs into our models, we have one remaining problem: Because the volume of news each day is not constant, input size differs from sample to sample. Following the solution offered by Ding et al. [2015], for each day in our sample, we summed up the vectors converted to from each word in the word embedding, and averaged the resulting sum across all embeddings associated with that day.

The first CNN model we tried (hereafter called CNN1) takes a sequence of word embeddings, consisting of long-term words embeddings (past 30 days), mid-term words embeddings (past 7 days), and short-term words embeddings (past day). The long-term and mid-term words embeddings are fed into a 1D convolutional layer (*filter size f=3, stride s=1, padding p=same*) and then a max pooling layer to obtain V^l and V^m, respectively. The model performs an average pooling on short-term words to get V^s, and it then feeds $V^c = (V^l, V^m, V^s)$ into a hidden layer with 100 neurons, which is the second last layer. Then, given the information of the hidden layer, the output layer gives a prediction which is either 1 meaning the stock price increases, or 0 meaning the stock price decreases. However, this model gives a poor performance regardless of the embedding method. The best test accuracy, which we got with SIF embeddings, was 0.46.

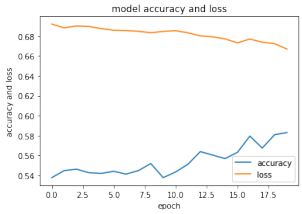

Figure 4: Training accuracy and loss for CNN2 when using SIF word embeddings

Table 1: Test confusion matrix for CNN2 when use SIF word embeddings

Confusion matrix	Predicted: 0	Predicted: 1
Actual: 0	2	103
Actual: 1	1	145

Due to the poor performance of CNN1, we decided to adjust the architecture and got our second CNN model (hereafter called CNN2). The input to CNN2 is a list of word embeddings of the past 20 days. CNN2 consists of a 2D convolutional layer (*filter size (f1,f2)=(3,size of word embeddings), stride s=1, padding p=valid, channels=64*), a 2D average pooling layer with pool size=(3,1), two hidden layers with dropout=0.4 and 0.2, respectively, and an output layer which outputs the probabilities of price increase and decrease. We use different word embeddings to train and test the model, and the result is as follows.

Embedding	Accuracy	Word2Vec	BERT	SIF	USE
CNN2	Training	0.55	0.55	0.58	0.56
	Test	0.56	0.48	0.59	0.57

We got the best test accuracy of 0.59 with SIF embeddings; corresponding confusion matrix and "accuracy and loss" graph are shown in Figure 4 and Table 1. The accuracy measures of CNN1 and CNN2 show that CNN models able to learn the connection between news headlines and stock price.

4.4 Recurrent Neural Network

Due to the sequential nature of RNN models, individual news headline embeddings were preserved as inputs. In particular, the input datasets were reshaped to (number of days, time steps in a day, sentence embedding feature size) 3D matrices. However, RNN requires time steps (axis = 1) to have the same dimension, that is, the amount of news should be the same everyday. Therefore, the average business day headline count of 500 was chosen as the time steps in a day. For days with fewer than 500 headlines, zeros were padded behind and for days with more than 500 headlines, only the first 500 were chosen.

Long short-term memory (LSTM) and gated recurrent unit (GRU) are two major building blocks of RNN. In this section, one layer of LSTM, bidirectional LSTM, GRU, bidirectional GRU, and cnn-LSTM were implemented using keras and trained on BERT, SIF and USE embeddings to establish baseline of RNN. For each baseline model, the latent-dim of recurrent unit is 128 followed by two fully connected layer and trained over 15 epochs. The training and test accuracy is summarized in the table below. SIF embedding, although with smallest feature size, has generally the best performance across different RNN units, and we were able to overfit SIF embeddings if trained for longer epochs (note that bidirectional models all used L2 regularization to prevent overfitting). Additionally, we observed that cnn-LSTM has the fastest training time due to 1D convolution significantly reducing feature size. Furthermore, increasing the number of recurrent neural cells (we tested from 2 to 10 LSTM cells) has minimum impact on metrics if properly regularized; therefore we only used one layer of RNN cell as building blocks for more complex models.

Embedding	Accuracy	LSTM	Bi-LSTM	GRU	Bi-GRU	cnn-LSTM
BERT	Training	0.67	0.55	0.71	0.75	0.52
	Test	0.59	0.46	0.59	0.38	0.59
SIF	Training	0.79	0.55	0.86	0.54	0.54
	Test	0.56	0.59	0.60	0.59	0.59
USE	Training	0.83	0.55	0.83	0.54	0.55
	Test	0.56	0.59	0.56	0.59	0.59

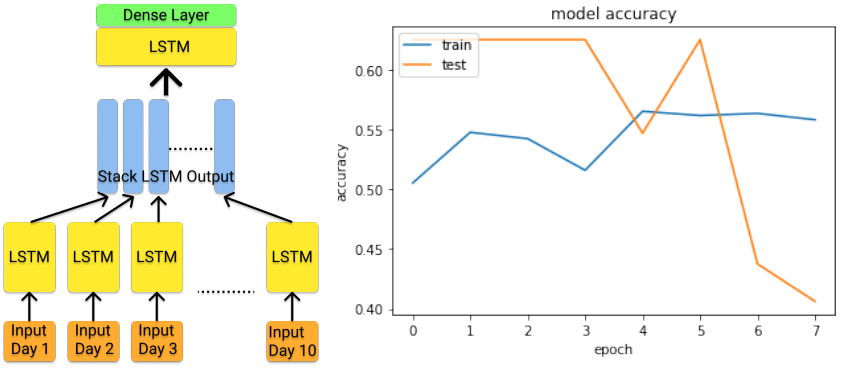

Figure 5: Complex LSTM model with 10 days input data for stock price prediction. Note only the last fully connected layer is shown in the figure.

Our best performing RNN model is built upon LSTM cell and illustrated in Figure 5. The input shape of the model (10, 500, 300) is equivalent to 10 consecutive days, 500 news headlines each day with 300 features for each headline. Each day is fed into a bidirectional LSTM cell with increasing number of hidden units from day 1 to day 10. The output of each LSTM cell is flattened out and stacked into another 3D dataset and fed into a new LSTM layer and finally connected to a fully connected layer and output as binary classification problem.

5 Results and Discussions

The results of our models show that there is a close relationship between financial news and stock price. The test accuracy of our logistic regression model is 0.58, and we assume the accuracy is not high mainly due to that it is a fairly simple model. The best test accuracy of CNN1 is 0.46 and the best word embeddings for CNN1 is SIF; we believe that CNN1 performs poorly because redundant information in the input distracts CNN1 from making good predictions. We thus came up with CNN2, which only uses the past 20 days' financial news as input, instead of using long-term, mid-term, short-term news as for CNN1. The best test accuracy of CNN2 is 0.59, given SIF word embeddings as input. As we can see from the confusion matrix and the accuracy and loss graph (Table 1 and Figure 4), CNN2 does learn from the training data; however, the trained CNN2 tends to predict that the stock price as increasing, and this could be due to a lack of cleaned training data.

For our best-performing RNN model, we achieved a training accuracy of 0.56 with regularization and a test precision of 0.64 for positive class and 0.49 for negative class. The performance of this model is on par with published work Ding et al. [2015], and shows that RNN is able to learn the connection between financial news and stock price to some degree. However, due to the large number of trainable parameters in RNN models, our training data is not sufficient and we are constantly overfitting the model and yielding less satisfactory results. Accumulating a larger dataset would alleviate this issue.

6 Conclusion and Next Steps

The highest test accuracy we have is comparable to the performance of the model obtained by Ding et al. [2015]. There are mainly three steps that can be undertaken to further improve the accuracy. First, we can remove irrelevant data from and expand our dataset. For example, our data set now contains non-financial news, which doesn't have a strong relation with stock price movements. Larger training dataset is also desirable for larger neural networks to reduce the overfitting problem. Second, we can experiment with more complex model architectures. One possible idea is to combine CNN2 with GRU. Lastly, we can also further tune hyperparameters and input features to train aggressive and preservative models.

7 Contributions

Each contributor is responsible for the code as well as associated sections on the report and poster for each component of the project.
YF: CNN models
SW: Preparing data labels; Open information extraction; SIF embedding; USE embedding
YZ: Initial data wrangling; Word2Vec; BERT embedding; RNN models

8 GitHub and YouTube Links

GitHub: `http://bit.ly/stockpricedl`
YouTube: `https://youtu.be/EowN_DrDjIk`

References

S. Arora, Y. Liang, and T. Ma. A simple but tough-to-beat baseline for sentence embeddings. 2016.

O. Borchers. Fast sentence embeddings. `https://github.com/oborchers/Fast_Sentence_Embeddings`, 2019.

D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

X. Ding, Y. Zhang, T. Liu, and J. Duan. Using structured events to predict stock price movement: An empirical investigation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1415–1425, 2014.

X. Ding, Y. Zhang, T. Liu, and J. Duan. Deep learning for event-driven stock prediction. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

E. F. Fama. The behavior of stock-market prices. volume 38, pages 34–105. JSTOR, 1965.

S. Kogan, D. Levin, B. R. Routledge, J. S. Sagi, and N. A. Smith. Predicting risk from financial reports with regression. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 272–280, 2009.

C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014. URL `http://www.aclweb.org/anthology/P/P14/P14-5010`.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

K. Radinsky, S. Davidovich, and S. Markovitch. Learning causality for news events prediction. In *Proceedings of the 21st international conference on World Wide Web*, pages 909–918, 2012.

J. A. Ryan and J. M. Ulrich. *quantmod: Quantitative Financial Modelling Framework*, 2019. URL `https://CRAN.R-project.org/package=quantmod`. R package version 0.4-15.

R. P. Schumaker and H. Chen. Textual analysis of stock market prediction using breaking financial news: The azfin text system. *ACM Transactions on Information Systems (TOIS)*, 27(2):1–19, 2009.