CS230

Stock Price Predictions from News Headline Embeddings

Ryan Almodovar
Department of Computer Science
Stanford University
ralmodov@stanford.edu

Abstract

The financial market is known to be informationally efficient, so stock prices reflect all known information and price movement can be in response to news or events [Fama, 1965]. Several Natural Language Processing (NLP) techniques have been applied over the recent years to explore financial news for predicting market volatility, such as bags-of-words, noun phrases, named entities, and sentiment analysis [Kogan et al., 2009; Schumaker and Chen, 2009]. For this project, we attempt to predict the SP 500 index stock price changes for upcoming days based on published headlines by exploring and applying some recent novel NLP architectures focused on embedding vector representing words or sentences, such as BERT [Devlin et al, 2018], Universal Sentence Encoder [Yang, 2018], and Word2Vec [Mikolov, 2013].

1 Introduction

The ability to predict stock prices has been a great interest for public companies and investors to be able to adjust to market volatility and manage portfolio profits and losses. The interest and motivation for this task is that predicting the stock market accurately was thought to be near impossible based on its stochastic nature (Malkiel, 1973), though with the advent of AI applications and Deep Learning, we may be able to gradually discover features that can guide our understanding of this system that were previously thought unattainable. The goal is to predict the change of S&P 500 index based on recent published headlines on a day-to-day basis. The input to the algorithm are collections of news headlines, and the training set combines them with labels if the stock increased or decreased on the day they were published. Depending on the model described in the Methods section (Word2Vec, BERT, Universal Sentence Encoder), the headlines are transformed into word/sentence vector embeddings and used as input to the model. Each of the models map the output to the probability range [0,1], where a 0 indicates that the SP 500 index is expected to decrease for the day, or 1 for an expected increase.

2 Related work

Several works have been published with interest in this problem, or similarly predicting the prices of individual companies rather than the general S&P 500 index. For example, Ding et al. used the Neural Tensor Network model to create trainable event embeddings with entities. Chen et al. separated news into categories based on earning rates, then predicted which category will have future price movement. Bao et al. decompose the noise using wavelet transforms, apply stacked autoencoders to generate most important features and then apply the output to an LSTM. Qin et al. uses a dual stage

RNN model for time series prediction, though applying 81 stocks' previous prices as input. Ryo et al. create a prediction based on numerical and textual information, using both an RNN and LSTM to capture significant events. The majority of these listed are based on time series based architectures such as LSTMs and RNNs to interpret statistical data, though only a few which use the generalized semantic embedding for headlines.

3 Dataset and Features

From the UCI News Aggregator Dataset provided by Kaggle [source], the data was first preprocessed by filtering invalid characters in each sentence. It was then filtered to be categorized only by business and technology sources. Downloaded SP 500 historical data from Yahoo Finance between 03/10/2014 and 10/01/2014 to match recorded news dates from the UCI News Aggregator Dataset (UCINA). The timestamp for each row in the UCINA was labeled in milliseconds, so a preprocessing step of each entry into the date format was required to be compatible with the stock data. From the stock data, binary classification labels were generated for each day by computing the opening and closing price delta values for each day, defined by:

$$y^{(i)} = \mathbb{1}\{D^{(i)}_{open} - D^{(i)}_{close}\} \tag{1}$$

Once the labels were generated, the data was then shuffled using a fixed random seed for reproducibility, and then split to create corresponding train, validation (dev), and test sets. The split percentages for the train, dev and test sets were 80%, 10%, 10% and resulted in 146,731, 18,342, and 18,343 total rows for each respective subset. To label the day offset variants, the rows required to be filtered even further since n days in advance may not have been present in the finance dataset. The headlines were then supplied as input for each model which subsequently would process them as embeddings respective to there architecture, as described in the following section.

4 Methods

Since text embeddings have become ubiquitous in modern NLP applications, the algorithms explored for this project mainly focus on the variations of the alternate embedding methods. The implementations that were explored included Word2Vec (W2V), Universal Sentence Encoder (USE), and the state-of-the-art Bidirectional Encoder Representations from Transformers (BERT). GloVe was also added but due to time constraints there was not enough time to get a working implementation and evaluate the results before the deadline. These models were chosen based on the idea of how different methods for text embedding can influence the outcome of this classification task. For example, Word2Vec is based on unidirectional word embedding, USE is sentence embedding, and BERT is the bidirectional sum of each token, segment, and positional embeddings.

The goal is to predict whether the market price either decreases or increases for the day given a set of headlines, so the problem can be defined as a binary classification task. Thus, the binary cross entropy loss function was used for the Word2Vec and USE models. The default implementation of the BERT model was implemented only with a categorical cross entropy softmax classifier, though by creating a custom module to evaluate N=2 categories, it effectively has the same outcome as a binary cross entropy sigmoid classifier. The binary cross-entropy loss function for the output layer of each model implemented is defined as:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\ \log(\sigma_\theta(z^{(i)})) + (1-y^{(i)})\ \log(1-\sigma_\theta(z^{(i)}))] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2 \tag{2}$$

where $z^{(i)} = W^T x^{(i)} + b$, $\sigma_\theta(z^{(i)}) = \frac{1}{1+exp(-(z^{(i)})}$, m is the total number of samples, λ is the standard L2 regularization rate, and θ are the trainable weights of the model. Adam (Kingma, 2014) was the main optimizer used for each model.

4.1 Word2Vec Overview

For the Word2Vec (W2V) model, I explored different variations as described in the subsections below, including a baseline as a single densely connected layer defined by a ReLU activation function, without any pre-trained vectors. The headlines were first tokenized to remove stop words and invalid

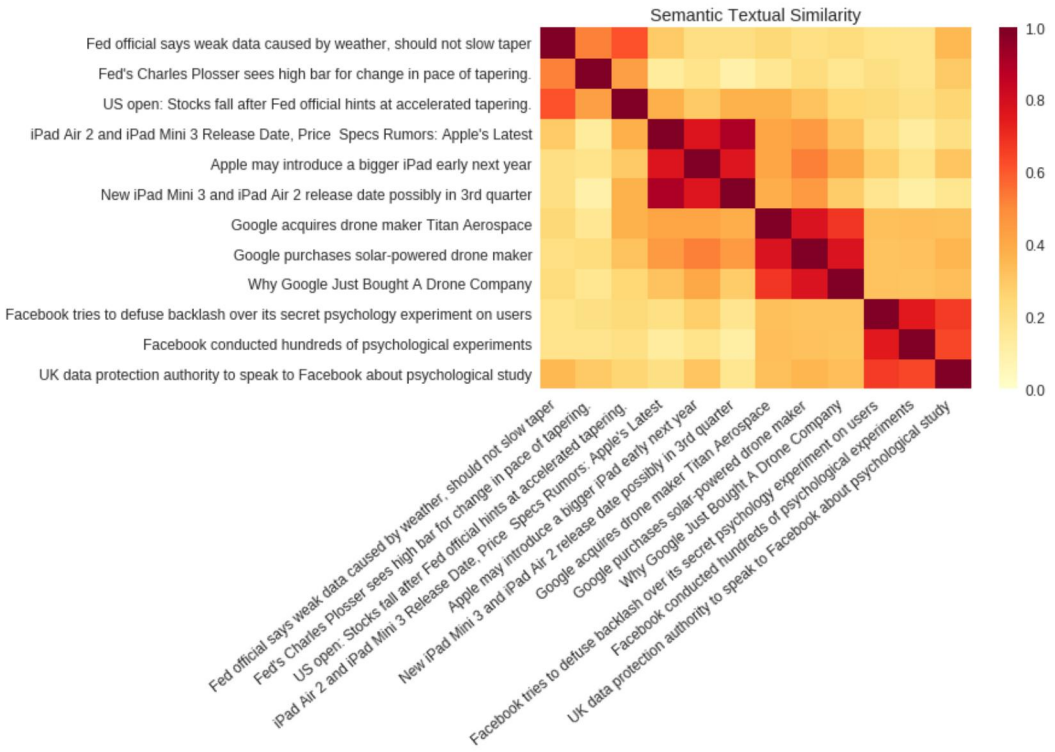

Figure 1: Headline Semantic Similarity

characters. Next, the texts were transformed into sequences of integers and padded to match the embedding size sequence length.

4.1.1 Pre-trained Variation

Initializing word vectors with those obtained from an unsupervised neural language model is a popular method to improve performance in the absence of a large supervised training set (Collobert et al., 2011; Socher et al., 2011; Iyyer et al., 2014). For this variation, word vectors that were trained on 100 billion words from Google News have dimensionality of 300 and were trained using the continuous bag-of-words architecture (Mikolov et al., 2013). Words not present in the set of pre-trained words are initialized randomly.

4.1.2 Pre-trained + CNN Variation

In addition to the embeddings from the pre-trained variation, I also included a CNN with max-pooling for sentence classification (Kim, 2014). A convolution operation involves a filter $\boldsymbol{w} \in R^{hk}$ which is applied to a window of h words to produce a new feature. For example, a feature c_i is generated from a window of words $x_{i:i+h-1}$ by $c_i = f(w\Delta x_{i:i+h-1} + b)$ where f is the hyperbolic tangent. The filter is then applied to each window of words in a sentence $\{x_{1:h}, x_{2:h+1}, ..., x_{nh+1:n}\}$to produce a feature map $c = [c_1, c_2, ..., c_{nh+1}]$. Next, a max pooling layer (Collobert et al., 2011) is applied to obtain $\hat{c} = max\{c\}$ which represents capturing the most important feature for each feature map. Finally, the output of the max pooling layer is the applied to a single fully connected layer with a sigmoid activation to obtain the prediction.

4.2 BERT Overview

BERT (Devlin et al. 2018) is designed to pre-train deep bidirectional representations by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT representations can be fine-tuned with just one additional output layer to create state-of-the- art models for a wide

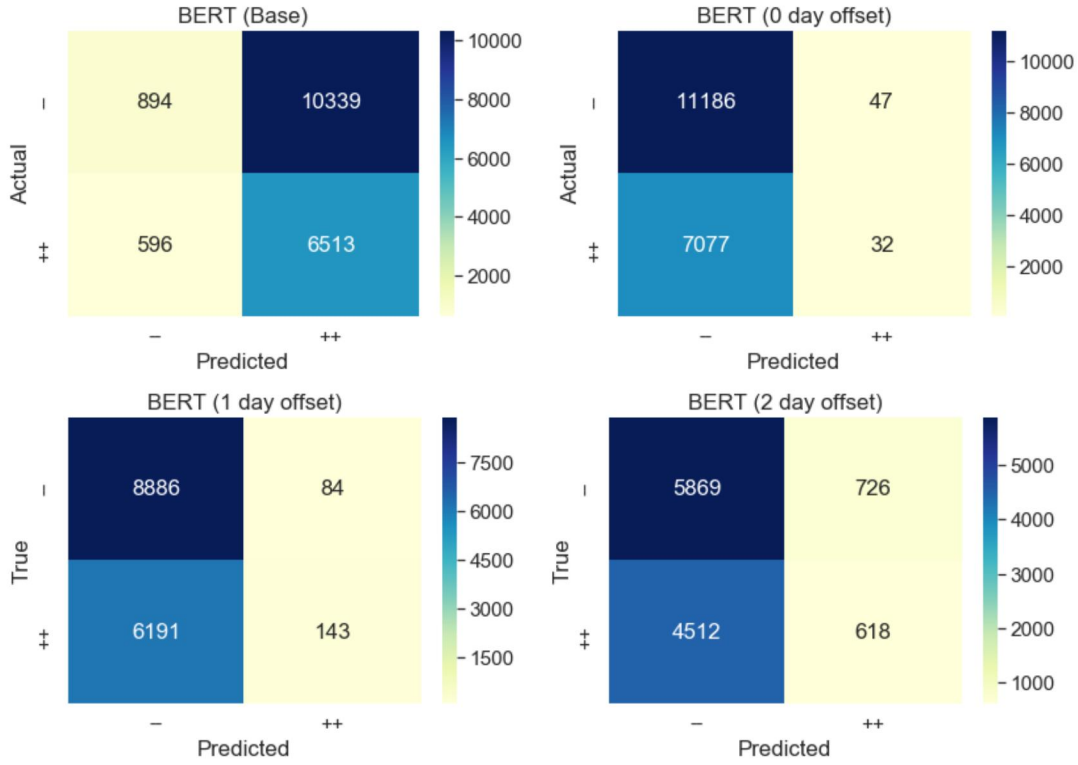

Figure 2: BERT confusion matrices: fig:ex3-a describes the first subfigure; fig:ex3-b describes the second subfigure; fig:ex3-c describes the third subfigure; and, fig:ex3-d describes the last subfigure.

range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications. The pre-trained BERT-Base model as described in the paper is evaluated in this project, as well as fine-tuned variations of the labels separated by same day, next day, and 2 day offsets.

4.3 Universal Sentence Encoder Overview

The headlines were extracted as input for the Universal Sentence Encoder which uses a Transformer architecture (Vaswani et al., 2017). The context aware word representations are converted to a fixed length sentence encoding vector by computing the element-wise sum of the representations at each word position. The encoder takes as input a lowercased PTB tokenized string and outputs a 512 dimensional vector as the sentence embedding. Using cosine similarity between two embeddings $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$cos(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{||\boldsymbol{x}|| \cdot ||\boldsymbol{y}||} \tag{3}$$

we have insight of the headline semantic similarity as seen in the figure. 4

5 Experiments/Results/Discussion

The hyperparameters that were modified for the different variations include the learning rate, batch size, number of training epochs, and dropout, as well as the shift of label offsets by 0, 1, and 2 days based on the reasoning that headlines may take some time before being read and understood by stock traders to have enough influence on the market. Due to time constraints and memory issues on the AWS EC2 instance, not all variations of the hyperparameters were able to be experimented on for each of the 3 different models, so the best performing variants after few test runs were chosen as the hyperparameter values as detailed in Table 1.

Model	Variation	Precision	Recall	Accuracy	ROC AUC
Word2Vec	Base	25.67	31.67	52.19	48.73
	Pre-trained (Google News dataset)	39.21	0.76	58.31	50.07
	Pre-trained+CNN+Dropout(0.25)+L2(0.01)	42.34	1.2	62.07	50.80
BERT	Base (cased, same day)	38.65	91.62	40.38	49.79
	Fine-Tuned (same day)	40.5	0.4	61.16	50.02
	Fine-Tuned (next day)	63.00	2.26	59.00	50.66
	Fine-Tuned (2 day)	45.98	12.05	55.33	50.52
USE	Base	-	-	-	-
	Dropout (0.2) + L2 (0.001)	-	-	-	-

Table 1: The variations include hyperparameter tuning, and submodels from the core Model. The Universal Sentence Encoder results were unable to be retrieved due to time constraints and out of memory errors while running on the AWS EC2 instance.

The confusion matrices listed for each of the BERT day offsets reveals that each fine-tuned model on the training data tends to predict the market will decrease much more often than increase. The matrix for the BERT-Base however tends to over predict that market will increase. This can be due to the thousands of negative headlines being combined for each single day, and with the current implementation, each headline may have an equal effect of influencing the stock market rather than being properly weighted in comparison with more influential entities. Interestingly, it appears that the precision evaluation for next day label offsets on the BERT model increased drastically from 40.5 to 63.0, indicating that news headlines may have more influence on the day after it is published.

Although each model had a relatively smooth loss and accuracy progression on the training set, almost all of the models experienced significant overfitting on the training data since the loss value on validation set gradually increased over each iteration. To mitigate this, I added dropout layers and L2 regularization terms in each densely connected layer and after the max pooling layer of the CNN word2vec model variation. This appeared to have a small regularization effect since the validation accuracy increased slightly, but was not significant to overcome the overfitting problem in general.

.

6 Conclusion/Future Work

The algorithm that appears to be the highest performing by a slight margin is the Word2Vec model combined with the CNN and pre-trained word vectors from the Google News dataset. This is most likely the case since the pre-trained vectors were sourced from Google News and most likely contained feature similarities to the UCI News Aggregator dataset. Although BERT and Universal Sentence Encoder are also pre-trained, it could be that the headline structure differs grammatically from typical sentences of the much larger corpus that these models were pre-trained on. For future work, if I had more time I would also want to augment BERT and Universal Sentence Encoder with more advance models on the embedding input vectors, since it seemed to improve the accuracy of the word2vec model by about 3%. I would also look into factoring in other features rather than only the label, such as the delta values, time series analysis, or the categorized hostnames and draw further insights by incorporating those features in the overall model.

Github link: https://github.com/rjalmo/cs230

References

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

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

Cer, D., Yang, Y., Kong, S. Y., Hua, N., Limtiaco, N., John, R. S., ... Sung, Y. H. (2018). Universal sentence encoder. arXiv preprint arXiv:1803.11175.

Ding, X., Zhang, Y., Liu, T., Duan, J. (2015, June). Deep learning for event-driven stock prediction. In Twenty-Fourth International Joint Conference on Artificial Intelligence.

Chen, K., Zhou, Y., Dai, F. (2015, October). A LSTM-based method for stock returns prediction: A case study of China stock market. In 2015 IEEE International Conference on Big Data (Big Data) (pp. 2823-2824). IEEE.

Collobert, Ronan, et al. "Natural language processing (almost) from scratch." Journal of machine learning research 12.Aug (2011): 2493-2537.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In Advances in Neural Information Processing Systems (pp. 5998-6008).