A text-based forecasting model for equity trading

Oleg Mitsik
Department of Computer Science
Stanford University
Stanford, CA 94305
omitsik@stanford.edu

Abstract

Equity analysts develop forecasting models to predict stock returns. These models are typically based on a number of numerical features, such as financial indicators, profitability margins and growth rates. However, a lot of price sensitive information is contained in a textual form, e.g. public announcements, press releases, news and analyst opinions. This paper suggests a simple neural architecture, based on attention-weighted BiLSTM encoder, which allows to easily incorporate textual features into a forecasting model. Because there are no straightforward baselines for this problem, the suggested architecture is first validated on CoNLL-2003 dataset for the NER task, but then it is tested on a real trading data collected for 44 blue-chip stocks. I find that the suggested forecasting model can increase predictive power by incorporating stock-related textual information, though the incremental effect largely depends on the quality of provided textual inputs.

1 Introduction

1.1 Background

The primary goal of this work[1] is to suggest a flexible framework for incorporating textual features into forecasting models, specifically stock return prediction models. This is a quite hot problem, as many fund managers claim that they use natural language processing to better manage their investment portfolios. However, they often restrict their attention to the sentiment analysis task (measuring sentiment with regard to a particular stock in news and social media). I believe this approach is limited, because a lot of price sensitive information may have neutral sentiment or because public opinion may be biased and not substantiated by cash flow analysis. Thus, a better approach could be applying the natural language understanding task instead of the sentiment analysis task. This can be achieved by using a modern neural network architecture which is trained to directly predict stock returns from textual inputs.

1.2 Related work

A possibility of using language features to predict financial market trends has been researched by scientific community for a while. Engelberg [1] used typed dependency parsing to show that qualitative earnings information has additional predictability for asset prices beyond the predictability in quantitative information. He also showed that linguistic information is likely to have greater long-term predictability for asset prices than quantitative indicators. Vivek Sehgal et al. [2] introduced a method to predict stock market using sentiment analysis. The method was based on scanning for financial message boards, extracting sentiments expressed by individual authors, and estimating

[1]The program code and constructed dataset are available at http://github.com/OlegMitsik/NLP_forecasting

Stanford / CS224n: Natural Language Processing with Deep Learning. Winter 2019.

correlation between the sentiments and stock prices. The method was able to predict the sentiment with a high precision and the performance of stocks turned out to be correlated to the extracted sentiments.

However, these early rule-based methods had some limitations and were unstable, so it was difficult to apply them in practice. Heeyoung Lee et al. [3] proposed a machine learning approach that forecasted companies' stock price changes in response to financial events reported in 8-K documents. Experiments with this approach indicated that using textual data can boost prediction accuracy over 10% (relative) over a strong baseline that only relies on financially-rooted quantitative indicators. The impact was most important in the short term (i.e., the next day after the financial event) but persisted for up to five days. The authors also demonstrated that additionally using sentiments did not significantly improve performance over the simpler unigram model.

Recent advances of deep learning models has inspired a new wave of stock market prediction research by analyzing stock-related textual data. For example, Chang et al. [4] developed a neural network to directly learn representations of news abstracts and showed that it can be an effective method for predicting the cumulative abnormal returns of stocks. Huicheng Liu [5] used a Bidirectional-LSTM to encode news and capture their context information, applying a self-attention mechanism to distribute attention on most relative words, news and days. This technique proved to be competitive with other state-of-the-art approaches in computational finance.

2 Problem statement

2.1 Requirements

An equity trading model must periodically predict stock prices based on observed input features. These input features can be both in numerical and in textual form. The output variable can be in multiple forms, such as direction or magnitude of stock price change, but it has to be closely related to the expected return from holding a particular stock. There are no strict requirements on the prediction accuracy of the model to be practically useful, but in general it should be better than random guessing and it should be at least good enough to cover associated trading costs, e.g. brokerage fees. All else being equal, simpler and faster models are preferred, since they can be applied to a wider range of stocks and they also allow to place trading orders quicker.

2.2 Constructed dataset

To be used as a text-based feature, a dataset of company significant development headlines was collected from Thomson One[2] for 44 blue-chip stocks in the period from 01/01/2010 to 03/01/2019. Significant developments is a news analysis and filtering service that identifies important company news on a real-time basis. It screens through multiple press releases, including 8-K reports, and briefly summarizes major company events. The items in significant Developments are selected and rated by trained analysts who continuously monitor various news sources (PR Newswire, Business Wire, The Wall Street Journal, etc.) on a real-time basis. After preprocessing, the number of headlines in the dataset totaled to 10,086.

Table 1 includes several examples that illustrate the collected headlines.

In addition, a dataset of numerical features has been collected which included returns of S&P500 index, money flow indicator (the flow of funds into and out of a security over last 14 days), forward P/E ratio (stock open price divided by 12-month earnings per share guidance), and short interest (percentage of short positions on a given stock). Taking stock index return as one of the features implies that stock beta will be automatically learned by the model, so there is no need to explicitly adjust stock returns for systematic risk. The purpose of the numerical dataset is to test the model in a more real-life environment, as well as to construct a baseline model, which is discussed in more detail in section 2.4.

[2] `http://www.thomsonone.com/`

Table 1: Example of significant development headlines.

	Company	Headline
1	Shell	Royal Dutch Shell PLC shuts San Pablo Bay pipeline, confirms small crude spill in California - Reuters
2	Apple	S&P assigns 'AA+' rating to Apple Inc's Australian dollar-denominated senior unsecured notes
3	Volkswagen	Porsche Automobil Holding SE And Volkswagen AG Merger On Track After U.S. Suit Dropped - Reuters
4	Novartis	Novartis AG Announces Positive Results From Final Phase III Omalizumab Registration Study In Severe Form Of Chronic Skin Disease CSU

2.3 Preprocessing pipeline

Preprocessing of significant development headlines consists of five steps. The first step includes filtering out those headlines that include non-ASCII symbols, e.g. written in a foreign language. On the second step, headlines are aggregated by release date so that multiple headlines released on the same date are concatenated into one sentence via a special "new line" token. On the next two steps, headlines are tokenized and PoS tagged using respective NLTK tools. Finally, each token is lemmatized by WordNetLemmatizer[3] based on the information received on the previous steps.

Figure 1 summarizes the discussed preprocessing pipeline.

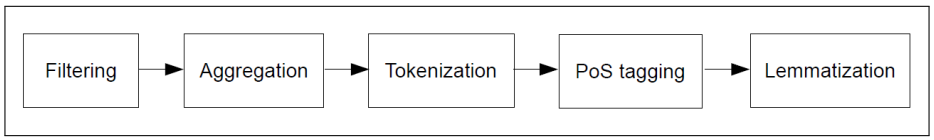

Figure 1: Preprocessing pipeline for significant development headlines.

2.4 Baselines

There are no straightforward baselines for the analyzed problem, so I use the following three benchmarks to evaluate the suggested architecture.

First, it is validated on CoNLL-2003 dataset[4] for the named entity recognition (NER) task. It is the task of tagging entities in text with their corresponding type. There are numerous similarities between NER and recognition of trading signals in press releases, so this dataset could be relevant in validating the suggested architecture and understanding whether it can learn efficiently. The benefit of using a well-known dataset such as CoNLL-2003 is that the result of the method can be compared to that of the current state-of-the-art solutions.

Second, I construct a custom baseline which essentially is the suggested architecture, but from which all text-based features are excluded, i.e. it can make predictions based on numerical features only. Instead of the textual inputs, I use randomly generated noise of the same dimension, so the model's structure remains unchanged. This approach allows to estimate the incremental effect of adding textual features into the forecasting model.

Finally, I compare the model with a third baseline based on a unigram bag-of-words model. I apply tf-idf vectorizer to the input text to create bag-of-words vectors. These vectors are compressed to the size of 500 through truncated SVD with 100 iterations. Truncated vectors are then concatenated with the rest of numerical features, and resulting set of features is used to run a support vector regression with the RBF kernel.

[3] `http://www.nltk.org/_modules/nltk/stem/wordnet.html`

[4] `http://www.clips.uantwerpen.be/conll2003`

2.5 Evaluation metrics

For the NER task, a common evaluation metric is f1 macro. I use this metric to compare performance of the suggested architecture to the current SOTA result. It is calculated as follows:

$$f_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}},$$

where precision is the percentage of named entities found by the algorithm that are correct and recall is the percentage of named entities defined in the corpus that were found by the program.

For the stock return prediction task, any common regression evaluation metric can be applied. Thus, I use RMSE, which has the following expression:

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\left(r_i^{true} - r_i^{pred}\right)^2}.$$

3 Suggested approach

3.1 Architecture overview

The suggested neural architecture is illustrated in Figure 2. It starts from two embedding layers, which are applied to each token of a tokenized textual feature. The first embedding layer produces word-level embeddings based on a pre-trained Word2Vec model. The second embedding layer is a Char CNN model that learns word embeddings from the sequence of each token's chars during model training.

Combined word embeddings then go through a BiLSTM layer. Its hidden states are used to compute attention weights, which determine those tokens which are relatively more important in the forecasting task. Embeddings produced by BiLSTM layer are then weighted by attention weights and summed up into a context vector.

Finally, the obtained context vector is combined with other numerical features that may have predictive power in the forecasting task. Combined feature vector serves as an input for a regression or classification model, depending on the nature of the predicted variable.

3.2 Word-level embeddings with fastText

I use pre-trained word-level embeddings from fastText English binary model[5]. This model is pre-trained in dimension 300 on Common Crawl and Wikipedia using CBOW with position-weights, character n-grams of length 5, a window of size 5 and 10 negatives [6].

FastText uses n-grams in its binary models to perform sub-word modeling, which helps to obtain vectors for most of out-of-vocabulary words. This approach assumes that a vector representation is associated to each n-gram, and words are represented as the sum of these representations. These vectors, while fast and quite easy to use, achieve state-of-the-art performance on a number of NLP tasks [7].

3.3 Char-level embeddings with char-CNN

The implemented char-level embeddings model is similar to that suggested by Yoon Kim, Yacine Jernite, David Sontag and Alexander M. Rush [8]. It consists of a trainable embedding layer with output size of 32, a 1-dimensional convolutional layer with window size of 5 and 64 filters, parametric ReLU activation, 1-dimensional global max pooling, a highway layer and 30% dropout. The maximum allowed word length is set to 16. The model is summarized in Figure 3.

3.4 BiLSTM encoder

Bidirectional LSTM layer consist of one forward and one backward LSTM layers with 16 hidden units. Each LSTM unit is implemented according to Sepp Hochreiter and Jurgen Schmidhuber [9].

[5] `http://fasttext.cc/docs/en/crawl-vectors.html`

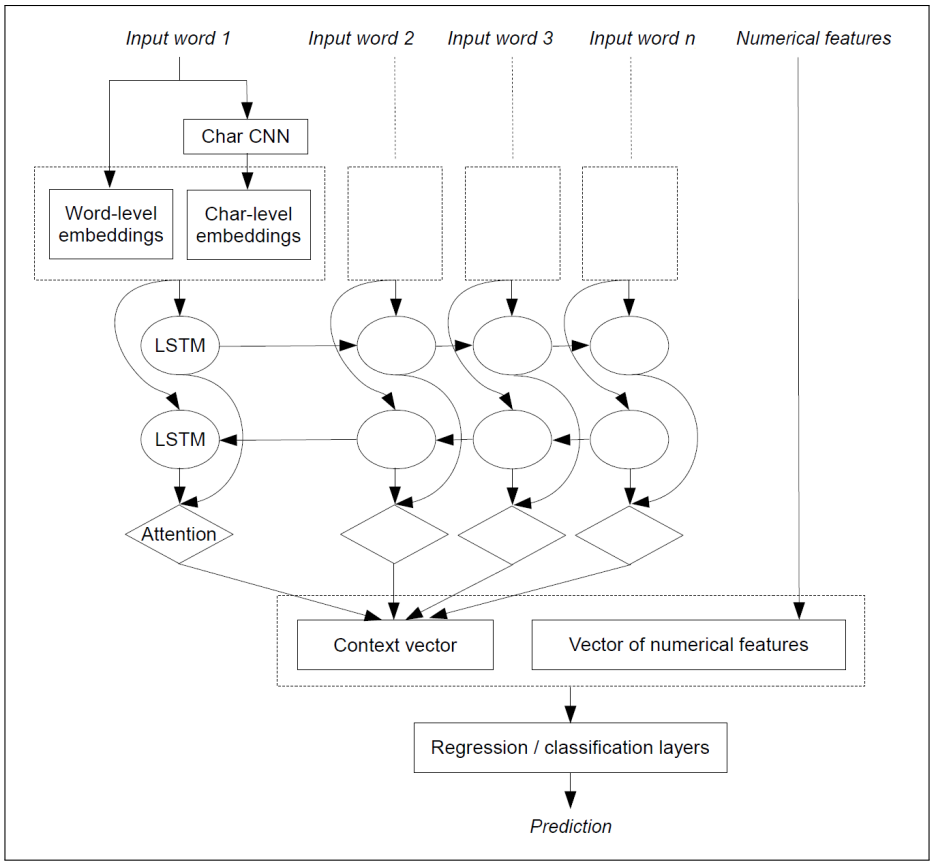

Figure 2: Architecture of text-based neural forecasting model.

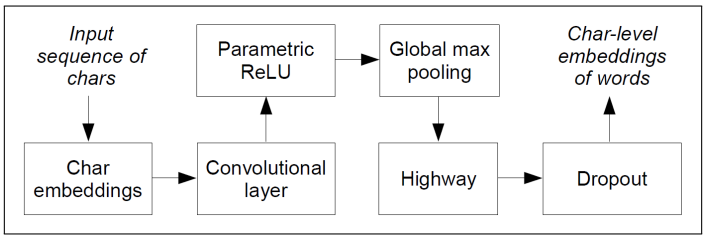

Figure 3: Char-level embeddings model.

3.5 Attention weighting

Attention mechanism has proven to be an efficient context aggregation solution in the text classification / regression tasks. For example, Peng Zhou et al. [10] used attention-weighted BiLSTM architecture on the SemEval-2010 relation classification task and showed that it outperforms most of the existing methods, while using only word vectors.

I implement attention mechanism as follows. First, forward and backward hidden states of the BiLSTM network are concatenated into one vector:

$$h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}].$$

Then, 16-dimensional attention score vectors are computed for each token based on BiLSTM network's output Y_t and its hidden state h_t:

$$S_t = \tanh((W_1 Y_t + b_1) + (W_2 h_t + b_2)),$$

where W_1, b_1, W_2, b_2 are trainable parameters. The resulting attention scores can be converted into a vector of attention weights by applying softmax transformation:

$$a_t = \text{softmax}(W_3 S_t + b_3),$$

where W_3, b_3 are also trainable parameters. The obtained attention weights are then used to compute context vector as the weighted-average output of the BiLSTM network:

$$\text{Context} = \Sigma_{t=1}^{T} a_t Y_t.$$

Lastly, I halve dimension of the context vector by passing it through a corresponding dense layer.

3.6 Output regression / classification

Once the obtained context vector is concatenated with the vector of numerical features, the remaining problem is just a standard regression / classification task (depending on the nature of the predicted variable, e.g. stock return vs. stock price movement direction). To solve this task, I use one hidden dense layer of size 16 placed after a 30% dropout layer, a parametric ReLU activation function and one output dense layer of size 1.

4 Experiments

4.1 NER task on CoNLL-2003 dataset

After training on 10 epochs, the model reaches f1 score of 0.921 on the test dataset. Each epoch takes ca. 185 seconds for training, i.e. training on 10 epochs takes around 30 minutes. The learning curve is illustrated in figure 4.

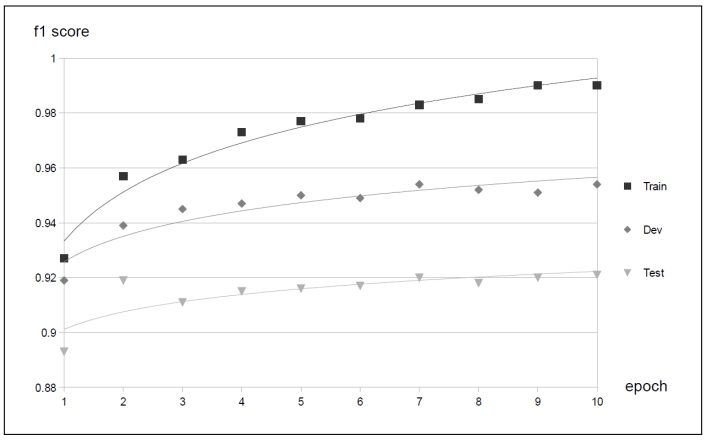

Figure 4: Learning curve on the NER task.

The obtained score is just lower by 0.01 than the current SOTA solution, which is based on flair embeddings. The latter was recently suggested by Alan Akbik et al. [11] and achieves f1 score of 0.931.

Thereby, the results of this test show that the suggested architecture, although is relatively simple to implement, can learn quite efficiently from the textual data.

4.2 Stock return prediction on constructed dataset

The dataset was split between train and test in proportion of 95% / 5%. I did not create a dev dataset because the number of observations is small.

Performance of the neural forecasting model with textual inputs started to flatten after training on just 5 epochs. The resulting RMSE on the test dataset is 2.11%. Each training epoch takes ca. 70 seconds, so training on 5 epochs takes around 6 minutes in total. The learning curve of the model is shown in figure 5.

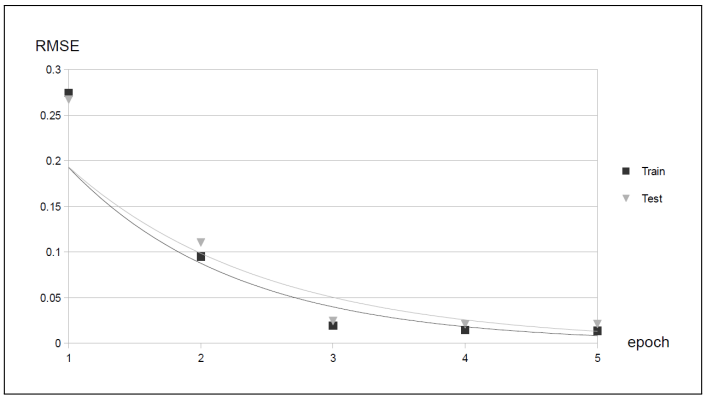

Figure 5: Learning curve on the stock return prediction task.

The discussed baselines for the stock return prediction task were also trained on the same train / test split of the dataset. A comparison of the models' performance is summarized in table 2.

Table 2: Evaluation of stock return forecasting models.

	Model	RMSE on train	RMSE on test
1	Neural forecasting model with textual inputs	1.37%	2.11%
2	Neural forecasting model without textual inputs	0.25%	2.36%
3	Support vector regression with tf-idf vectors	2.72%	2.98%

5 Discussion of results

First of all, experiments with the neural forecasting model showed that it performs better than a bag-of-word approach combined with a traditional machine learning regression. This result may suggest that it is important to consider combinations and interactions of factors to make a good stock return prediction.

Secondly, although neural forecasting model with textual features performed better on the test dataset than the same model without textual features, the incremental effect in the conducted experiment was relatively low - just 0.25% decrease in RMSE. Taking into account the good performance achieved on the NER task, this result may evidence low quality of the textual data that was used in the experiment. Thus, in order to make use of the suggested architecture, it might be important to explore a wide range of sources that provide textual descriptions of analyzed stocks.

Lastly, it is interesting to note that the neural forecasting model started to overfit after I removed textual features from its input space and substituted them by random noise. This is illustrated by the fact that RMSE on the train dataset improved from 1.37% to 0.25%, although performance on the test dataset got worse. Thus, the benefit of adding textual inputs into a stock return forecasting model might include not only obtaining additional signals of stock price change, but also decreasing the number of false signals produced by the model.

6 Conclusion and future work

This paper presented a simple neural architecture, based on attention-weighted BiLSTM encoder, which allows to easily incorporate textual features into a stock return forecasting model (or any other forecasting model). The model demonstrated a relatively good performance on CoNLL-2003 dataset for the NER task, as well as it was able to produce an incremental increase in predictive power on a real trading dataset by additionally incorporating stock-related textual information. However, the incremental effect appears to largely depend on the quality of provided textual inputs, so it might be important to explore a wide range of sources that provide textual descriptions of stocks in order to make use of the suggested architecture.

A future work on this topic may relate to a number of areas, such as (i) construction and analysis of a wider dataset of financial news and reports, (ii) utilization of intraday stock return data that can better capture the immediate effects of news and reports publication, (iii) differentiation among sources of textual data, since they may affect stock price asymmetrically, (iv) incorporation of deep contextualized word representations that can model polysemy, such as ELMO [12], (v) development of a transformer forecasting architecture [13] which can potentially improve the applied RNN-based sequence-to-sequence approach and others.

Acknowledgments

I would like to acknowledge contribution of my CS224n mentor Pratyaksh Sharma.

References

[1] Joseph Engelberg (2008) Costly Information Processing: Evidence from Earnings Announcements. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1107998

[2] Vivek Sehgal, Charles Song (2007) Stock Prediction Using Web Sentiment. https://ieeexplore.ieee.org/document/4476641

[3] Heeyoung Lee, Mihai Surdeanu, Bill MacCartney, Dan Jurafsky (2014) On the Importance of Text Analysis for Stock Price Prediction. https://nlp.stanford.edu/pubs/lrec2014-stock.pdf

[4] C.-Y. Chang, Y. Zhang, Z. Teng, Z. Bozanic, B. Ke (2016) Measuring the information content of financial news. http://aclweb.org/anthology/C16-1303

[5] Huicheng Liu (2018) Leveraging Financial News for Stock Trend Prediction with Attention-Based Recurrent Neural Network. https://arxiv.org/pdf/1811.06173.pdf

[6] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, Tomas Mikolov (2018) Learning Word Vectors for 157 Languages. http://arxiv.org/pdf/1802.06893.pdf

[7] Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov (2017) Enriching Word Vectors with Subword Information. http://arxiv.org/pdf/1607.04606.pdf

[8] Yoon Kim, Yacine Jernite, David Sontag, Alexander M. Rush (2015) Character-Aware Neural Language Models. http://arxiv.org/pdf/1508.06615.pdf

[9] Sepp Hochreiter, Jurgen Schmidhuber (1997) Long short-term memory. http://www.bioinf.jku.at/publications/older/2604.pdf

[10] Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, Bo Xu (2016) Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification. https://www.aclweb.org/anthology/P/P16/P16-2034.pdf

[11] Alan Akbik, Duncan Blythe, Roland Vollgraf (2018) Contextual String Embeddings for Sequence Labeling. https://alanakbik.github.io/papers/coling2018.pdf

[12] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer (2018) Deep contextualized word representations. https://arxiv.org/pdf/1802.05365.pdf

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin (2017) Attention Is All You Need. https://arxiv.org/pdf/1706.03762.pdf