Utilizing a Reweighting Hyperparameter to Improve Deep Learning Twitter Based Binary Stock Price Direction Classifiers

Nakul Upadhya(Junior) [1]

Abstract

In this paper, we attempt to test whether or not the Reweighting method described in ”Learning to Reweight Examples for Robust Deep Learning” (Ren et al., 2018) can be used to improve the accuracy of a Non-Specific Binary Stock Movement Classifier that utilizes congregated Twitter data. Our work determined that an unbalanced meta set can be used to create a weight hyper-parameter for our prediction costs, which allows our classifier to better match tweet sentiment to stock price directional movement.

1. Introduction

Understanding the sentiment of the crowd could lead more effective swing-trading due to the crowd collectively having more money than before, and therefore more market sway. Events such as the Gamestop (GME) spike in January 2021 have shown that stock prices and sentiment are highly linked (Umar et al., 2021). The power of market sentiment has only increased with the introduction of the ”retail investor,” with Robinhood users collectively increased their stock holdings during the March 2020 (Welch, 2020). It is reasonable to say that the buying and selling patterns of this formerly small group now has a much bigger impact compared to before. This is supported by the EMH that says the current stock market fully reflects all available information (Nguyen et al., 2015), including public sentiment.

Despite this, one section of thought is that the causation between price direction is very weak and following the sentiment of the crowd not an accurate source of wisdom (Checkley et al., 2017). Additionally, sentiment analysis models work better on a very short term, around 20-30 minutes, and models attempting to predict the day-to-day changes rarely have a substantial accuracy (Nguyen et al., 2015). This is a problem for many retail and beginner traders since most brokerage platforms limit the amount of day trades one can make in a certain period, forcing them towards swing trades that rely on anticipating price movements on a day-to-day basis.

One potential reason for this is that social media data can be noisy, and often automatically pulling data from a platform can result in pulling unrelated data into your analysis (Nguyen et al., 2015). Tweets mentioning a certain stock ticker may not have that stock as the main focus of the tweet. As a result, the sentiment of the tweet may not match the price movement of the stock in many cases.

This paper attempts to connect stock sentiment to stock movement by treating data points where the sentiment does not match the next-day stock movement as noise, and then utilizing a re-weighting hyper-parameter (Ren et al., 2018) so that noisy or unrelated data points can have less of an impact on training a deep-learning sentiment model, resulting in a greater prediction accuracy. A successful implementation of this algorithm will mean that others attempting to predict price changes from social media will have less of a reliance on utilizing MTurk and other crowd-work sites for their data processing, and can rather rely on more automated means like automated API calls. Additionally, this same method can be extended to other NLP based stock prediction methods utilizing other kinds of data sets.

2. Related Work

2.1. Use of Social Media for stock market prediction

Attempting to predict the stock market using social media is not a new problem, but almost everyone who attempts to tackle this issue utilizes a different method.

(Si et al., 2013) created a daily, one-day-ahead prediction model using a continuous Dirichlet Process Mixture. This model learns the daily topics going around on Twitter daily, and then uses those topics to create a sentiment time series model to predict the market.

(Nguyen et al., 2015) proposed using a ’topic-sentiment’ feature derived using a joint sentiment/topic model to improve stock market prediction. Their model attempting to capture the price movement using this feature at $t-1$ and $t-2$ where t is a transaction date.

Dickinson & Hu 2015 utilized an n-gram and word2vec textual representation technique alongside a random forest classification algorithm to predict the sentiment of tweets. These values were then evaluated to identify correlation between stock prices and Twitter sentiment for companies mentioned in those tweets.

Xu & Cohen 2018 worked on creating a model that uses temporally-dependent predictions from chaotic data by utilizing recurrent, continuous latent variables to better handle the stochastic nature of the market. This model also utilizes neural variational interference to address the intractable posterior inference and a hybrid objective to flexibly capture predictive dependencies.

2.2. Sentiment Analysis without Social Media sources

Social media has not been the only avenue for capturing market sentiment. In fact, sites like Twitter and Reddit are now only coming up, with news being the main source for information (Nguyen et al., 2015).

A 2018 study looked at Dow Jones newswires text feed for a period of four years spanning from 2013 till 2017 (Yıldırım et al., 2018). This study used a Bag-of-ngrams approach for text representation and Term Frequency-Inverse Document Frequency for text weighting. These were fed into Logistic Regression , Support Vector Machines , k Nearest Neighbours and Multinomial Naïve Bayes classifiers with Grid search hyper parameter. After comparing these, the paper found that SVM and Logistic Regression outperformed the other models on 5 different performance measures: success rate, precision, recall, F-1 measure, and area under ROC (Yıldırım et al., 2018).

Convolution Neural Networks similar to the one used in this paper have also been used in identifying directional movements in the S&P500 (Vargas et al., 2017). This study found that while Recurrent Neural Networks were better at catching context information and temporal characteristics, Convolution Neural Networks were better at capturing semantics from text. As we are attempting to understand text sentiment, we will be utilizing a CNN for this paper.

2.3. Technical Inspiration

The main inspiration for the method proposed in this paper was the Learning to Re-weight paper. The step-by-Step psuedocode described in the paper is the basis for our modified model.

In order to effectively create a model various online resources were utilized. One online resource was Ben Trevett's PyTorch Sentiment Analysis guide (Trevett, 2017). Specifically, our baseline model is based heavily off of the Convolution Sentiment Analysis tutorial created. The data used by ”Stock Movement Prediction from Tweets and Historical Prices” was utilized to train both our baseline and modified model (Xu & Cohen, 2018). The code from this paper however was not used due to its complexities.

3. Method

To see the effects of the reweighing method, we will first create a baseline Convolution Neural Network and evaluate how this baseline model performs with regards to various modifications to the data inserted and other hyperparameters. After collecting data on the baseline model, we will implement the reweighing method, and re-run the same tests.

3.1. Data and Model

We will be using a 4 layer Convolution Network as a baseline for our model as developed by GitHub user Ben Trevett (Trevett, 2017) in `PyTorch`. The first layer will have 2 filters, the second layer will have 3 filters, and the third layer will have 4 filters. Our last layer is a Linear FC Layer. We will be using SGD as our optimization algorithm and Binary Logistic Cross-Entropy Loss.

This network will utilize all the Tweets sent mentioning a specific stock on day n to predict the general price movement (either increase or decrease) of that stock on day $n + 1$.

We will be adapting GitHub user Ali Haydari's `PyTorch` implementation of the Learning to Reweight automatic differentiation pseudo-code (Haydari, 2020), which uses `higher` (Grefenstette et al., 2019) for the inner-loop meta learning involved in the method.

3.1.1. Input Data Specifics

We will be using Tweets and pricing data from the StockNet data set (Xu & Cohen, 2018). This data set contains pricing data and tweets sent about 87 different stocks organized by date. This data set was processed into two groups: a messy set and a clean set. For the clean data set, stop words and punctuation were removed using `NLTK`. Additionally, tokens used to replace Twitter specific information, like `AT_USER` and `URL`, were removed from the text. Lemmitization was also applied to the text to ensure consistency. None of this was done for the messy data set.

For both data sets, all the tweets from a single day were combined into one string. The text was then tokenized using `spaCy`. The resulting text was then labeled either a 0 if the closing price of the current day is more than the closing price of the stock for the next day (a next day price decrease), or a 1 if the stock price if the current day closing price is less than the next day closing price (a price increase).

Both the messy and clean sets were then split into a training, validation, and testing set, eaching having their own meta-set. The meta-sets are ”noiseless” subsets of the respective master sets. The meta-sets are created by analyzing the sentiment of each data point's text using `flair` (Akbik et al., 2019), and seeing if the sentiment returned by `flair` matches the the next-day price direction (a positive sentiment matches a price increase, a negative sentiment matches a price decrease). Any point where the direction matches the sentiment, and `flair` was able to predict the sentiment with over a 75% confidence was then added to the meta sets. The training meta-set is used for the re-weighting method, while the validation and testing meta-sets serve as benchmarks to understand how a network would work with noiseless behavior.

3.2. Experimental Methods

To evaluate the validity of the re-weighting method

First, we used only sentiment to predict a stock and passed our messy and clean testing data into `flair` . We predicted an increase if there was a positive sentiment and a decrease if there was a negative sentiment. This provided us a goal for our other models to meet or surpass.

We then trained and evaluated our baseline CNN on our training data set to create a baseline. We also trained and evaluated our baseline CNN on our meta sets to understand how a normally trained network would perform on ”noise-less” data. We will then repeat these experiment using the messy version of the training and meta sets to understand how data cleaning affects this problem.

We will then repeat all these tests while using the re-weighting method on our CNN. Our goal is to achieve similar, or better, results compared using only sentiment analysis or a normally trained CNN.

Additionally, we will also vary the size of the data set used for training and testing our baseline CNN and reweighted CNN. This will help us understand whether or not the reweighting training method's performance varies heavily with the amount of data used to train it, as the method relies on weighting the loss of many points to zero, effectively not training those points.

For each model and data set combination, we run the model for a set number of epochs, and evaluate and record the testing and validation performance for each epoch along with the meta set performance if applicable. To ensure we get the best results across all the epochs, we save the state of the network in the epoch that has the lowest validation loss and then load this state before we evaluate and collect the accuracy, loss, precision, and recall on our testing set. We repeat this training procedure 4 times and average the metrics per run for our end results.

4. Experiment

4.1. Flair Experiments

To explore the viability of traditional sentiment analysis on our data set, both the messy and clean testing set data was passed into `flair` and the label was predicted with only the `flair` sentiment as a factor for prediction.

	Accuracy	**Loss**	**Precision**	**Recall**
Messy Data	0.488	—	0.511	0.190
Clean Data	0.492	—	0.503	0.448

Table 1. Flair Results

As seen in table 1, the performance with only sentiment is dismal. The accuracy on both the messy and clean data were both under 50% and were quite similar in value. One detail to note however is the especially low recall of 0.190 on the messy data set, in comparison of a recall score of 0.448 on the clean data set. The low score indicates that poor data cleaning makes it harder for `flair` to label a point with a positive sentiment. This could have a potential impact when we run our messy testing data through the model using the re-weighting method as the distribution of labels in meta set constructed by flair could be heavily skewed towards negative labels, which could potentially have a detrimental impact on the accuracy and loss.

Loss for this method was not collected as the `flair` package only returns a label, and not a score. No experiments were run on the meta sets, as `flair` was used to create the meta sets, therefore the experimental results would be redundant.

4.2. CNN Baseline Experiments

4.2.1. Full Data Set

	Accuracy	**Loss**	**Precision**	**Recall**
Messy Data	0.516	0.692	0.500	0.352
Clean Data	0.510	0.694	0.498	0.375

Table 2. Baseline Full Testing Set Results

After running the CNN on our data, we can see that the accuracy is better than using only a sentiment model, but barely so. Additionally, while the messy data performed better in terms of accuracy, the difference is minor and not very significant. However, the messy set is much more resource intensive, and training took around 28 minutes on average, while the clean set only took around 10 minutes on average.

	Accuracy	Loss	Precision	Recall
Messy Data	0.833	0.436	NaN*	0.0*
Clean Data	0.648	0.634	0.616	0.587

Table 3. Baseline Full Testing Meta Set Results

Our CNN had interesting behavior when training and testing the network using the meta sets. As we saw in table 1, `flair` found it difficult to predict a positive sentiment when the data was messy. As we used this tool to create our meta set, the distribution in the meta set is highly skewed towards negatives. As a result, our CNN learned to always predict negative for the best score. As a result, the number of true positives and false positives was zero, resulting a recall score of zero and an invalid precision score. This could potentially impact the performance of the reweighted CNN when messy data is passed in, as it uses the meta set to created the weighting hyper-parameter.

When the clean meta sets were used to train test the network, the model performed significantly better compared to the normal testing and training sets. Not only is the accuracy higher, but both the precision and recall scores are well above 0.5. This makes sense as the meta sets are theoretically "noiseless" and the sentiment of the tweet matches the label for all the points. However, this set is unusable for practical use as actual data will always be noisy. Our hope is that we can use the reweighting training method to achieve similar accuracy results while using our normal, noisy data set.

4.2.2. Partial Data Set

	Accuracy	Loss	Precision	Recall
Messy Data	0.510	0.693	0.492	0.474
Clean Data	0.498	0.695	0.488	0.468

Table 4. Half Data Set Baseline Results

Our networks performed worse with smaller testing sets, but not significantly worse. One interesting behavior however is that the recall scores in the smaller sets are significantly higher. This could suggest that the number of false negatives has dropped when passing in smaller sets.

	Accuracy	Loss	Precision	Recall
Messy Data	0.836	0.441	0.NaN*	0
Clean Data	0.603	0.661	0.687	0.301

Table 5. Half Meta Data Set Baseline Results

The behavior of the smaller messy meta training and testing sets is very similar to the full data set. The network still learns to only predict 0's / negative next-day price direction. Additionally, the accuracy and loss value difference between the full data set and the half data set is negligible.

4.3. CNN With Reweighting

4.3.1. Full Data Set

	Accuracy	Loss	Precision	Recall
Messy Data	0.874	0.484	0.484	1.0
Clean Data	0.697	0.489	0.488	0.907

Table 6. Full Data Set Reweighting Training Method Results

When the re-weighting training method is used to train this network, promising results are seen. On the clean data set, we get an accuracy of 0.697, which is significantly higher than the performance of `flair` (Table 1) and our normally trained CNN (Table 2) as both of these methods were only around 0.5 accuracy. The accuracy of the re-weighted CNN is even higher than the accuracy of the baseline CNN on the meta set (0.697 vs 0.648). Additionally recall score of the clean set completely eclipsed the recall scores of the the baseline trained on the normal and meta training sets. However the precision score is very similar to the baseline performance on the meta set, and is actually worse than the precision score of the baseline trained on the meta set. This indicates that our model has a very small number of false negatives.

The performance of using the messy sets is even more shocking. The accuracy is 0.874 which is significantly higher than any other recorded accuracy for any parameters. Additionally, the recall score is a 1.0, indicating that the model has no false negatives. This behavior could be attributed to the nature of the meta training set that we saw when training the baseline on the messy meta sets. The messy meta training set is heavily weighted towards 0's, and as a result the traditionally trained baseline model learned to only predict negative price decreases. This meant that there were no true positives. In comparison, the reweighting training method seemed to utilize this reweighted data set to be more selective of when it predicted a decrease, resulting in a much higher accuracy. This is supported by the precision score of 0.484, indicating that the network had a decently high number of false positives. This model's behavior is that it does not perfectly predict all price decreases, but its price decrease prediction is perfect. To confirm this, we attempted to create an imbalance in the meta set of our clean data by removing data points from the meta set. We mimicked the approximately 4:1 negative:positive split and ran the reweighting method using the full clean data set, and the modified meta set.

	Accuracy	Loss	Precision	Recall
Mod. Set	0.859	0.487	0.489	1.0

Table 7. Full clean data set results with a modified meta set

The results when utilizing a modified meta set on the clean data are very similar to the results. The network again did a very good job at preventing false negatives and had an above 85% accuracy. This result affirms that unbalancing the data in the meta set causes higher accuracy, not having uncleaned data.

4.3.2. Partial Data Set

	Accuracy	Loss	Precision	Recall
Messy Data	0.873	0.483	0.482	1.0
Clean Data	0.700	0.490	0.490	0.958

Table 8. Half Data Set Reweighting Training Method Results

Changing the number of data points used to train the network had no signficant impact on the model performance.

4.4. Training Behavior

One difference between training our CNN traditionally and using the reweighting training method is the behavior the CNN exhibits during the training process.

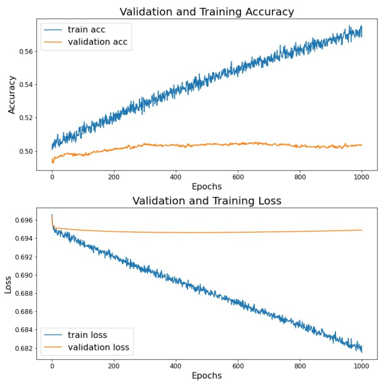

Figure 1. Baseline CNN training behavior on non-meta clean data

While using a traditional training, the training accuracy continuously increases, while the training loss continuously decreases (Figure 1). The validation accuracy during training initially increases, but then remains relatively constant. However, the training behavior of the CNN trained using the reweighting method (Figure 2) is drastically different. Like the traditional method, the validation accuracy and loss remains steady. However, the training and meta accuracy are also both very constant after around the first 100 epochs, which is different from the traditional method. Additionally, the training accuracy actually decreased during the first few iterations. The behavior of the meta loss and accuracy seems to suggest that the reweighting method causes the network to converge very fast in comparison to the traditional network. One more interesting point is that the validation accuracy started at a much higher point with the reweighting method, which suggests that the impact of the reweighting method causes higher accuracy from the very first epoch.

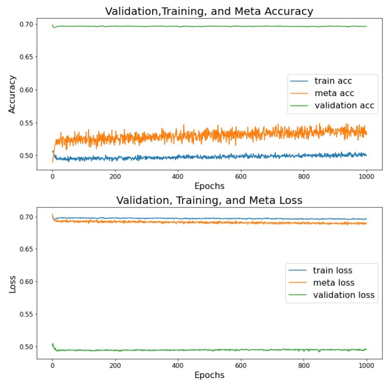

Figure 2. Reweighting training method behavior on clean data

The training behavior of our baseline CNN on messy data (Figure 1) is similar to its behavior on clean data (Figure 3). Both have a relatively constant validation accuracy and loss and the first few epochs, while the training performance continuously gets better throughout the epochs.

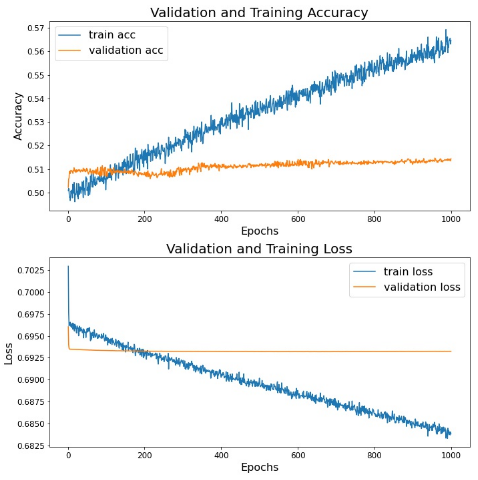

Figure 3. Baseline CNN training behavior on non-meta messy data

On big difference between this and the CNN trained on clean data is the magnitude of the initial performance jump on the validation set. On the clean set, the validation accuracy barely changed during the first few epochs, staying around

0.49 (Figure 1). In comparison, the validation loss using the messy set resulted in a jump of 0.01 before it stayed constant.

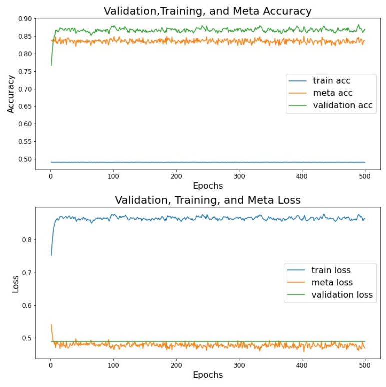

Figure 4. Reweighting training method behavior on messy data

Like the clean data, the validation, meta, and training accuracy and loss remains constant after the first few epochs when using the reweighting method. However, unlike the clean data, the training loss on the messy data jumps rapidly during the first few epochs. Additionally, the validation accuracy remains relatively constant during the first few epochs when clean data is used. However, the validation accuracy rapidly jumps up when messy data is used to train the network. One more difference between utilizing clean data and messy data is the meta set accuracy and loss. When clean data is used, the meta set accuracy and loss has a closer resemblance to the training loss and accuracy. The opposite behavior happens when we use messy data and the meta loss and accuracy more resembles the validation loss and accuracy.

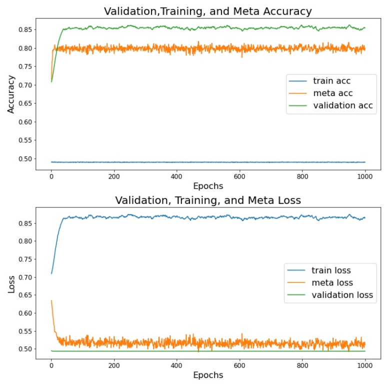

Figure 5. Full Clean Data set Training Behavior with Unbalanced meta set.

More accurately, this behavior occurs when we utilize an unbalanced meta set to perform the reweighting method. After unbalancing the clean meta set, the validation, meta, and training curves (Figure 5) of the clean data change. Now, all the curves are almost identical to the curves present when using the reweighting method on with the messy data (Figure 4).

5. Conclusion

5.0.1. Limitations and Uses

The reweighting training method has consistently shown great results in predicting the next day stock direction especially when compared to only relying on sentiment or a traditionally trained sentiment model.

Despite the method's success, there are a few limitations to this research that need considering. One limitation of our experiments is that we do not consider time in this and treat every data point as distinct when in reality the sentiment from multiple days could have an impact on the price direction for a day. Additionally, this model does not directly account for events surrounding the stocks like impactful world news, earning releases, or product releases. However, these events are still semi-captured through the sentiment of twitter users. Additionally, the experiments performed on the network were limited by access to hardware and computing power. This caused quite a few problems, one being that the messy data sets were only able to be trained for 500 epochs before memory issues were encountered, instead of an ideal number of 1000 epochs.

One practical limitation to the usefulness of our findings is related to the type of model developed. While the CNN trained using the reweighting method has a high accuracy, it is still only a binary classifier. Knowing that a stock price will rise or fall gives no indication on the magnitude of the price change. This makes it hard to utilize this information for option-based swing trading strategies, since option pricing is determined by the Black-Scholes equation, and the spot price of an asset is only one variable in that equation.

Despite this, the reweighted model is incredibly useful for risk management and short term stock-trading. When run with the messy data, the reweighting method has an false negative rate of 0, making it incredibly useful as an indicator to either sell stock to retain gains, or short sell stocks to make some profit.

5.0.2. Further Study

As mentioned before, our end result was a very good classifier, but it is still only a binary classifier. Further study would involve extending our classification model from only using positive and negative labels to a series of labels describing the magnitude of the price change. This endeavor would also require us to identify reliable sentiment models similar to `flair` that also have extra labels - like Very Positive or Very Negative - in order to effectively create a meta set for the expanded model.

Our current reweighted CNN could also be used in conjunction with an LSTM time series stock-price model to confirm the outputs of the LSTM price predictions. Additionally, we could also utilize the predictions from our reweighted CNN as a basis for deciding what transition rates and probabilities we can use in a continuous Markov Chain, allowing us to potentially create a intra-day day trading model.

To actually use the reweighted CNN to assist swing trading, the network will need to be mounted onto either a server or a cloud platform along with a program to feed Twitter data into the CNN using the Twitter API. Additionally, work will be needed to decide what time of day tweets should be collected at, as many drastic price changes could happen right as the day opens or closes, and information given about the market during the market close could potentially be outdated once the market opens up again. Cost would also be a large factor in actually using the reweighted CNN. More work is needed to optimize the computing performance of the reweighting method, as mounting the current version of the CNN on a cloud provider such as AWS or GCP would result in tremendous costs whenever the model needs to be trained, as the reweighting training method has high requirements for RAM and computing power.

References

Akbik, A., Bergmann, T., Blythe, D., Rasul, K., Schweter, S., and Vollgraf, R. Flair: An easy-to-use framework for state-of-the-art nlp. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pp. 54–59, 2019.

Checkley, M., Higón, D. A., and Alles, H. The hasty wisdom of the mob: How market sentiment predicts stock market behavior. *Expert Systems with Applications*, 77:256–263, 2017. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2017.01.029. URL `https://www.sciencedirect.com/science/article/pii/S0957417417300398`.

Dickinson, B. and Hu, W. Sentiment analysis of investor opinions on twitter. Technical Report 24, 2015. URL `https://www.scirp.org/html/2-2680089_57841.htm?pagespeed=noscript`.

Grefenstette, E., Amos, B., Yarats, D., Htut, P. M., Molchanov, A., Meier, F., Kiela, D., Cho, K., and Chintala, S. Generalized inner loop meta-learning. *arXiv preprint arXiv:1910.01727*, 2019.

Haydari, A. Learning-to-reweight examples for robust deep learning with pytorch-higher. `https://github.com/TinfoilHat0/Learning-to-Reweight-Examples-for-Robust-Deep-Learning-with-PyTorch-Higher`, 2020.

Nguyen, T. H., Shirai, K., and Velcin, J. Sentiment analysis on social media for stock movement prediction. *Expert Systems with Applications*, 42(24):9603–9611, 2015. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2015.07.052. URL `https://www.sciencedirect.com/science/article/pii/S0957417415005126`.

Ren, M., Zeng, W., Yang, B., and Urtasun, R. Learning to reweight examples for robust deep learning. *CoRR*, abs/1803.09050, 2018. URL `http://arxiv.org/abs/1803.09050`.

Si, J., Mukherjee, A., Liu, B., Li, Q., Li, H., and Deng, X. Exploiting topic based twitter sentiment for stock prediction. volume 2, pp. 24–29, 2013. URL `https://www.scopus.com/inward/record.uri?eid=2-s2.0-84906932179&partnerID=40&md5=27afd8acdf3426494abcdf2c01a82a46`. cited By 135.

Trevett, B. Pytorch sentiment analysis. `https://github.com/bentrevett/pytorch-sentiment-analysis`, 2017.

Umar, Z., Gubareva, M., Yousaf, I., and Ali, S. A tale of company fundamentals vs sentiment driven pricing: The case of gamestop. *Journal of Behavioral and Experimental Finance*, 30:100501, 2021. ISSN 2214-6350. doi: https://doi.org/10.1016/j.jbef.2021.100501. URL `https://www.sciencedirect.com/science/article/pii/S2214635021000459`.

Vargas, M. R., de Lima, B. S. L. P., and Evsukoff, A. G. Deep learning for stock market prediction from financial news articles. In *2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, pp. 60–65, 2017. doi: 10.1109/CIVEMSA.2017.7995302.

Welch, I. The wisdom of the robinhood crowd. Working Paper 27866, National Bureau of Economic Research, September 2020. URL `http://www.nber.org/papers/w27866`.

Xu, Y. and Cohen, S. B. Stock movement prediction from tweets and historical prices. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1970–1979, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1183. URL `https://www.aclweb.org/anthology/P18-1183`.

Yıldırım, S., Jothimani, D., Kavaklıoğlu, C., and Başar, A. Classification of ”hot news” for financial forecast using nlp techniques. In *2018 IEEE International Conference on Big Data (Big Data)*, pp. 4719–4722, 2018. doi: 10.1109/BigData.2018.8621903.