Reproduce Stock Price Prediction Tasks with News Analysis

NYU FRE-7871 Final Project Report

Pei-Lun Liao
New York University
pll273@nyu.edu

ABSTRACT

In this project , selected related works will be reproduced with different financial news dataset. The goal is to understand techniques and challenges in stock price prediction.

1 INTRODUCTION

Nowadays, traders apply machine learning technique to extract information from financial news data. The information can be used to boost the accuracy of stock price prediction [4, 5, 15]. The project aims to reproduce selected related works [15]. Hence, people can learn the challenge and technique that are crucial in stock price prediction.

2 APPROACHES

2.1 Data

Data quality is crucial to make the accurate prediction [17]. Unfortunately, it is not easy to find a publicly available financial news dataset. For example, the public financial news dataset published in [4] was currently unavailable due to license issue [13]. Another corpus available online requires membership and subscription fee [14]. Also, it is not practical to crawl dataset from business news media in this one month project. Most of the time will be spent on data collecting, cleaning, and processing instead of understanding the technique that works for stock price prediction.

2.1.1 Webhose.io and IEX API. Fortunately, an available financial news dataset was found on Webhose.io [16]. The data was crawled from the Internet from July to October in 2015. 47,851 news articles were collected in machine-readable format. However, there is no paper shows the dataset could help stock price prediction. We filtered new articles by finding the S & P 500 company name in the article as described in section 2.2. After that, we have 5,375 articles left.

News articles from Webhose.io come globally as shown in Figure 1 and 2. Each day we have around 50 articles as shown in Figure 3. Generally, more than half of the news was published after the close time.

End of date S & P 500 stock prices from July to October in 2015 was downloaded from IEX API [1]. However, we can't find the S & P 500 stock price by minutes. Therefore, we could not have the current stock price to predict the next 20 minutes stock price as the experiment in AZFinText. What we can do is to predict the trend between the open time and close time and to see if we can capture some text pattern.

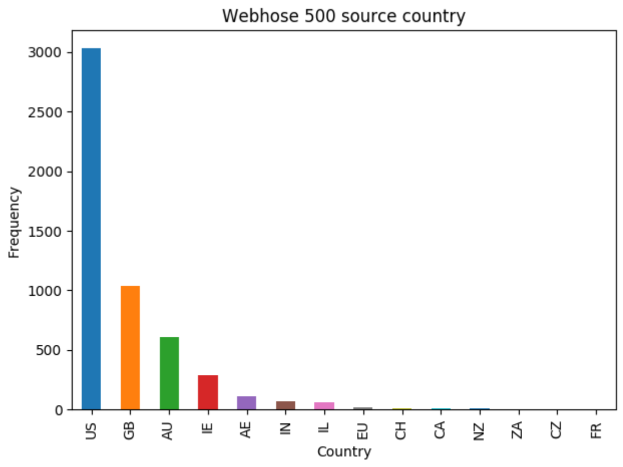

Figure 1: Webhose.io with S & P 500 news article country source

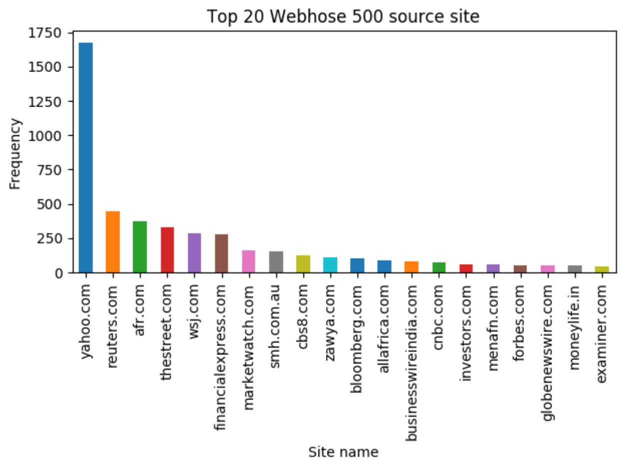

Figure 2: Webhose.io with S & P 500 news article site source

2.1.2 Reuters and IEX API. Due to the quality issue in Webhose.io dataset, we need to find another way to experiment. Hence, we crawled S & P 500 stock prices by minute from IEX API [1] and collected 9013 business news articles from Reuters [2] in the period between July 2nd 2018 to September 30th 2018. Due to the limited available open stock price data, I can only collect three months dataset. The data quality is not guaranteed as well. Finally, 1107 articles can be mapped to a specific stock name, and only 308

The source code was released on https://github.com/plliao/FRE-7871-project

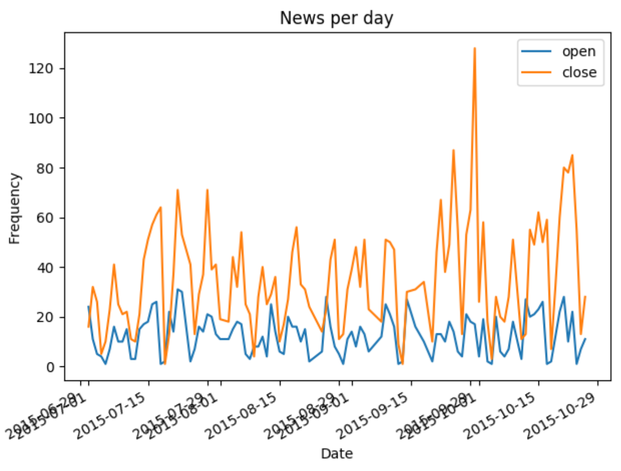

Figure 3: Webhose.io S & P 500 news per day

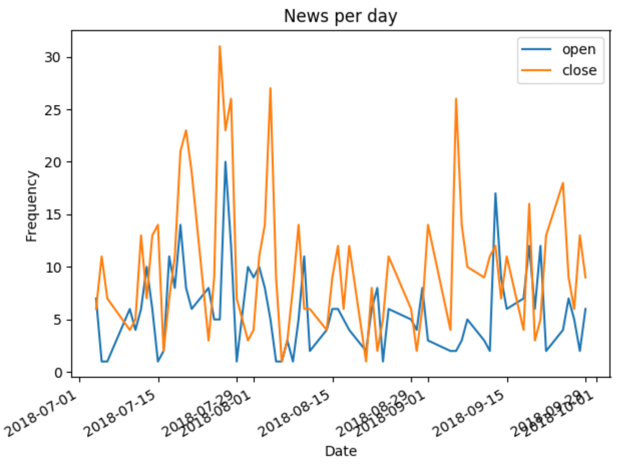

Figure 4: Reuters S & P 500 news per day

articles were published in open trading time. Each day, we have around 15 articles as shown in Figure 4.

2.2 Tickers labeling

We cleaned S & P 500 company names by removing punctuations and common terms like "Inc.", "Corp.", or ".com" and mapped an article to the stock if we can find the name in the article. Also, some formal company name like "Alphabet Class A" and "Alphabet Class C" were replaced to "Google" since reporters often use the word "Google" instead of "Alphabet" in their articles. We removed the terms like "Google+" and "Google Plus" to get rid of the sharing on social media text. We ignore the ambiguous company name for data quality as well. For example, "CA, Inc." was ignored because the word "CA" is ambiguous in an article.

The distribution of tickers on Webhose.io and Reuters dataset can be found in Figure 5 and 6. Google contributes the most news articles in both Webhose.io and Reuters dataset. In this process, we

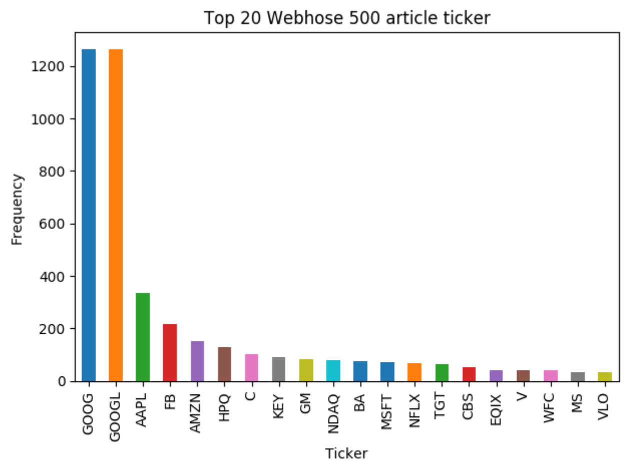

Figure 5: Top 20 S & P 500 tickers in Webhose.io dataset

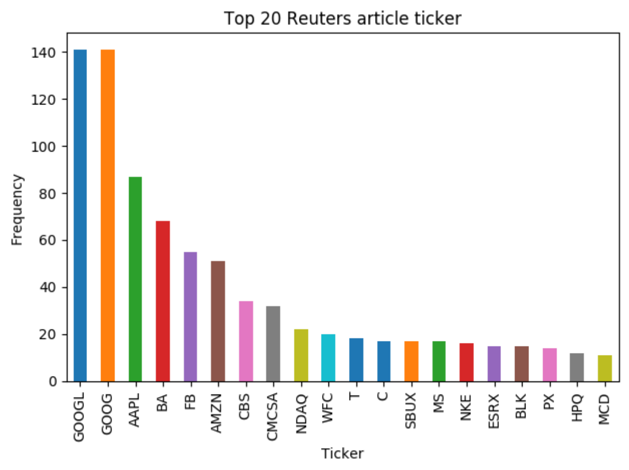

Figure 6: To 20 S & P 500 tickers in Reuters dataset

found several company names can be mentioned in the same article. Hence, we will have the same text features with different label price. Sometimes, people will mention rival companies in the same article [17] and the article will describe good and bad things at the same time. It will hurt our performance. The better way to do that is identifying the company mentioned in a sentence and collecting those sentences into an article. However, we didn't implement that.

2.3 Price Labeling

For each news article, we attached prices by the ticker name we found.

2.3.1 Webhose.io and IEX API. Since we didn't have the stock price by minutes, we attached the prices we have before and after the news published. For example, assume the stock market is open on 10/01. The news published at 10:00 on 10/01 will be attached the prices at 09:30 and 16:00 on 10/01. It is the open and close price on

10/01. We are interested in the news relationship with the market price. For the same reason, if the article published at 19:00 on 10/01, we will attach the close price at 16:00 on 10/01 and the open price at 09:30 on 10/02 to the article. After that, Webhose.io dataset has 4074 articles left.

2.3.2 Reuters and IEX API. For the articles on Reuters, since we have the stock price by minutes, we attached the next 20 minutes price if the price existed. For example, assume the stock market is open on 10/01. The news published at 10:00 on 10/01 will be attached to the high market price at 10:00 and the low market price at 10:20 on 10/01. We choose 20 minutes as the same setting in AZFinText [15]. The reason for selecting the high market price at the published time and the low market price next 20 minutes is that I assume we at least could buy the stock with highest market price and sell it with the lowest market price. If the article published at 15:45 or 09:10 on 10/01, since the stock is not open or close, we ignore the news. Finally, we have 308 articles left. The amount of data is also insufficient, and the experiment result may not be convincing.

2.4 Original Plan

2.4.1 Stage 1: bag-of-words. The goal in stage 1 is reproducing the experiment result in AZFinText system [15]. AZFinText represented article in the bag-of-words with only proper nouns. The datasets were Yahoo Finance news articles and the S & P 500 index. Reproducing the experiment in AZFinText can prove that the data from Webhose.io or Reuters also works for stock price prediction. Moreover, we examined whether only using proper nouns helps the performance.

The experiment setup will follow the AZFinText paper. We will represent articles in the bag-of-words and use SVR to predict stock price. Finally, we evaluate the result by the rate of return. The strategy is buying the stock as the predicted price is greater than or equal to 1% movement from the stock price at the time the article was released. Then, sell the stock after 20 minutes. In stage 1, we will have four different results to compare.

- 1. Stock price
- 2. Stock price + News in bag-of-words
- 3. Stock price + News in bag-of-words without stopwords
- 4. Stock price + News in bag-of-words with only proper nouns

The expected performance will be 4 > 3 > 2 > 1 as the paper described.

2.4.2 Stage 2: word and sentence representation. In this stage, We are curious how modern deep learning techniques like word2vec, seq2seq, and CNN-LSTM language models could help improve the performance.

- 5. Stock price + News feature in average word2vec [9, 10]
- 6. Stock price + News feature in Skip-Thought vector[7]
- 7. Stock price + News feature in CNN-LSTM encoding [6] (may try to find pre-trained model)

The performance in stage 2 should be better than the one in stage 1.

3 EXPERIMENT

3.1 Data processing

We remove stop words, punctuation, number, low-frequency words, and high-frequency words in the article. The threshold of the low and high frequency are 10 and 100 for the Webhose.io dataset and 5 and 50 for the Reuters dataset.

3.2 Tasks

We have three different tasks. The classification task is to predict if the price goes up. The regression problem is to predict the price difference. The final one is applying a simple strategy to see what we can gain. The strategy is trading as the predicted price difference is positive. We use logistic regression, ridge regression, and SVR in the experiment.

- Classification task: close price goes up or down
- Regression task: close and open price difference
- Trading strategy task: compute return using the prediction price

The reason not using the same strategy as in AZFinText is for all models we could not have 1 % return. Hence, we have 0 return rate for all models and features. So the result is not comparable as well. Hence, we start with a more straightforward strategy.

3.3 Features

We experiment five different features on our tasks. Most of the features are described in section 2.4. However, I didn't use stock price features in the experiment. The idea is to see how good a text feature can contribute.

- Bag-of-words: The bag-of-words features are the words count with L2 norm normalization. We tried the bag-of-words with and without stop words. We also experimented using bag-of-words with tri-gram.
- Bag-of-words with pos tag filtering: We tag our words using NLTK [8] and only pick the words in the selected tags. We explore several different settings as below.
  - Proper noun: ['NNP', 'NNPS']
  - Noun: ['NNP', 'NNPS', 'NN', 'NNS']
  - Noun and verb: ['NNP', 'NNPS', 'NN', 'NNS', 'VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ']
  - Noun and adj: ['NNP', 'NNPS', 'NN', 'NNS', 'JJ', 'JJR', 'JJS']
  - Noun, verb, and adj: ['NNP', 'NNPS', 'NN', 'NNS', 'VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ', 'JJ', 'JJR', 'JJS']
- tf-idf: We compute the tf-idf in our corpus without stop-words.
- Average word2vec: We use Gensim [12] with two pretrain models, Glove[11] and Google word2vec[9, 10]. The word embedding size is 300.
- Skip-thought vector[7]: Only the first 100 words are feed to avoid zero embedding vector. I used two pretrained TensorFlow[3] model, unidirectional and bidirectional model[7] to generate the features. The feature size is 2400.

3.4 Evaluation

Since the data is time series, we can't apply standard K-fold cross-validation to our model because we can have the forward bias.

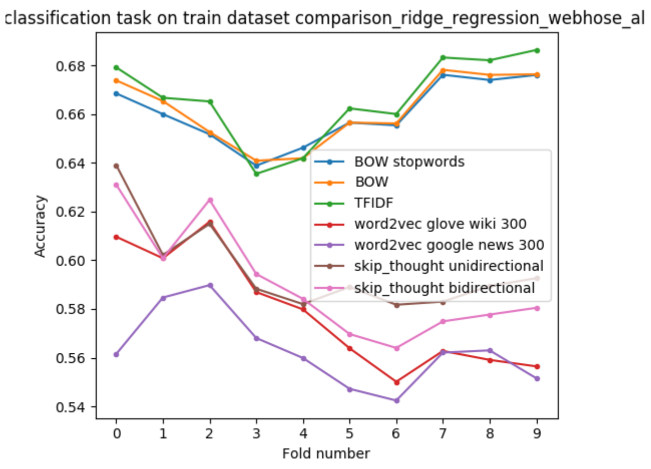

Figure 7: Classification task on Webhose.io train data using Ridge regression

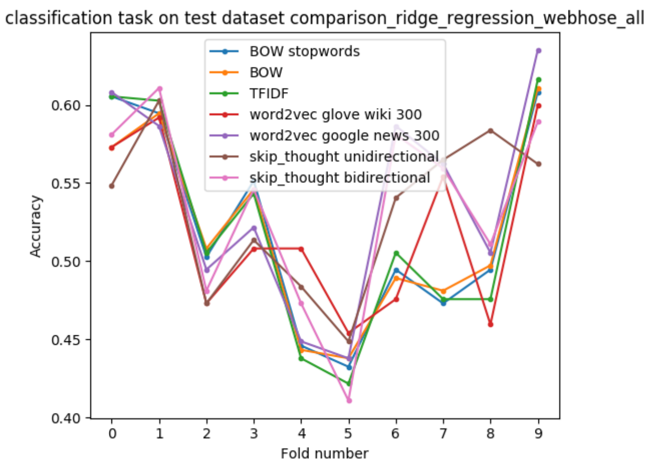

Figure 8: Classification task on Webhose.io testing data using Ridge regression

Hence, we use K fold time series cross-validation instead. We choose K as 5 for Reuters and K as 10 for Webhose.io dataset. We built word dictionary and scaled our features only on the observable dataset. Then, we predict the dataset on the next fold. If there are new words in the article, we ignore them. We evaluate accuracy for the classification task, mean square error for the regression task and return in dollar for trading strategy task.

4 RESULT

4.1 Features comparison

We evaluate the performance with different features. Figure 7 and 8 show the result on Webhose.io dataset. We can found bag-of-words and tf-idf feature outperform word embedding feautes in training data. This because the the discrete word features often capture some specific words and the model will overfit on that words. Hence, we can find in Figure 8 the discrete word features do not perform as it did in training data. The word analysis on the model could be found in Figure 9 and 10. In Figure 10, we can see the term "Ruth Porat", CFO of Google, and "Jeff Bezos", CEO of Amazon are two of the most important positive terms. This is because in July to Oct 2015 Google and Amazon's stock price grew a lot. Then, the model learned their names as positive term. However, it doesn't imply anything in the future. It is the reason for the bad performance.

The result for Reuters can be found in Figure 11 and 12. There is not much difference between the features. Several reasons could cause it. For example, different companies share the same text feature as mentioned in section 2.2 could make it happened. Insufficient data, 308 articles, can make things worse.

4.2 Models comparison

We exam the performance using different models in Figure 13, 14, 15 and 16. In Figure 13, SVR has the best performance. It may explain the reason to choose SVR in AZFinText [15]. However, the result is opposite on Reuters data as shown in Figure 15 and 16. Ridge regression performs stable among all case.

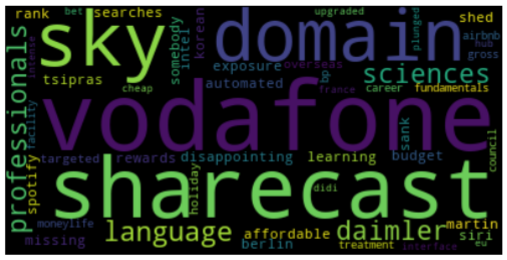

Figure 9: Negative terms on Webhose.io data by the weight of ridge regression

4.3 Pos-tag filter comparison

We exam the performance among different selected pos-tag filter in Figure 17 and 18. The result shows choosing only proper noun has little difference with other bag-of-words features. The good things for SVR and proper noun are that it is not easy to overfit. This is because the feature size is small by filtering out other terms.

4.4 Trading

Figure 19 shows the performance of trading using SVR and different pos-tag filter. We can find with only proper we can have profit in fold 3 (the only time we can gain among all models) but lose in fold 2. However, since we don't have much data, we can not conclude the result in the long run.

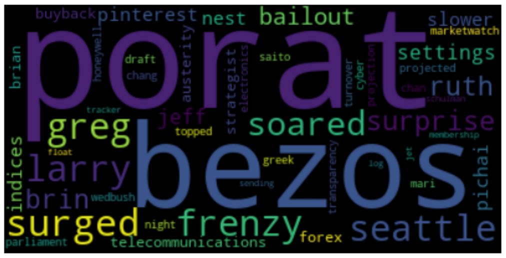

Figure 10: Positive terms on Webhose.io data by the weight of ridge regression

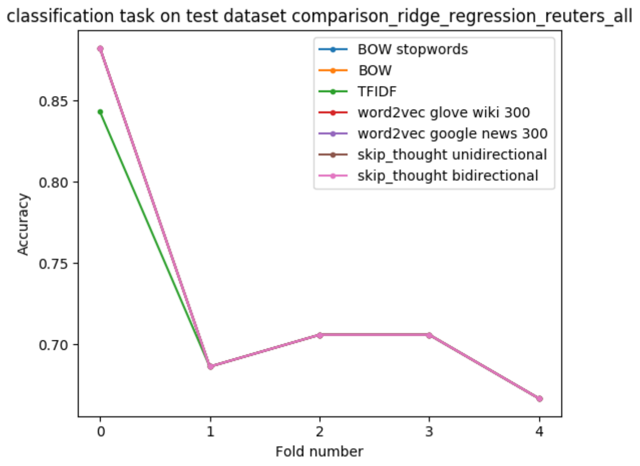

Figure 12: Classification task on Reuters testing data using Ridge regression

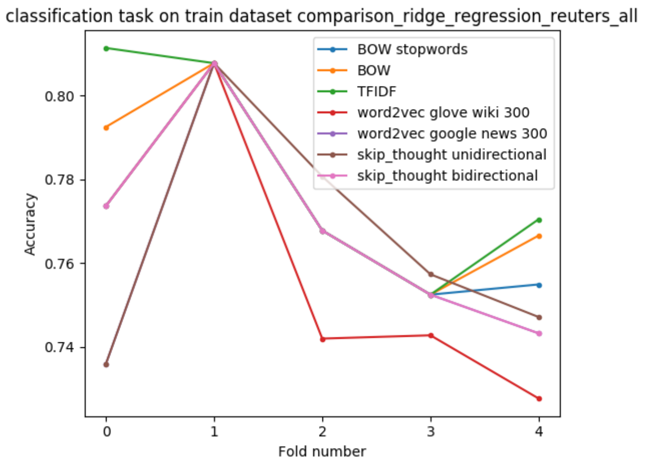

Figure 11: Classification task on Reuters training data using Ridge regression

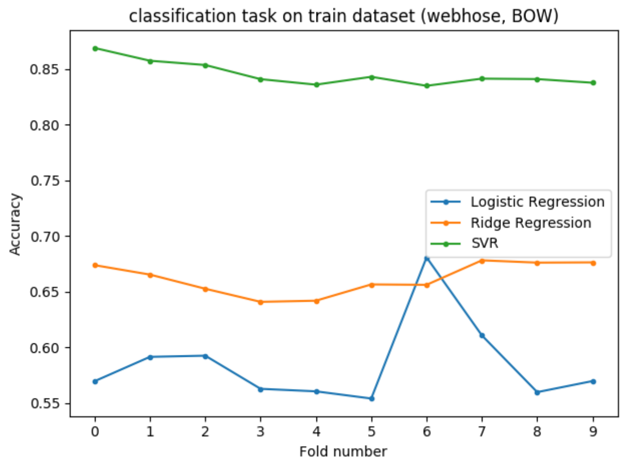

Figure 13: Classification task on Webhose.io training data with BOW features

5 CONCLUSION

In this project, I crawled and built data pipeline for Webhose.io and Reuters datasets for stock price prediction using text features. Experiments are performed on three tasks and five different features to explore features contributions on stock price prediction. Among all features, there is no significant difference in testing data. However, continuous text features have better generality than discrete features like bag-of-words. Among the different models, ridge regression is the most stable one. We can not have any gain with the model. However, we could earn with SVR but lose money at a different time. Among all pos-tag filters, the difference is not huge. Selecting only proper noun generates less feature size which is useful for avoiding overfitting. We also described why data qualities, insufficient data, various source, and ambiguous company identity, are still significant issues. Due to the inadequate Reuters data, the trading experiment result cannot be concluded quickly. The only thing can say is the model is not likely to make money.

REFERENCES

[1] 2018. IEX API. (2018). https://iextrading.com/developer/docs/#chart
[2] 2018. Reuters. (2018). https://www.reuters.com/resources/archive/us/
[3] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). https://www.tensorflow.org/ Software available from tensorflow.org.
[4] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2014. Using Structured Events to Predict Stock Price Movement: An Empirical Investigation. In *EMNLP*.

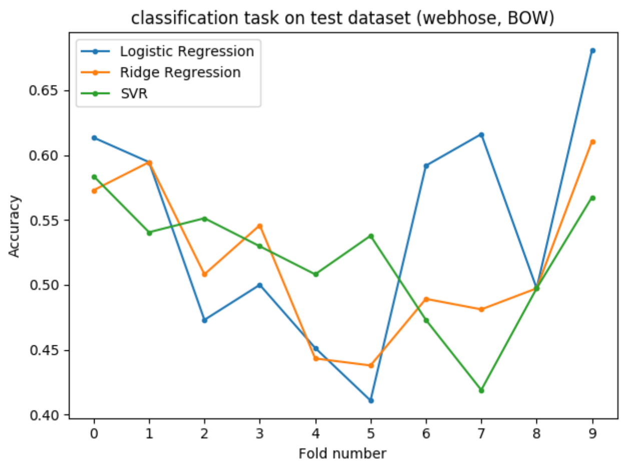

Figure 14: Classification task on Webhose.io testing data with BOW features

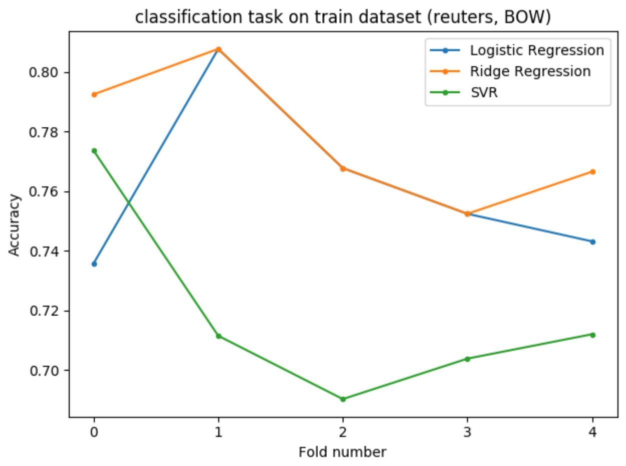

Figure 15: Classification task on Reuters training data with BOW features

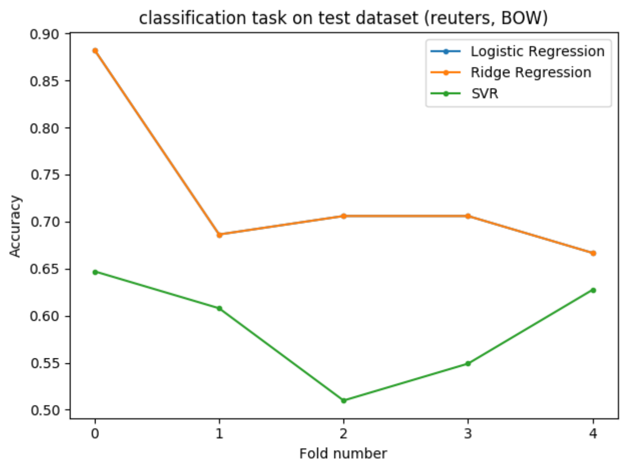

Figure 16: Classification task on Reuters testing data with BOW features

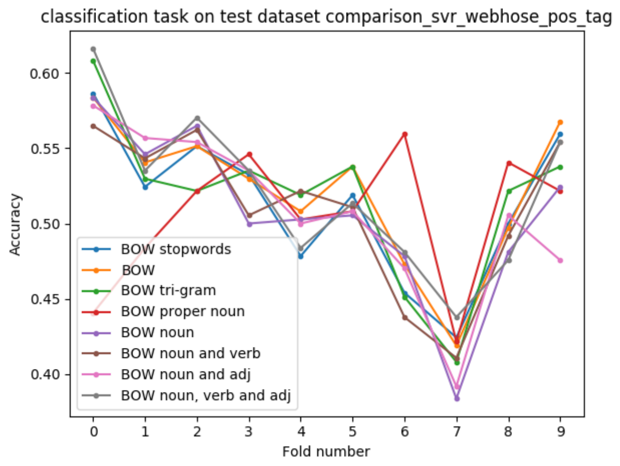

Figure 17: Classification task on Webhose.io testing data using SVR with different pos-tag filter

[5] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2015. Deep Learning for Event-driven Stock Prediction. In *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI'15)*. AAAI Press, 2327–2333. http://dl.acm.org/citation.cfm?id=2832415.2832572

[6] Zhe Gan, Yunchen Pu, Ricardo Henao, Chunyuan Li, Xiaodong He, and Lawrence Carin. 2016. Unsupervised Learning of Sentence Representations using Convolutional Neural Networks. (11 2016).

[7] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-Thought Vectors. In *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.). Curran Associates, Inc., 3294–3302. http://papers.nips.cc/paper/5950-skip-thought-vectors.pdf

[8] Edward Loper and Steven Bird. 2002. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1 (ETMTNLP '02)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 63–70. https://doi.org/10.3115/1118108.1118117

[9] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* abs/1301.3781 (2013). arXiv:1301.3781 http://arxiv.org/abs/1301.3781

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

[11] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *In EMNLP*.

[12] Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, 45–50. http://is.muni.cz/publication/884893/en.

[13] Philippe RÃĨmy. 2014. Financial News Dataset from Bloomberg and Reuters. https://github.com/philipperemy/financial-news-dataset. (2014).

[14] Evan Sandhaus. 2008. The New York Times Annotated Corpus. https://catalog.ldc.upenn.edu/LDC2008T19. (2008).

[15] Robert P. Schumaker and Hsinchun Chen. 2009. A quantitative stock prediction system based on financial news. *Information Processing & Management* 45, 5 (2009), 571 – 583. https://doi.org/10.1016/j.ipm.2009.05.001

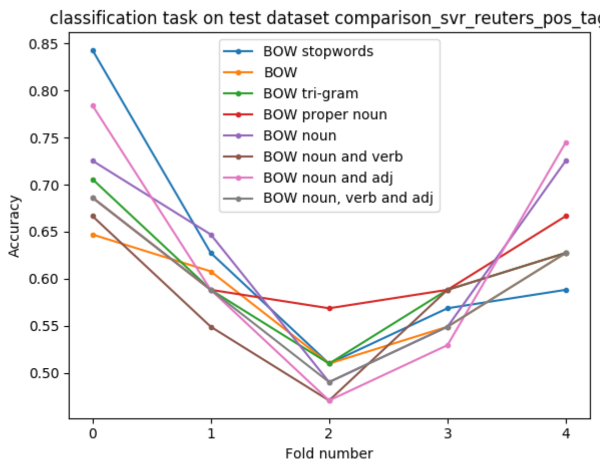

Figure 18: Classification task on Reuters testing data using SVR with different pos-tag filter

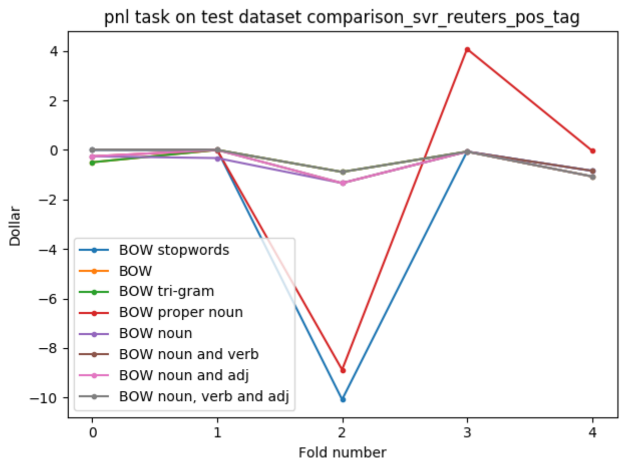

Figure 19: Trading task on Reuters testing data using SVR with different pos-tag filter

[16] Webhose.io. 2018. Webhose.io. Available at: https://webhose.io/datasets/ accessed on 2018-09-17. (2018).
[17] Jinjian Zhai, Nicholas Cohen, and Anand Atreya. 2011. CS224N Final Project: Sentiment analysis of news articles for financial signal prediction. (March 2011).