Stocks Prediction by Headlines

Shay Sirek
Omer Zamir
Omri Shtamberger

Submitted as final project report for the NLP course, COLMAN, 2021

Contents

1 Introduction

The project is about classifying daily stock trend based on news headlines, using multiple sentiment analysis techniques. We chose this problem since we are interested in using it for a later project which will be based on this project.

1.1 Related Works

1. Daily News for Stock Market Prediction Dataset
2. Transfer Learning - Sentiment Analysis for Financial News
3. Text Classification model using PyTorch
4. Sentiment Analysis with BERT and Transformers by Hugging Face using PyTorch and Python

2 Solution

2.1 General approach

Our general approach was using multiple sentiment analysis techniques and different models until we reach a model that preforms good enough for us. We found an existing dataset on Kaggle of top 25 daily news headlines, and a label to tell whether the DJIA stock went up or not, ranging over 8 years.

After we found our dataset, we had to find which model works best for our task. Our options were using BERT(1), LSTM(2), LSTM with transfer learning, N-Gram(3) document term matrix(4).

We wanted to test each model option and compare their performance. We planned using BERT, expecting it would perform well for the task, then we tried using LSTM with and without transfer learning but both over-fitted, and finally we tried using N-Gram.

2.2 Design

We tried using BERT incorporating a logistic regression layer. We found BERT-base-uncased(5) pre-trained model from a library called transformers by Hugging Face(6), and used it to extract features from headlines. The model we used produced 768 features that we gave as an input to a logistic regression layer. We also tried using different pre-trained BERT models like Distil-BERT(7), BERT-base-cased(8).
We expected it would perform well, but it didn't so we had to try other alternatives.

Since BERT failed to perform the task, we tried other neural network models based on LSTM.
For our LSTM models we used Torch-text(9) and GloVe(10) to build the vocabulary and used its pre-trained weights as an embedding layer. The first model includes an embedding layer with a size of 100, 2 bi-directional LSTM layers each with hidden size of 256, and finally a linear layer with Sigmoid activation function, which outputs 1 neuron representing the probability of the label. We trained the model with Binary Cross Entropy loss function and used Adam optimizer(11). But it did not perform well as we excepted.

We assumed that our model could learn more when using a different dataset of sentiment analysis for financial news, then we tried using a transfer learning approach.
We trained a LSTM model on that dataset, which includes news headlines and a sentiment - positive, negative, neutral. When training the model we used Cross Entropy loss function and used Adam optimizer.
We built a second model based on the pre-trained model. This model includes an embedding layer with a size of 100, sentiment extraction using the pre-trained model. It also has a linear layer, with Sigmoid activation function which gets 3 neurons representing the sentiment from the first pre-trained model, and outputs 1 neuron representing the probability of the label. When training this model we used Binary Cross Entropy loss function and used Adam optimizer. We again saw a bad performance result.

After considering and trying multiple approaches that involve neural network models, we decided to try a simpler approach. We decided to try using N-Gram in our next model attempt. The model used tri-grams to extract document term matrix and flattened it into a logistic regression layer. This model performed the best, for reasons we will discuss later.

3 Experimental results

We split our dataset into two subsets: train and test. We chose accuracy as a measurement metric because we were dealing with classification task. We sometimes used precision and recall to see the full picture.

Following are tables of our experimental result:

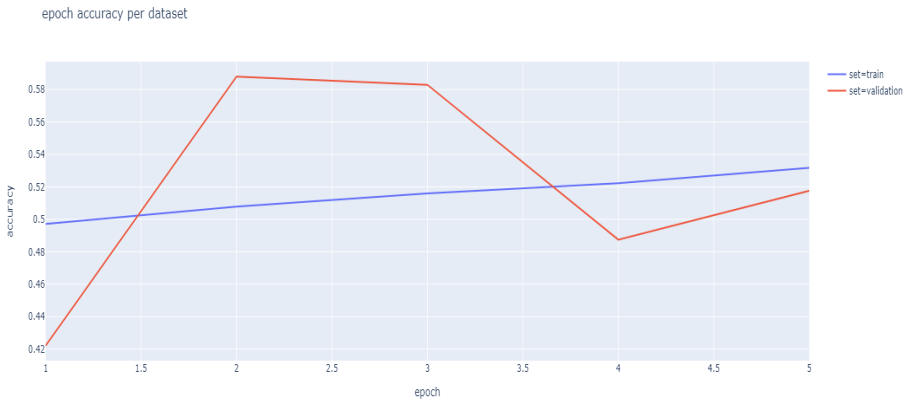
3.1 Models table

Model	Ratio	Epochs	Training time
BERT with Logistic Regression	80:10:10	5	1.52sec
LSTM	70:30	3	2.60sec
LSTM with Transfer Learning	70:30	3	3.12sec
Trigrams with Logistic Regression	75:25	3	59.40sec

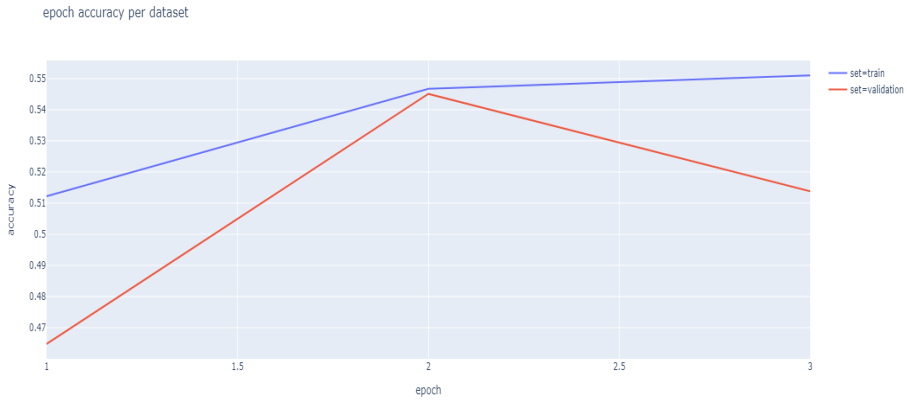
3.2 Accuracy

Model	Train	Test
BERT with Logistic Regression	0.532	0.543
LSTM	0.551	0.514
LSTM with Transfer Learning	0.529	0.469
Trigrams with Logistic Regression	1.000	0.844

3.3 BERT accuracy

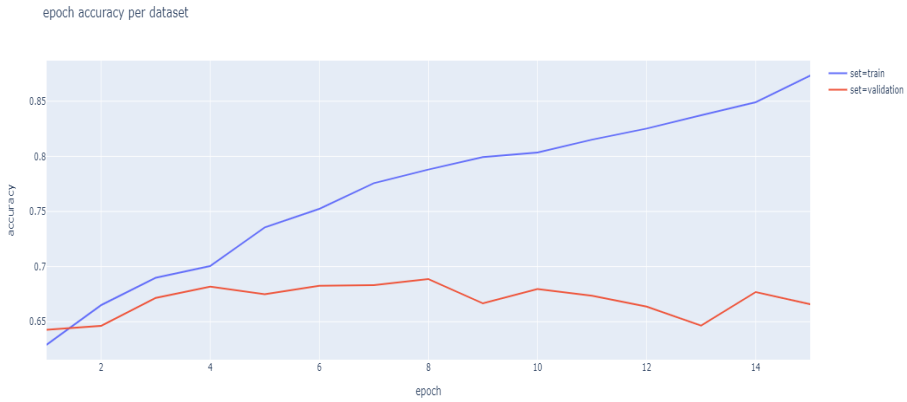


3.4 LSTM accuracy

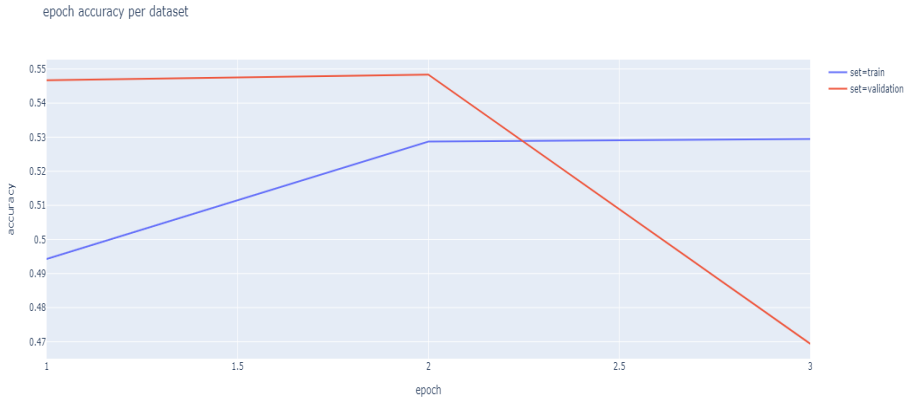


3.5 LSTM with Transfer Learning accuracy

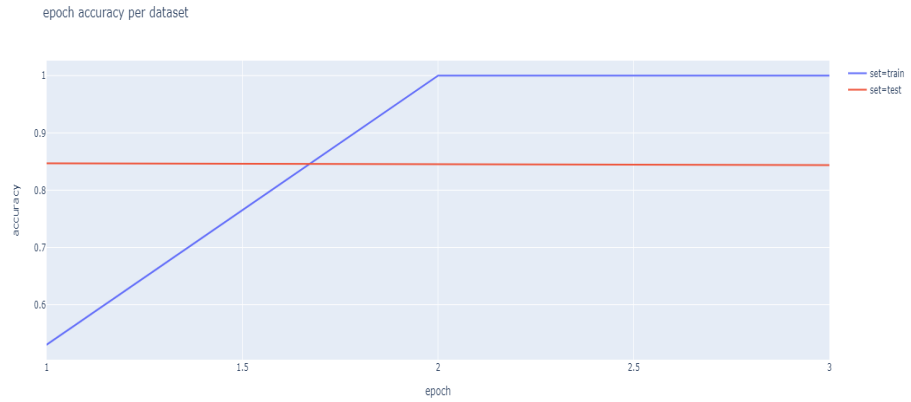
Base model



Final model trained with base model



3.6 Tri-grams with Logistic Regression accuracy



4 Discussion

Regarding the usage of the dataset, we joined the top 25 headlines. For BERT we used a stemmer to improve the quality of the vectorized headlines representation. For LSTM we used a tokenizer called spacy, and built our vocabulary with GloVe.

We started using BERT-base-uncased and had bad results, and found about Distil-BERT online, which is a "smaller" version of BERT, and tried it too.

Since our models were over-fitted we consider using weight decay regularization(12), but it didn't improve our result.

The N-Gram based model unexpectedly and unreasonably performed much better. We observed that all other models performed much worse, and we also saw online notebooks with other various models performed badly. We have several assumptions why that could be. We think the dataset might be the cause for that, since the quality of the data wasn't good enough. It could also be that the headlines are not representative enough of this task.

Based on our assumptions regarding the dataset we used for this project, we think it's better for future projects to collect our own data.

5 Code

`https://github.com/omier/NLP-final-project`

Bibliography

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[2] "Lstm." [Online]. Available: https://en.wikipedia.org/wiki/Long_short-term_memory

[3] "N-gram." [Online]. Available: https://en.wikipedia.org/wiki/N-gram

[4] "Document term matrix." [Online]. Available: https://en.wikipedia.org/wiki/Document-term_matrix

[5] "Bert base uncased." [Online]. Available: https://huggingface.co/bert-base-uncased

[6] "Hugging face transformers." [Online]. Available: https://huggingface.co/transformers/

[7] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," 2020.

[8] "Bert base cased." [Online]. Available: https://huggingface.co/bert-base-cased

[9] "Torch-text." [Online]. Available: https://pytorch.org/text/stable/index.html

[10] "Glove." [Online]. Available: https://nlp.stanford.edu/projects/glove/

[11] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[12] Z. Xie, I. Sato, and M. Sugiyama, "Stable weight decay regularization," 2021.