Project Phase 2 Report - Data Modeling

Team Members: Arno Dunstatter, Ethan Reyna, Daniel Fernandez

Our GitHub repository can be found here.

Building and Comparing Models

In this phase of our project, our group worked on developing our own LSTM model and working with pre-trained models that would help us in our journey to analyze and determine the general financial sentiment of article headlines. We first began this phase by researching different libraries we could use to determine sentiment based on the dataset we put together and the preprocessing we went through in the first phase. In our research, we were able to find three different pretrained models that observed tokenized sentences and, using a number of NLP factors, determined a sentiment value that we could then divide into positive and negative final sentiments. These models will be used to determine the overall value of our model's accuracy in comparison to methods that currently exist and are used in real-world applications. The models we found include TextBlob, NLTK's VADER, and FinBert.

LSTM is an artificial recurrent neural network (RNN) architecture. The main reason we decided to use this model as opposed to standard feedforward neural networks, was that it also has feedback connections, meaning it could process not only single data points but entire sequences of data. However, in implementing this with Python code, we could not find a way to properly get ternary classification as we had intended, as such, we decided to settle on just getting binary classification. So, for the purpose of comparing the various models, we utilized a subset of our total data that contained only positive or negative headlines. Additionally, the other models, while they could produce ternary classification, were implemented in such a way as to only produce binary classification between negative and positive sentiments. Gridsearch was performed using embedded for-loops to cycle through various hyperparameter settings for layers, units, dropout rate (to prevent overfitting), activation functions, batch, learning rate, and epochs and thereafter select the best performing model configuration as evaluated by accuracy. 10-Fold Cross Validation was performed on the best 25 models,

with the very best model having an average accuracy of about 85% and as such we were able to strike a decent balance between the Bias-Variance Tradeoff.

Textblob is a Python library for processing textual data which provides a consistent API for diving into common NLP tasks. Specifically for our use, we apply the polarity function, which takes in a “TextBlob” and determines the overall sentiment on a value of -1 to 1. We started by running our data frame of financial headlines through a loop that would change all headlines to ‘textblobs’ and recorded the polarity value for that given headline. We then appended the final 1 or -1 value to the end of the overall data frame. Once this loop finished, we determined the accuracy of this library’s function by comparing it to the original sentiment values. It was determined that this model’s implementation was able to determine the correct sentiment of the headlines 72% of the time. This was not a very high accuracy value, however, this was to be expected, as the polarity function was not meant to only determine financial sentiment, it's often used for much more general uses.

VADER is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. Similar to the TextBlob polarity tool, this is not specially trained for financial sentiment analysis, however, it will still be helpful to know as a comparison. For application to our data frame of headlines, we decided to use the SentimentIntesityAnalyzer, and pull the polarity scores from the data that it produces. This analyzer outputs different values of intensity in three different categories: positive, negative, and neutral, each taking a portion of 1 total value depending on the sentiment of each statement. For our use, we’re only looking at positive and negative sentiments, therefore we halved the neutral value and added it to the other two so we could find which was more present. With the two final values, we determined the higher of the two, and used that to find Vader's final sentiment prediction for each headline, and attached a 1 or -1 at the end of the data frame. Similar to what we did with the TextBlob analysis, we use the same practice to determine the accuracy of this function. In the end, we actually found that the Vader sentiment analyzer had a 69% accuracy rating when analyzing our dataset. This also is not very surprising as it was trained to specifically determine the polarity of social media posts. However, this did mean we had a lot of room to improve upon current public sentiment analyzing models.

FinBERT is an adaptation of Google's BERT that was developed by Dogu Araci for his Masters Thesis. It essentially further trained the BERT model on a dataset of financial text to produce the specialized finBERT model. This model is hosted on HuggingFace from which we made use of the ProsusAI/finbert module for both tokenization and classification. By default the model assigns polarity scores for positive, negative, and neutral classes, however since the LSTM could only produce binary classification, for the purpose of comparison we simply divided the neutral score in half and added that halved value to both the positive and negative scores and then classified that headline according to whichever score was greater, with the edge case of them being equal being counted as a negative sentiment. Ultimately this was our best performing model, achieving an average predictive accuracy of approximately 96% over 21 folds of 97 observations each (i.e. the entirety of all known to be positive or negative headlines).

Findings

Performance Evaluations

- Since we used neural networks and TextBlob, which is a rule-based approach, the only metric we could use to evaluate all of them was their accuracy:

	TextBlob	VADER	LSTM	FinBERT
Accuracy	71.97%	69.81%	85.435%*	95.9%
Details	Rule-based		Neural Network	
	Trained on General Data		Trained on Financial Data	

* Value of our best model's average accuracy

More on LSTM Performance

Below we show the top 10 LSTM Models and their hyperparameters (sorted by validation accuracy):

	ACCURACY	LOSS	LAYERS	UNITS	DROPOUT	ACTIVATION_FUNCTION	BATCH_SIZE	EPOCHS	LEARNING_RATE
0	0.891304	0.331752	2	25	0.25	sigmoid	50	5	0.0010
1	0.891304	0.318145	4	25	0.25	tanh	50	5	0.0010
2	0.880435	0.427110	1	25	0.10	relu	25	10	0.0100
3	0.880435	0.467468	2	25	0.05	sigmoid	25	25	0.0001
4	0.880435	0.354164	2	25	0.10	tanh	50	25	0.0001
5	0.869565	0.372019	1	12	0.25	relu	25	5	0.0010
6	0.869565	0.383393	1	12	0.25	sigmoid	25	5	0.0010
7	0.869565	0.511447	1	25	0.25	relu	25	25	0.0100
8	0.869565	0.499623	2	25	0.05	relu	25	25	0.0001
9	0.869565	0.504906	2	25	0.05	tanh	25	25	0.0001

Note: The information of all 2916 models can be found in the file model_info_with_AF.zip (a compressed csv file)

Checking the top 25 models with 10-fold cross validation:

```
Rechecking the 25 best models (in terms of accuracy):
Model 0 | ACCURACY: (69.565-88.043%) Average: 83.91300000000001%, LOSS: (0.36951-0.62956) Average: 0.43934
Model 1 | ACCURACY: (69.565-88.043%) Average: 84.674%, LOSS: (0.35245-0.63032) Average: 0.42599
Model 2 | ACCURACY: (69.565-88.043%) Average: 83.261%, LOSS: (0.37165-0.6296) Average: 0.4557
Model 3 | ACCURACY: (69.565-89.13%) Average: 84.45700000000001%, LOSS: (0.34842-0.62955) Average: 0.43337
Model 4 | ACCURACY: (69.565-88.043%) Average: 84.02199999999999%, LOSS: (0.34848-0.62955) Average: 0.44634
Model 5 | ACCURACY: (69.565-89.13%) Average: 84.348%, LOSS: (0.33244-0.62958) Average: 0.42942
Model 6 | ACCURACY: (72.82600000000001-88.043%) Average: 84.783%, LOSS: (0.35415-0.66505) Average: 0.4373
Model 7 | ACCURACY: (69.565-88.043%) Average: 84.239%, LOSS: (0.34924-0.62745) Average: 0.42835
Model 8 | ACCURACY: (69.565-88.043%) Average: 84.565%, LOSS: (0.34001-0.62956) Average: 0.43192
Model 9 | ACCURACY: (69.565-88.043%) Average: 84.565%, LOSS: (0.35622-0.6296) Average: 0.43346
Model 10 | ACCURACY: (69.565-88.043%) Average: 84.783%, LOSS: (0.35566-0.62898) Average: 0.43365
Model 11 | ACCURACY: (69.565-88.043%) Average: 84.783%, LOSS: (0.34117-0.62963) Average: 0.42904
Model 12 | ACCURACY: (69.565-89.13%) Average: 85.0%, LOSS: (0.34247-0.62961) Average: 0.42866
Model 13 | ACCURACY: (69.565-88.043%) Average: 83.91300000000001%, LOSS: (0.36471-0.62953) Average: 0.43422
Model 14 | ACCURACY: (69.565-89.13%) Average: 84.239%, LOSS: (0.33131-0.62957) Average: 0.4278
Model 15 | ACCURACY: (69.565-88.043%) Average: 83.91300000000001%, LOSS: (0.36325-0.6296) Average: 0.448
Model 16 | ACCURACY: (75.0-88.043%) Average: 84.565%, LOSS: (0.32382-0.6359) Average: 0.43662
Model 17 | ACCURACY: (69.565-88.043%) Average: 84.45700000000001%, LOSS: (0.3331-0.6296) Average: 0.42753
Model 18 | ACCURACY: (68.47800000000001-89.13%) Average: 84.674%, LOSS: (0.34573-0.67182) Average: 0.43585
Model 19 | ACCURACY: (69.565-88.043%) Average: 84.02199999999999%, LOSS: (0.36494-0.62953) Average: 0.44351
Model 20 | ACCURACY: (69.565-88.043%) Average: 85.435%, LOSS: (0.33257-0.62977) Average: 0.42395
Model 21 | ACCURACY: (69.565-89.13%) Average: 84.891%, LOSS: (0.3521-0.62956) Average: 0.42879
Model 22 | ACCURACY: (69.565-88.043%) Average: 84.565%, LOSS: (0.35739-0.62954) Average: 0.43194
Model 23 | ACCURACY: (69.565-88.043%) Average: 84.783%, LOSS: (0.35974-0.62938) Average: 0.42712
Model 24 | ACCURACY: (69.565-88.043%) Average: 84.13000000000001%, LOSS: (0.35412-0.62957) Average: 0.42711
```

Of the top 25 models (unique hyperparameter combinations) in the grid search:
The best 2 models (based on average accuracy):

```
Model 20 | ACCURACY: (69.565–88.043%) Average: 85.435%, LOSS: (0.33257–0.62977) Average: 0.42395
Model 12 | ACCURACY: (69.565–89.13%) Average: 85.0%, LOSS: (0.34247–0.62961) Average: 0.42866
```

The best 2 models (based on its lowest accuracy):

```
Model 16 | ACCURACY: (75.0–88.043%) Average: 84.565%, LOSS: (0.32382–0.6359) Average: 0.43662
Model 6 | ACCURACY: (72.82600000000001–88.043%) Average: 84.783%, LOSS: (0.35415–0.66505) Average: 0.4373
```

The maximum accuracy that was achieved was `89.13%` (which 6 of the 25 models achieved this accuracy at least once).

Most of the models had an accuracy range (min–max accuracy) of exactly `69.565–88.043%` and an average accuracy of about `84%`.

Most of the models approximately had a loss range of `(0.35–0.63)` and an average loss of about `0.43`.

Additional Note for Further Research

Had we had more time, we may have also implemented a hybrid approach, using the LSTM and a rule-based approach based on the tags that we had created and the top words in our dataset with a notable sentiment. This hybrid approach was conceived, but was not fully understood how to implement properly as each of the neural network grid searches took nearly an entire day to run. However, in theory, this hybrid approach would have likely led to even better results.