1 Introduction

The field of Natural Language Processing (NLP) has raised much attention in recent years. As many models are slowly attaining close to human level understanding of textual data, the field of financial forecasting has picked up on this development and a new research domain, Natural Language Financial Forecasting (NLFF) came up [1]. While different models have been trained with reasonable success for intra-day trading [2] we aim to develop a model that captures larger scale changes in returns that can be measured with daily returns. Bidirectional Encoder Representations from Transformers (BERT) can be considered the state of the art model used in NLP. [3]. The aim of this project is to show the application of the BERT network for semantic analysis of financial press statements. To achieve such, financial press statements were gathered with web crawling and labeled according to returns of given dates. These data was used to train a BERT classification model. We distinguished three different classes; increase, decrease and non-relevant to account for the three main types of financial news. The performance of the NLP model was compared to established Time Series prediction models for financial return predictions. Time Series models included in our analysis were the ARIMA and LSTM models. As financial returns as well as financial press releases contain a large amount of background noise we tried to isolate the impact of press statements on the stock returns by proposing a new way of labeling gathered text data. We were able to train different models that capture the semantics in the financial press statements and reach an overall accuracy of 87.1% with 66.0% accuracy per class.

2 Data Generation

2.1 Web Crawling and Text Collection

The generation of text data sets is often performed via web crawling. In order to train a meaningful model, one requires high quality data. The source of the used financial press statements is, therefore, an integral part of successfully designing a press statement classifier. As many financial news services publish only a limited amount of articles with open access summary news portals such as https://www.prnewswire.com/ or https://finance.yahoo.com/ were investigated. As these portals use different mechanisms to ensure the inquirer to be human, the Selenium web driver open source library [4] was used. While one could find much more longitudinal data for single stocks on the prneswire platform, most of them were of poor quality. Most of the articles contained only noise and were not related to the stock at all. The yahoo! finance platform contains a much smaller quantity of articles per stock queried, but the articles found were of higher quality and directly linked to the stock. Therefore, ten articles per stock were collected from the yahoo! finance platform for all the stocks listed in the S&P500. It is to be noted that a much better model might be trained if one applies fine tuning of the model to the respective stock, but due to the small amount of articles collected per stock, it was decided to use a general model for all stocks.

2.2 Obtaining Stock Prices

As discussed before, the aim of this projects is to classify financial press statements based on their effects on the stock price returns. Accordingly, the target variables of the models are daily stock returns. Consequently, daily stock prices are required to calculate the return of each stock. The main portal for obtaining daily stock prices is https://finance.yahoo.com/. However, contrary to financial press releases, there is no need to make web crawling to obtain prices. It is possible to obtain all prices with the help of the **yfinance** package. After collecting S&P500 tickers with HTML scrapping, one can easily download all historical data of listed stocks for a given time horizon. The data shows, taht all of the ten found articles for each stock were recently, namely in 2020 released. Hence, it is logical to download historical prices for the year 2020 for each S&P500 index stock.

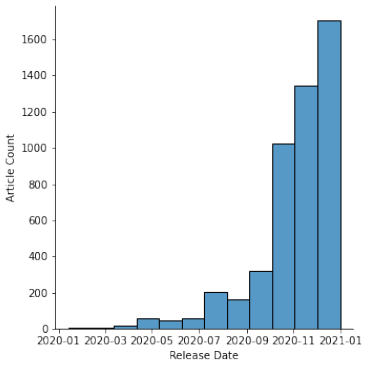

Figure 1: Release date histogram of financial press news.

After collecting historical prices for all S&P500 stocks, one needs to choose most logical historical price variable among opening, closing and adjusted closing prices. In this project, since unadjusted prices are affected from the dividend distributions and might not represent true nature of the news, it is rational to use adjusted closing prices to calculate daily stock returns.

2.3 Data Cleaning and Preprocessing

HTML strings found by the web crawler were parsed directly with the help of the Selenium tool to get the article title, subtitle, date and text. Before saving them, special characters were removed. Furthermore, all line or tab breaks were replaced with white spaces and quotation marks were unified to avoid complications when saving or loading the text data. As not much information is considered to be lost, all text was converted to lower case. This action was taken in relation to the much higher performance of the BERT lower case model. No further data cleaning steps classically related to NLP tasks, such as stemming, lemmatizing, stop words removal etc. were required. This is due to the nature of the BERT model applied as explained in subsection 3.1

While most of the articles found are relatively short and have word counts at around 500, some articles are extremely long and present word counts of 17.5k.

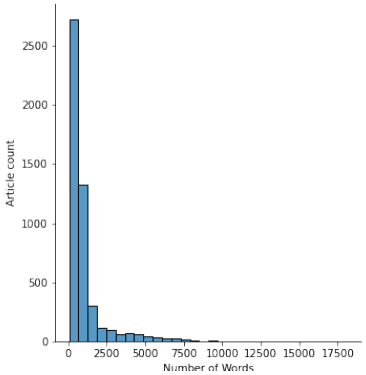

Figure 2: Distribution of text lengths in the collected articles; mostly short articles were found.

Daily stock returns can be calculated in two ways: Simple daily return and daily log return. In this project, daily returns are calculated as simple returns in all of the models as shown below:

$$r^s_{t+1} = \frac{S_{t+1} - S_t}{S_t}$$

To apply the BERT classification model, we were required to bring the data in a tabular format with labels required for supervised training. Defining labels on training data is a difficult task and several approaches were investigated:

- **Binary of return:** The most naive approach resulted in a binary classification task where the simple sign of the return was used as label.

$$L_{a^s_t} = \begin{cases} \text{increase} & \text{if } r^s_{t+1} \geq 0 \\ \text{decrease} & \text{if } r^s_{t+1} < 0 \end{cases}$$

  Where L is the label of article a linked to stock s on day t, with return r.

- **Ternary of return:** One can distinguish the returns in up and down, but also in not significant changes. This was introduced to account for the huge amount of noise found in the articles. To define the thresholds for significance of returns, the normalized returns over the last year were analysed and a threshold of ± 0.05 was set.

$$L_{a^s_t} = \begin{cases} \text{increase} & \text{if } r^s_{t+1} > 0.05 \\ \text{not rel.} & \text{if } -0.05 \leq r^s_{t+1} \leq 0.05 \\ \text{decrease} & \text{if } r^s_{t+1} < -0.05 \end{cases}$$

- **Ternary of return, S&P500 corrected:** As overall good or bad trading days will have a substantial impact on the returns, even bad news for a specific stock could be annotated with a positive label if the general market increased on that day. This effect was mostly observed to render non relevant messages significant if they were published on good or bad overall trading days. Therefore, a correction of the relative returns by the relative S&P500 returns was applied.

$$L_{a^s_t} = \begin{cases} \text{increase} & \text{if } r^s_{t+1} - r^{snp}_{t+1} > 0.05 \\ \text{not rel.} & \text{if } -0.05 \leq r^s_{t+1} - r^{snp}_{t+1} \leq 0.05 \\ \text{decrease} & \text{if } r^s_{t+1} - r^{snp}_{t+1} < -0.05 \end{cases}$$

  Again, the respective cut-off values were defined by visual inspection of the returns distribution of S&P500 stocks over the last year.

With increased complexity of the label definition, more assumptions flow into the process. As mentioned before, defining labels is a difficult task and all outcomes are subject to bias. Nevertheless, only labels which isolate the effect of the news release on the return in the best possible manner can lead to a meaningful model. Therefore, it was decided to pursue the corrected ternary return prediction mentioned last.

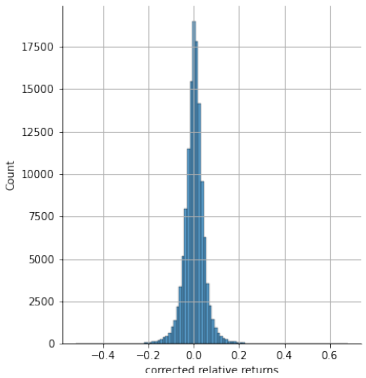

Figure 3: Distribution of observed relative two daily returns in the year 2020. Returns were corrected by the relative two daily returns of the S&P500. As the distribution shows little skew the cut-off values were defined symmetric. Furthermore, the steep increase at ± 0.05 was decided to be a reasonable threshold.

The labels engineered that way are unfortunately not balanced as one can see in Figure 4. The most represented class is the not relevant class. This is best understood when considering the nature of news releases, as only in rare cases press statements will trigger significant responses from single stocks.

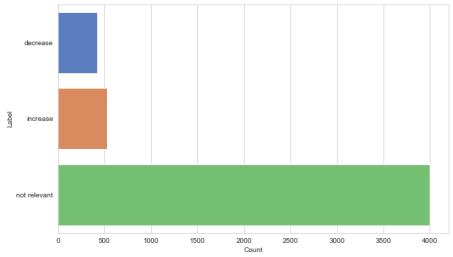

Figure 4: Bar plot of the number of articles found corresponding to each label annotated with it. Labels were constructed as discussed in subsection 2.3. The 'not relevant' label is the predominant label for the articles, where increase and decrease count 529 and 425 respectively.

3 Methods

3.1 BERT classification

The Bidirectional Encoder Representations from Transformers (BERT) model has changed the natural language processing (NLP) landscape drastically since its publication in 2018. The model uses transformer based NLP pre-training to be fine-tuned on specific text data for various tasks. The learning approach used by BERT is transfer-learning, where the model has been heavily pre-trained to learn word embeddings and general language representations. The pre-training performed is based on a bidirectional approach, where other pre-trained NLP models used to predict words in an unidirectional model (left to right, right to left or both). The first approach BERT takes is the one of a masked language model (MLM): The pre-training objective constitutes of predicting randomly masked tokens correctly. As a second pre-training step, the model received pairs of sentences. These sentences were either subsequent sentences that belong together or randomly drawn sentences from the training corpus. The model then needed to classify each sentence as being a new sentence or a following sentence. This enabled the model to learn word and sentence embeddings. The two pre-training tasks were performed in parallel to minimize the combined loss function.
This approach loses the directional issues and enables much more flexible language representation. These deep bidirectional representations enable easy application to various tasks as only the input and output layer of the model need to be altered when applying to a new task. Then all parameters in the model are fine tuned based on the specific task.

In the scope of this project, the BERT model was fine-tuned to perform text classification in order to capture the semantics in the financial news statement. To do so, the BertForSequenceClassification model was used for a ternary class prediction problem. The specific model used was the bert-base-uncased model to keep training times in a reasonable frame. The information loss due to lower casing is considered neglectable in press statements.

As the BERT model is pre-trained on a large corpus of real text, the nature of the transformer model will automatically account for widely known pre-processing steps such as stemming, lemmatizing, stop words removal etc [5]. The encoding layers in the network will neglect meaningless tokens in the text directly and reduce the text representation to the relevant information only. The pre-processing steps are replaced by the tokenization. The tokenization takes the input string and separates it into single words and subwords. These are then usd as tokens to be converted to the corresponding IDs of the vocabulary established during pre-training. The BERT model requires a fixed length of sentence as input. Shorter sentences will then be padded to the length with the PAD tokens. Unknown words are replaced by the UNK token.

As this project deals with a classification task, several widely used measures were applied to compare model performance. The most standard measure used is the accuracy score.

$$\text{Accuracy} = \frac{TP + TF}{TP + TF + FP + FN}$$

But with unbalanced classes a high accuracy score can be misleading, as simple predicting the dominant class in all cases can give a good performance already. Therefore, the per class accuracy takes the accuracy in each class and weights it equally.

$$\text{per class accuracy} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{TP_c + TF_c}{TP_c + TF_c + FP_c + FN_c}$$

Where $\mathcal{C}$ is the space of all classes in the task. Another measure to define classification performance is the F1 score. In binary classification the F1 score is the harmonic mean of precision and recall.

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

Similar to the per class accuracy, these values are calculated for every class and then averaged to give the F1 score of a multiclass task.

3.2 ARIMA Time Series model

In addition to the implementation of the BERT classification model, one requires a benchmark model. To compare the NLP approach with other established financial forecasting models, Time Series models are the matter of choice. The first that comes to mind for Time Series predictions is the ARIMA model. The Autoregressive Integrated Moving Average (ARIMA) model is a generalized parametric Time Series model which combines Autoregressive (AR) and Moving Average (MA) models. The AR part of the model captures the effect of the lagged observations on the current value of the Time Series and the MA part represent the effect of

lagged residuals calculated from lagged observations. Lastly, the Integrated (I) part is used to stationarize the Time Series when there is a trend with using differentiation [6]. The general ARIMA(p,d,q) model can be written as shown below:

$$\left(1-\sum_{i=1}^{p}\varphi_i L^i\right)(1-L)^d X_t = \mu + \left(1+\sum_{i=1}^{q}\theta_i L^i\right)\epsilon_t$$

One of the main advantage of parametric models such as ARIMA is the ability of providing robust predictions even with a limited number of past data. In the case of this project, there are around 250 daily returns data in accordance with the working days of a year. Hence, models that require a high number of past data for training might not perform well. Consequently, using ARIMA predictions as a benchmark model for next day return prediction is a natural choice. Implementation of the ARIMA models are usually a manual process. First of all, the graph of the daily returns Time Series of an example stock, **"AAPL"**, can be seen in Figure 5.

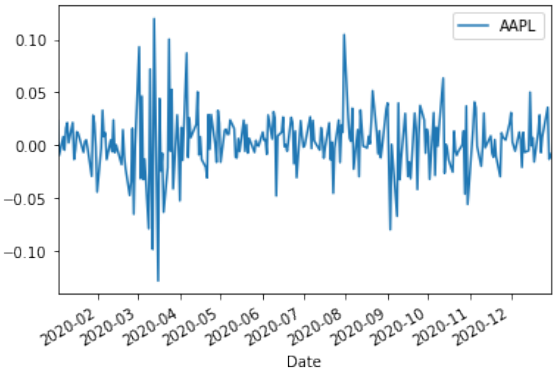

Figure 5: Time Series plot of "AAPL" daily simple returns in 2020.

Following that, Time Series data was divided into 3 datasets: Training, validation and test sets as shown in Figure 6. Training data is used to fit the ARIMA model and then obtained models are applied to validation set to optimize parameters (p,d,q). Afterwards, the best performing model is applied to the test set.

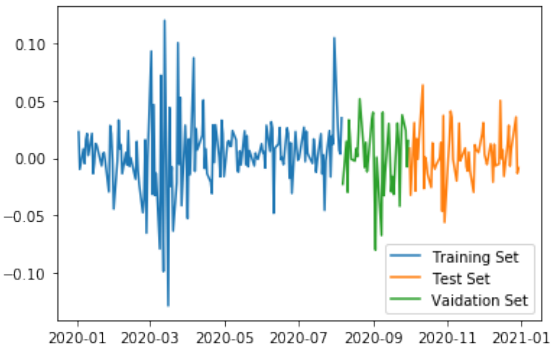

Figure 6: Division of "AAPL" daily simple returns data.

As mentioned before, optimization of parameters (p,d,q) are usually a manual process. As can be seen from the Figure 5, most of the stock returns, similar to graph of "AAPL", do not show any trend. Hence, it can be said that they are stationary in terms of mean and the differencing parameter d can be set to 0 for all stocks. Following that, to optimize the parameters p and q, one needs to analyze the ACF and PACF graphs of the training set. The ACF and PACF graphs of **"AAPL"** can be seen in Figure 7. AR(p) process usually has exponentially decreasing but significant ACF values until lag p. On the other hand MA(q), process usually has exponentially decreasing but significant PACF values until lag q. However, in the case of this many stocks, it is not possible to manually optimize the model for each stock. Hence, a couple of possible lag values are chosen from the example graph. Following that, the optimal parameters for each stock are found by using Grid Search.

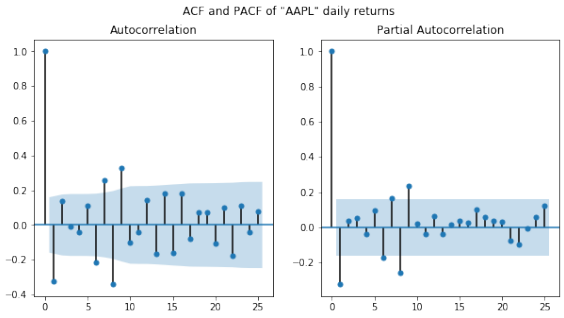

Figure 7: ACF and PACF graph of "AAPL".

One important factor to discuss here is that selection of the cost function for the optimization of parameters. There are two main options: Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). Since the model is trying to capture abnormal daily returns and RMSE punishes large errors, in this case it is logical to use RMSE of which the formula is given below:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(X_i - \hat{X}_i)^2}{N}}$$

Lastly, as discussed before, ARIMA is a robust method even when there is a scarce number of data. However, Time Series data needs to fit into some assumptions for ARIMA model to perform well. The most important assumption of the ARIMA model is that Time Series data should be stationary.
Stationarity can be defined in two ways: strong and weak stationarity. A Time Series process is strong stationary if it has same joint distribution function for all vectors. On the other hand, a process is called weak stationary if it has same mean and variance for all time points and the covariance between two points is only a function of lag. As can be seen from Figure 5, the daily returns process can be said to have the same constant mean over time [7]. However, this does not apply to the variance. One can see from the graph that the daily returns have volatility clusters. It means that higher volatility time points are next to each other and lower volatility time points are next to each other. Hence, it can be said that ARIMA model for daily return data might not perform as expected. Thus, another model to

feed BERT classification model with Time Series prediction might be needed. In the next section, an alternative to ARIMA model, LSTM, is discussed.

3.3 LSTM Time Series model

Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) which can also remember past sequence of Time Series data. Each LSTM is a system module which data streams are seized and stored. Each module has gates based on a sigmoidal neural network layer. These gates have the ability to discard or keep the data while connecting past data with the current one [6, 8]. Advantages of the LSTM compared to ARIMA model are that the LSTM model does not require Time Series process to be stationary and it can work with non-linear Time Series models. On the other hand, it requires bigger data sets to perform well. In the case of this project, each stock has around 250 past daily returns data and when one creates sequences data size gets even smaller. However, it can be seen from the Figure 8 that randomly selected stocks have positive correlation with each other in 2020.

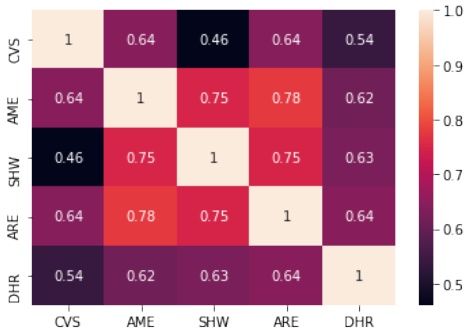

Figure 8: Correlation among randomly selected five stocks.

Consequently, it is logical to add other stocks' past daily returns as independent variable to increase performance of LSTM model. Lastly, the assessment metric used for LSTM model optimization is Mean Squared Error (MSE). Following the definition of the models, implementation and the results of the models will be discussed in the following section.

4 Results

4.1 ARIMA

As discussed in subsection 3.2, the daily simple returns dataset for each stock was divided into 60% training set, 15% validation set and remaining 25% were used as test set. Moreover, the parameters p and q were optimized with Grid Search. The options for p values were (1,2,3) and options for the q values were (0,1,2) based on the example ACF and PACF graph (d=0 as discussed before). After obtaining the optimized parameters, future daily stock returns were predicted with those parameters. Thus, daily stock return predictions were obtained from "10/1/2020" until "12/31/2020". However, to have a meaningful comparison with LSTM model which has predictions starting from "10/8/2020" 7 days of predicted returns were omitted from comparison.

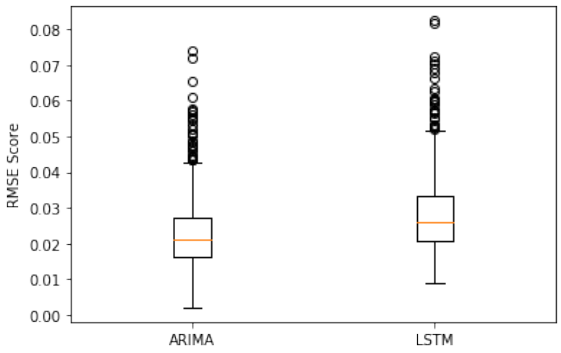

Figure 9: Boxplots of RMSE Scores of ARIMA and LSTM models.

As can be seen from Figure 9, ARIMA model predictions have an average RMSE of around 2%. Another important score that can be analyzed is accuracy of predicting sign of the next day return as our main BERT model is trying to predict the sign of the return. Figure 10 shows that accuracy scores of ARIMA models deviate around 50% and have a mean of 50% which is equal to random guessing.

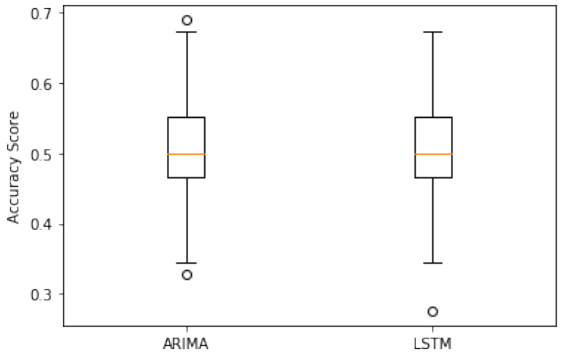

Figure 10: Boxplots of Accuracy Scores of ARIMA and LSTM models.

4.2 LSTM

Similarly to the previous section, there were around 250 daily stock returns for each stock. As independent variable the sequence of the last 20 days' stock returns of all stocks were taken. However, the predictions were still made for individual stocks to obtain better results. Thus, hyper-parameters (LSTM units, dropouts, epochs, batch size and optimizer) optimization has been done manually for an example stock. Obtained parameters were applied to all other stocks on the Google colab GPU to obtain faster results. Again we performed a train-test split, this time with the ratio 75:25. After fitting the model to training data, the daily stock return predictions from "10/8/2020" to "12/31/2020" were obtained for each stock. The RMSE score for the LSTM models were around 2.5% as can be seen in Figure 9. The comparison of the ARIMA and LSTM prediction in

terms of RMSE score can be seen in Figure 11. In the graph, values are calculated by subtracting LSTM RMSE scores from ARIMA scores. One can see that most of the time the difference is negative. Hence, it can be said that in terms of RMSE score ARIMA models are performing better than LSTM models.

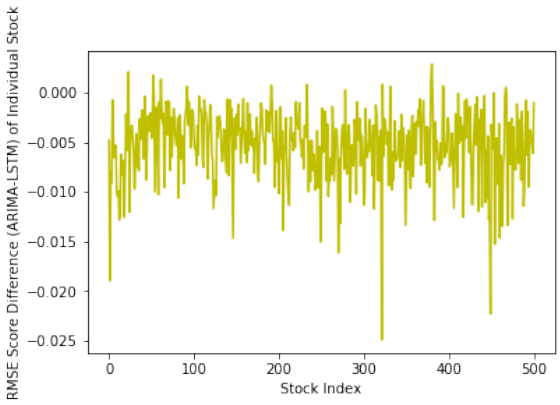

Figure 11: RMSE comparison (ARIMA-LSTM).

Following that, similarly to the previous section, the accuracy score of predicting the sign of the returns can be analyzed. Figure 10 shows the accuracy scores box plot for the LSTM models. Similarly to ARIMA models, the LSTM Models' accuracy deviates around 50% with a mean of 50%.
It can be said that both models are performing equivalently poor in terms of accuracy. It can be seen from Figure 12 that while for some stocks one model performs better than the other, the average of differences of accuracy scores is around zero. Thus, at this point, it is logical to conclude that the benchmark model (ARIMA) to compare to the BERT model is performing better than the other Time Series model (LSTM).

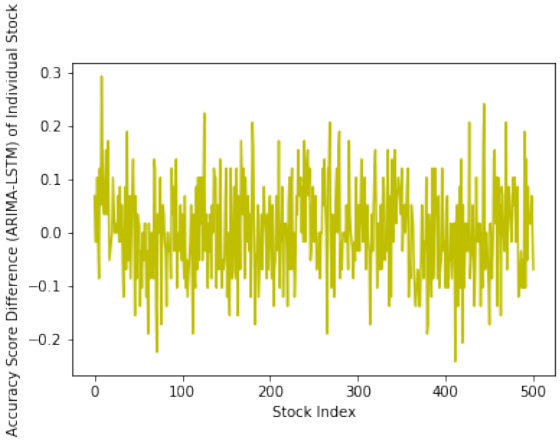

Figure 12: Accuracy comparison (ARIMA-LSTM).

4.3 BERT

The dataset used for BERT fine-tuning training of the classification task was as described in subsection 2.3. We applied a train-test split with the ratio 85:15 and used the random state 17 as seed. The train-test split was stratified to class labels as we faced a big class unbalance as discussed in subsection 2.3.

The initial BERT model was implemented with the help of the Hugging Face transformers package [9]. Tokenization of the texts were performed with the Bert-Tokenizer for the base uncased model. The maximal tokens length of 512 was used. The rational behind this was to use the shortest possible tokens length to increase the model speed, while remaining meaningful. As one can see in Figure 2, most text lengths were around 500 words. Therefore, 512 tokens should cover most of the words in the majority of the texts. For the longer text, one can argue that most of the meaningful information is found in the beginning of the article [10] and we can therefore safely discard later words.

Training loops were performed with a batch size of 8 to not overload the GPU memory as the token sequence was quite large. The optimizer applied was the weighted Adam [11]. The learning parameters used were a learning rate of 10^{-5} and the epsilon 10^{-8} as proposed by the BERT authors. The training was run for 10 epochs and again the random seed 17 was used for the data loader. We performed the training on the Google colab GPU to accelerate learning.

Observing training and validation loss during the training shows increasing values for the validation loss with a decrease in the training loss. Normally, this can be attributed to an over-fitting model. In our case, we are also over-fitting, as we are fitting noise in the training date due to the insufficient labelling performed on the data.

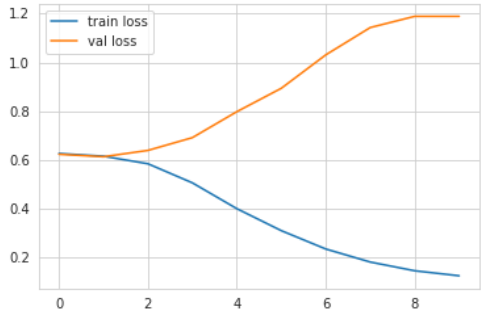

Figure 13: Training and validation loss over the training epochs. While the training loss decreases constantly during training, the validation loss increases.

We can see that the 74.3% overall accuracy achieved by the model is quite high for a ternary classification task. But this is mainly due to the large class unbalance and the model learns to predict 'not relevant' in most of the cases. We, therefore, introduced the measure of per class accuracy, and took the average of the three classes. This weights the accuracy of each class the same and, therefore, reveals a more precise measure. The accuracy per class achieved is with 38.6% extremely poor and only slightly better than assigning random labels. By looking at the trajectories of the different scores over

the training epochs in Figure 14, we can see that there is a decrease in overall accuracy. This is due to the model predicting the dominant class all the time in the first epochs. The per class accuracy remains constant at more or less the level of random guessing. Therefore, we can state that the model is not able to learn a meaningful representation for the semantics of financial press statements. This is highly likely related to the construction of the labels in the training data, since BERT has been proven to be performing extremely well for various semantic analysis tasks.

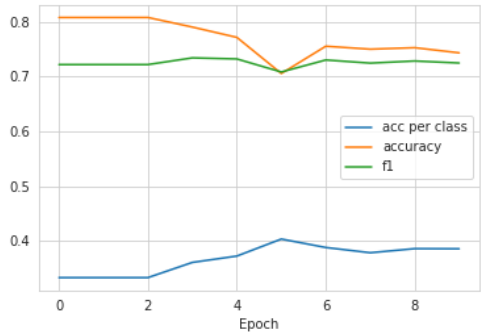

Figure 14: The different performance measures observed over the epochs of the pure BERT model. While the overall accuracy decreases over the training, the per class accuracy increases.

4.4 Adjusted BERT

As seen in the previous section, the pure BERT model was not able to recover semantics from the press releases. One can argue that this is due to insufficient precision when labelling the collected text data. We, therefore, propose a new model. This model uses the predictions made by the Time Series models, ARIMA or LSTM, to determine the label given to an article. We argue that the Time Series model uses periodicity and trend factors to determine the returns of a stock. When looking at financial press statements and their possible impact on the stock, we want to isolate such factors. The labelling is therefore performed with the following formula:

$$\hat{r}_t^s = r_t^s - r_t^{snp} - \bar{r}_t^s$$

$$L_{a_i^s} = \begin{cases} \text{increase} & \text{if } \hat{r}_{t+1}^s > 0.05 \\ \text{not rel.} & \text{if } -0.05 \leq \hat{r}_{t+1}^s \leq 0.05 \\ \text{decrease} & \text{if } \hat{r}_{t+1}^s < -0.05 \end{cases}$$

Where $\bar{r}_t^s$ is the return prediction for stock s and date t given by either the ARIMA or the LSTM model. We used in turn the same set up as in the previous section. The only difference was that due to the limited number of computation time, only a fraction of the returns of year 2020 were predicted with these two models. We included only articles in the data which were dated between 01.10.2020-30.12.2020 and 08.10.2020-30.12.2020 for the ARIMA and LSTM adjusted data respectively. This slightly reduced the number of texts included but looking at the date distribution of the collected articles in Figure 1, will not have a great impact. We furthermore assume that the content of the articles excluded by this do not differ greatly from the included articles.

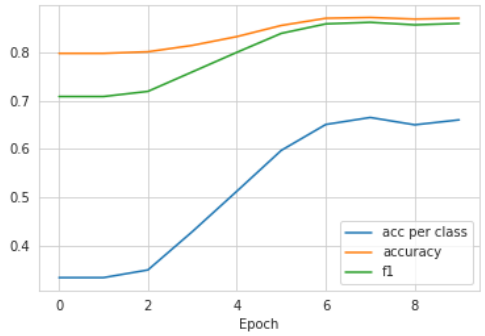

Figure 15: The different performance measures observed over the epochs of the BERT model combined with the ARIMA return predictions. We can see that after an initial phase of poor per class accuracy, the model learns to predict other classes than 'not relevant' correctly and the per class score increases drastically.

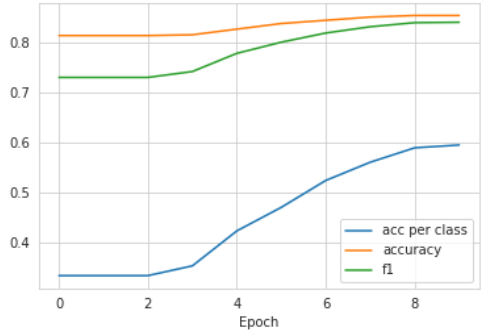

Figure 16: The different performance measures observed over the epochs of the BERT model combined with the LSTM return predictions. Similar to the ARIMA adjusted model, the BERT model is now able to learn semantics and properly predict classes from text data.

Figure 15 and Figure 16 show a very similar behaviour. Due to the isolation of periodic stock movements and general trends by the Time Series models, the BERT model is now able to grasp the semantics in a press statement. The new models, both, achieve a per class accuracy of around $60 - 65\%$ which can be considered solid for a ternary classification problem. Furthermore, the overall accuracy is increasing over the training epochs which shows a promising trend for the possibility of further label engineering.

4.5 Model comparison

Our initial goal was to be able to improve return predictions with the use of NLP. Therefore, we compare the Time Series prediction models with the BERT classification models. In order to do so, we need to bin the results of the Time Series model in the three defined classes. We do so by applying the simple ternary of

return equation explained for the pure BERT model in subsection 2.3. The binning was also performed for the true returns. We then proceed to calculate the F1, accuracy and per class accuracy for all models and summarize results in Table 1.

Model	F1	ACC	pc ACC
ARIMA	**94.6%**	**96.3%**	58.1%
LSTM	93.7%	**96.3%**	58.5%
BERT pure	72.5%	74.3%	38.6%
BERT ARIMA	86.0%	87.1%	**66.0%**
BERT LSTM	84.0%	85.4%	59.5%

Table 1: Summary table of all observed performance measures on all models. The measures for the pure Time Series models are averages over all stocks used. The best performing models were the pure Time Series models on the weighed measures, as constant class prediction is penalized more heavily under the per class accuracy. The overall best performing model is the ARIMA corrected BERT model.

While the Time Series models perform seemingly well for the class prediction task, their performance should be interpreted with care. The task assigned can be solved with an extremely high accuracy of 96.3% by constantly predicting the same class. This is due to the high class unbalance. We therefore strongly focus on the per class accuracy measure that is increased in the adjusted BERT models. We proof that the combination of the BERT model on financial news texts with Time Series prediction models can provide additional use to predict financial returns.

4.6 Error analysis

To better understand wrong predictions made by the model we can observe the labelling performed by the model versus the true classes in Figure 17. We can deduce from the figure that we have no semantic capturing in the pure BERT model and constant class prediction. For the other two models the miss-classifications are only one class off. So the models rarely predict completely opposite semantics which means that the learning performed is indeed related to the text semantics. The models, nevertheless, have a hard time capturing the difference between a relevant or not relevant text, which is understandable due to the fact that labels were engineered and do not forcibly represent the truth. We would argue that with more label engineering and better understanding of the stock markets relations with press statements, one could further increase the model performance.

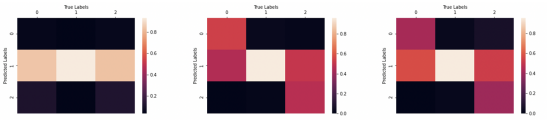

Figure 17: The three heat maps correspond to the pure, ARIMA-, LSTM-adjusted models. The true labels can be found on the x-axis while the predicted label is on the y-axis. Scores for each entry were normalized by the number of samples in each column. While the pure model predicts the 'not relevant' label in all cases, the Time Series adjusted models perform much better in caputring the semantics. Furthermore, we can see that almost all erroneous predictions are only shifted by one field. Therefore, the model almost never predicts a complete opposite sentiment.

5 Conclusion and Future Work

We have successfully scraped financial press statements to generate a new dataset containing text data related to different stock symbols. Although, the size of the dataset is limited and, as we saw in section 4.3, the learning of semantics in such a dataset fails to make generalizations for validation data. Nevertheless, we found a way to remove a part of the noise found in the financial returns and text data to be able to apply a semantics classification model on financial text successfully. We are very well aware of the heavy engineering applied in the dataset and in the process of data labelling. The best performing model, BERT ARIMA adjusted, is performing sufficiently well to proof the concept of using NLP for financial forecasting. There is a lot more work to be done until a model can be considered sufficiently robust to be deployed in the real world. As mentioned, our efforts to isolate periodic and correlated stock movements from the impact of financial news are not advanced and we are convinced that further tuning of the Time Series prediction models will result in a better NLP model. Furthermore, the isolation of press impact on the stock would require more fine grained timescales in the model. The market reacts quickly to news and applying shorter time periods to calculate returns might result in a better model. We also realize that determining the ground truth of the text labels might be infeasible even for domain experts. Stock markets have many influence layers and are genuinely difficult to predict. Further efforts could be taken are the analysis of the predictions with the interpretability package Captum [12]. In a follow up work, the effect of applying unsupervised learning could be used to cluster similar press statements together. The use of such semantic classes that are not directly related to stock returns could be manifold; we imagine a model that uses such classes as an additional input feature.

Overall, we are convinced that the use of Time Series models to facilitate label engineering on financial text data is very promising and can lead to very well performing supervised NLP models. This project has

clearly proven the potential of such even with simple Time Series methods. Additional testing and hyperparameter tuning could reveal the true potential of the proposed method.

6 Learnings from the Project

The collection of useful data is extremely difficult, as we have seen multiple iterations of changes were required on the web crawling part until we have collected a sufficient amount of high quality textual data. While multiple prepared datasets are available for the purpose of education and benchmarking of models, we were interested in solving a real world problem to proof the use of our model in financial decision making. Although this required much more time from our end to construct, understand and curate the dataset, it suits the scope of a course project much better as much more learning result from such a project. We have demonstrated the effect of class unbalance on the performance of classification models and how to address such. We are very well aware of the lack of consistency in the train-test splits, dates selected and datasets compared. These would clearly need to be addressed should this work pass peer reviewing. Nonetheless, we decided against rerunning most of the projects computation due to time constraints. We still decided to explicitly note them for all the models in order to show awareness to the the critical reader.

References

[1] Frank Z. Xing, Erik Cambria, and Roy E. Welsch. Natural language based financial forecasting: a survey. *Artificial Intelligence Review*, 50(1):49–73, Jun 2018. Citation Key: xingNaturalLanguageBased2018.

[2] Wataru Souma, Irena Vodenska, and Hideaki Aoyama. Enhanced news sentiment analysis using deep learning methods. *Journal of Computational Social Science*, 2(1):33–46, Jan 2019.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805 [cs]*, May 2019. arXiv: 1810.04805.

[4] Selenium. https://github.com/SeleniumHQ/selenium, 2018.

[5] Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. *arXiv:1905.05950 [cs]*, Aug 2019. arXiv: 1905.05950.

[6] S. Siami-Namini, N. Tavakoli, and A. Siami Namin. A comparison of arima and lstm in forecasting time series. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1394–1401, 2018.

[7] Ling Hu. Lecture 1: Stationary time series.

[8] Y. Wang, S. Zhu, and C. Li. Research on multistep time series prediction based on lstm. In *2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE)*, pages 1155–1159, 2019.

[9] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and et al. Huggingfaces transformers: State-of-the-art natural language processing. *arXiv:1910.03771 [cs]*, Jul 2020. arXiv: 1910.03771.

[10] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? *arXiv:1905.05583 [cs]*, Feb 2020. arXiv: 1905.05583.

[11] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv:1711.05101 [cs, math]*, Jan 2019. arXiv: 1711.05101.

[12] Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. Captum: A unified and generic model interpretability library for pytorch, 2020.