University of Southampton
Faculty of Physical and Applied Sciences
Electronics and Computer Science

# FinDeBERTa: Deep Bidirectional Transformers for Financial Sentiment Analysis


Author: Roshan Patel

Project Supervisor: Adam Prugel-Bennett
Second Examiner: Geoff Merrett

May 11, 2021

A project report submitted for the award of BSc (Hons) Computer Science

# Acknowledgements

I would like to thank my supervisor Professor Adam Prugel-Bennett for providing me with weekly guidance and never failing to promptly answer any queries I had.

I would like to thank my family for their support during the project.

I would like to thank Southampton University for allowing the use of the Iridis 5 supercomputer cluster for this project.

Finally, I express my gratitude to my second examiner, Professor Geoff Merrett for offering his support throughout the project.

# Statement of Originality

– I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.

– I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.

– I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

– I have acknowledged all sources, and identified any content taken from elsewhere.

– Several Python Libraries were used in this study: pandas, numpy, requests, bs4, scikit-learn, tensorflow, pytorch, alphavantage, redditpushshift, weightsandbiases, seaborn, matplotlib, pyplot, huggingface, simpletransformers.

– I did all the work myself and have not helped anyone else.

– The material in the report is genuine, and I have included all my data/code/designs

– Parts of this work have been adapted from the Semester 1 Progress Report.

# Abstract

Pretrained transformer-based models have recently achieved great results across a range of sequence classification tasks. However, financial polarity classification is a relatively underexplored problem in this domain. This paper introduces the FinDeBERTa model, a pretrained ternary sequence classifier intended for financial polarity analysis. The model uses Microsoft's DeBERTa as its base architecture and leverages additional pretraining on the semantically similar GLUE MNLI benchmark dataset for increased performance. Through additional Bayesian optimisation and mitigations for catastrophic interference, it achieves state-of-the-art results on every measured metric for the Financial Phrase-bank data set, with an f1-score improvement of 5% and 6% for the Base and Large models respectively. The Large model achieves the first perfect f1-score on the subset of sequence labels with 100% label agreement. The second part of the paper uses an LSTM to compare daily closing price predictions using four different feature sets that include sentiment data produced by FinDeBERTa. The results show that the best performing of these feature sets has a 54.7% decrease in RMSE compared to technical analysis alone.

# List of Abbreviations

**AI** = Artificial Intelligence

**BERT** = Bidirectional Encoder Representations from Transformers

**DeBERTa** = Decoding-enhanced BERT with disentangled attention

**ELMo** = Embeddings from Language Models

**FPB** = Financial Phrase-bank

**GLoVE** = Global Vectors for Word Representation

**GLUE** = General Language Understanding Evaluation

**LSTM** = Long Short-Term Memory

**MACD** = Moving Average Convergence Divergence

**MLM** = Masked Language Modelling

**MNLI** = Multi-Genre Natural Language Inference

**NASDAQ** = National Association of Securities Dealers Automated Quotations Exchange

**NLP** = Natural Language Processing

**NSP** = Next Sentence Prediction

**RMSE** = Root Mean Squared Error

**RNN** = Recurrent Neural Network

**RSI =** Relative Strength Index

**SOTA** = State of The Art

**STLR** = Slanted Triangular Learning Rate

**ULMFiT** = Universal Language Model Fine-Tuning

## Table of Contents

# Chapter 1

# Introduction

## 1.1 Motivation

Predicting the daily changes of a stock price is a problem that many traders face. Aside from fundamental and technical indicators, market sentiment plays a significant role in the price of a stock. Being able to accurately gauge market sentiment by considering information from a variety of sources is a challenging problem. In the paper *Attention is all you need* (2017) [1], researchers introduced the transformer model, leveraging the mechanism of attention for deep contextual language representations. This led to a paradigm shift in natural language processing where the vast majority of SOTA performances on benchmark data sets are now from transformer-based models. Despite this, transformers remain relatively unexplored in the field of financial polarity analysis..

## 1.2 Objectives

The project will primarily focus on creating a transformer-based model FinDeBERTa for ternary sequence classification. The purpose of the model will be to classify financial texts as either negative, neutral, or positive. The project will also implement a range of subsidiary models to assess different performances for the same task. The performances of these models will be assessed using the labelled Financial Phrase-bank data set, introduced later in the paper.

The secondary objective of the paper is to explore whether the financial polarity analysis conducted by FinDeBERTa is of benefit when included as part of a feature set in a machine learning model. To achieve this, textual data is collected from a variety of sources over a 750-day period to be classified by FinDeBERTa. A time-

series machine learning regressor will be implemented to assess seven constructed feature sets comprised of a combination of historical price data, technical data, and sentiment data. The goal of the regressor will be to predict the daily closing stock price of the selected target company. The relative performance of these feature sets will be compared to determine the utility of the financial polarity features generated by FinDeBERTa. The target company chosen was NASDAQ:FB (Facebook) for reasons outlined later in the paper.

# Chapter 2

# Background Theory

## 2.1 The Stock Market

The stock market is a publicly accessible exchange where buyers and sellers can purchase securities[1], most frequently in the form of *common shares*. These represent part ownership of a company. The trading management of these publicly listed companies is conducted by an exchange, the two largest of which are the NYSE and the NASDAQ. To be traded publicly, a company must become listed, which is a lengthy process that requires an initial public offering (IPO). In an IPO the private company meets with an investment bank and makes decisions such as the number of shares to distribute and their price, based on performance factors such as the debt to equity ratio, revenue consistency and more [2]. The investment bank takes on the task of underwriting, assuming legal responsibility for the shares and dictating a set of terms required for listing. This includes the exchange's own terms such as minimum number of shareholders to qualify for listing. There are many benefits to becoming publicly listed, such as increased prestige and diversified ownership. However, there are also drawbacks, such as forced public disclosure, pressure for short-term growth, and potentially prioritising market sentiment over intrinsic value.

### 2.1.1 Forces That Move Common Stock

The price at any given moment is ultimately a result of the supply and demand at that point in the market. However, there are 3 main factors that influence this:

[1] Security – A tradeable financial asset, usually in the form of a stock, bond, or option

*Fundamental Factors*, *Technical Factors*, and *Market Sentiment*. ***Fundamental factors*** refer to a combination of two things, which have a variety of metrics to determine them:

1.) The earnings base of the company.

2.) The valuation multiple (expectations about the future).

***Technical factors*** are external conditions that affect the supply and demand of a company's stock price, such as inflation [2]. Finally, ***market sentiment*** refers to the general outlook of investors toward a particular company and can be influenced by a variety of fundamental and technical factors. With the rapid availability of news as well as opinions on forums and social media constantly being shared, market sentiment plays a large role in the volatility of a stock [3].

### 2.1.2 Moving Average Convergence Divergence (MACD)

The MACD is a momentum indicator that highlights the relationship between two moving averages. It is calculated by subtracting the 26 day Exponential Moving Average (EMA) from the 12 day EMA.

The Exponential Moving Average shows how the price of a security changes over a certain period of time. It is primarily chosen over other forms of moving average due to its property of being more reactive to recent price changes. It is calculated as follows:

1. Obtain the first EMA values by taking the Simple Moving Average (SMA).
2. Calculate a smoothing constant defined as $k = \frac{2}{Time\ Period+1}$ (1)
3. $EMA = (Today's\ Closing\ Price \times\ k) + (\ Previous\ EMA \times (1-k))$ (2)

The MACD is then calculated as follows:

$$MACD = 12\ day\ EMA - 26\ day\ EMA \tag{3}$$

From here, a signal line is plotted which is the 9 day EMA of the MACD. The MACD and Signal plots can be used to provide buy and sell signals for traders [4].

When the MACD value crosses above the signal line it suggests the security is bullish[2] and when the MACD crosses below the signal line it suggests the security is bearish[3].

### 2.1.3 Relative Strength Index (RSI)

The RSI is a momentum indicator primarily used for the evaluation of overbought or oversold securities. It is measured between 0 and 100, where common interpretation is that a value below 30 indicates an oversold condition and over 70 indicates that a stock is overbought. A trader will usually want to purchase a stock when it is oversold, as it suggests the stock is undervalued, and conversely may want to avoid a stock that is overbought, as it may soon experience a corrective pullback in price [5].

The RSI is calculated by first calculating the Relative Strength (RS):

$$RS = \frac{Average\ Positive\ Change\ During\ Time\ Period}{Average\ Negative\ Change\ During\ Time\ Period} \tag{4}$$

The difference in daily closing price changes are determined and a Simple Moving Average of these changes are taken over a time period, most commonly 14 days. From here, the average gain is determined, which is equivalent to the mean of the upward changes. Likewise, the average loss is determined, which his equivalent to the mean of the downward changes. The Relative Strength is then normalised to a value between 0 and 100 to determine the Relative Strength Index as follows:

(5)

[2] Bullish – characterised by rising share prices
[3] Bearish – characterised by falling share prices

$$RSI = 100 - \frac{100}{1 + RS}$$

## 2.2 Artificial Intelligence (AI)

The ability of machines to perform Artificial Intelligence can be broken down into 3 categories [5]:

1. Artificial Narrow Intelligence – machines that operate within a pre-defined, pre-determined range, even if it has the appearance of being much more sophisticated.
2. Artificial General Intelligence – machines that exhibit human intelligence, successfully performing any intellectual task that a human being can.
3. Artificial Super Intelligence - *"any intellect that greatly exceeds the cognitive performance of humans in virtually all domains of interest" – Prof. Nick Bostrom, University of Oxford* [5] .

Currently humans have created machines that can perform at the Narrow level of AI. Artificial Intelligence can be partially broken down into the following sub-fields that are pertinent to the content discussed in this paper [6]:

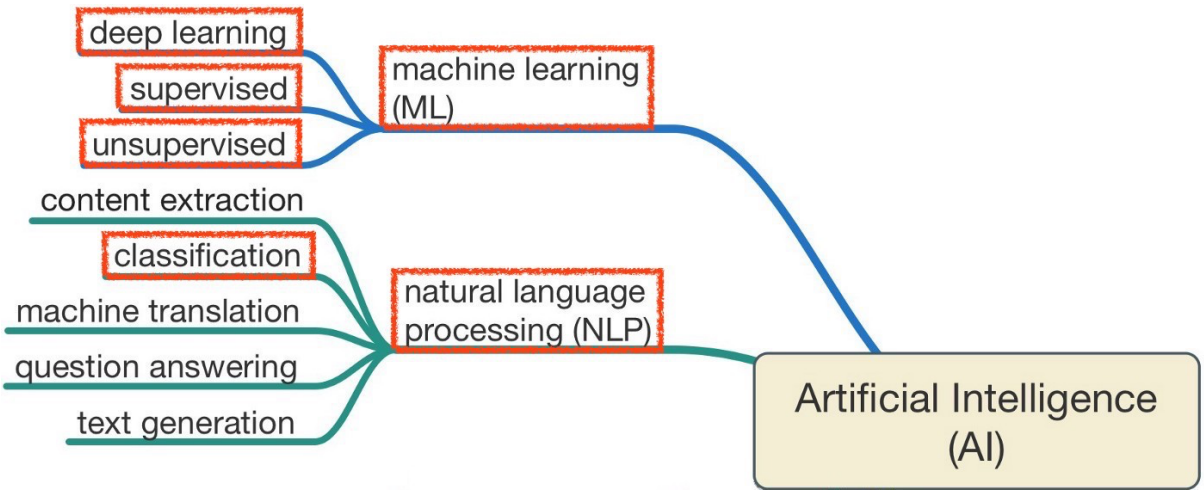



*Figure 1: Partial Sub-fields of AI. Image adapted from:* [6].

They are defined as follows [7]:

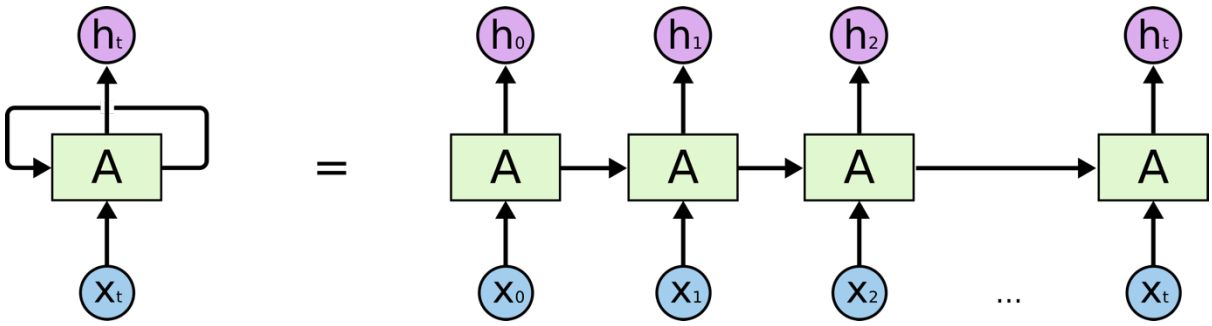
- Machine Learning – algorithms that improve through experience and by the use of data.
- Supervised Learning – algorithms that match an input to an output, where the true output is labelled in the training data allowing the algorithm to evaluate itself on the training data.
- Unsupervised Learning – algorithms that match an input to an output, where the true value for the output is unknown and so the algorithm tries to extract patterns independently.
- Deep Learning – inspired by the biological structure of the human brain to make use of artificial neural networks.
- Natural Language Processing – branch of artificial intelligence that aims to help computers understand, interpret and manipulate human language.
- Sequence Classification – the task of dividing sequences of inputs into predefined classes.

All of these elements of artificial intelligence contribute to the ternary sequence classification task explored in this paper; **correctly classifying financial text as either negative, neutral or positive**.

## 2.3 Long Short-Term Memory (LSTM)

The LSTM is a form of Recurrent Neural Network (RNN). RNN's were developed to resolve the issue of Artificial Neural Networks not being able to leverage previous input values to inform later decisions. This was particularly damaging for time-series tasks such a stock price forecasting, where techniques such as trend analysis and future prediction rely on some form of memory mechanism for past data. RNN's. achieved this by introducing loops.

*Figure 2: A RNN Cell. Image reproduced from:* [8].




(a) RNN Cell (b) Artificial Neural Network representation of RNN Cell

Though the RNN allowed connecting previous information to the present task, it still suffered when it came to using information from states that were too far back due to vanishing gradients and exploding gradients [8]. LSTMs were explicitly created to avoid this long-term dependency problem. This was achieved by adapting the RNN's repeating module from a single tanh neural network layer to a four layered neural network .

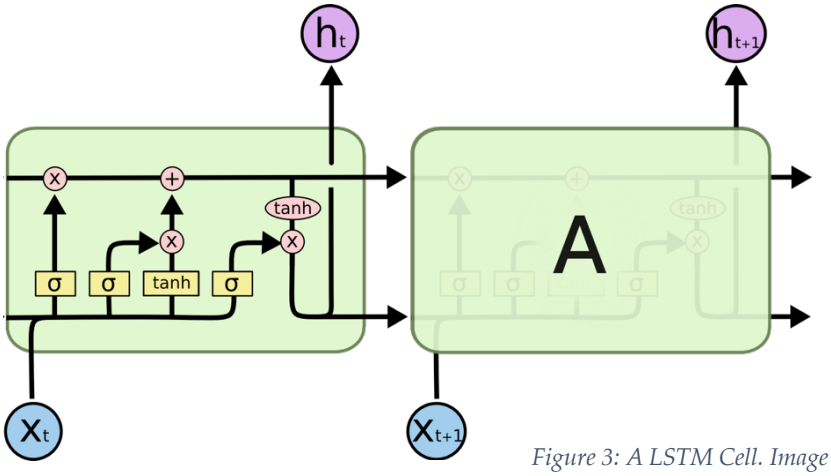



*Figure 3: A LSTM Cell. Image reproduced from:* [8].

In the LSTM cell shown in figure 3, there are three layered gates that control the cell state. These gates are: the forget ($f_t$) gate, the input gate ($i_t$) and the output gate ($o_t$). The forget gate is a sigmoid layer that decides whether a piece of information that has been stored is still relevant via a sigmoid activation function ($\sigma$). This function

outputs a value of either 0 (signal the gate to block everything) or 1 (signal the gate to let everything through). The input gate is also a sigmoid layer that decides what new information is significant enough to be stored in the cell state. The resultant state will be passed to the output gate and filtered with a sigmoid layer to decide the output contents via a *tanh* function, by setting the values to either 1 or -1 to dictate whether the information is output or not.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{6}$$

$$i_t = \sigma(w_i[h_{t-1}, X_t] + b_i) \tag{7}$$

$$f_t = \sigma(w_f[h_{t-1}, X_t] + b_f) \tag{8}$$

$$o_t = \sigma(w_o[h_{t-1}, X_t] + b_o) \tag{9}$$

Where:

| | |
|---|---|
| $w_x$ | Neuron Gate Weight |
| $h_{t-1}$ | Previous Cell Output |
| $X_t$ | Current Input |
| $b_x$ | Gate Bias |

The Neuron Gate Weight ($w_x$) denotes the strength of the connection and the Gate Bias ($b_x$) shifts the activation function to fit the data.

## 2.4 Sequence Classification with Pre-Trained Language Models

This section will provide a background of the neural architectures implemented in this paper and aim to provide a narrative of the progression of NLP models in recent years.

### 2.4.1 Global Vectors for Word Representation (GLoVE)

In order for words to be interpreted by machine learning models, they need a form of numeric representation for models to use in calculations. This was achieved by using a vector to represent words which simultaneously enabled capturing semantic and syntactic relationships between them in latent space. For example, the semantic relationship between the words 'London' and 'England' or the syntactic equivalence between pairs like ('had' / 'has') and ('was' / 'is').

Researchers realised using embeddings pretrained on incredibly large data sets was an effective way to capture these relationships, which inspired GLoVE. GLoVE is an unsupervised learning algorithm for generating vector word representations trained using an LSTM. The algorithm uses the frequency and relative position of the words in a large corpus to measure the semantic similarity between corresponding words. This is achieved by using a corpus to form a global word-word cooccurrence matrix. The matrix is factorised to yield a lower-dimensional matrix by normalising the counts and log-smoothing them. The word embeddings are context independent and as such, there is just one word vector representation for each word and the model is not robust to polysemy[4]. The rationale is that the probability ratios have encoded meaning, demonstrated by the example in Table 1, recreated from the original paper [9]:

*Table 1: Word probability ratios used in GLoVE.*

| Probability and Ratio | k = solid | k = gas | k = water | k = fashion |
|---|---|---|---|---|
| P(k \| ice) | $1.9 \times 10^{-4}$ | $6.6 \times 10^{-5}$ | $3.0 \times 10^{-3}$ | $1.7 \times 10^{-5}$ |
| P(k \| steam) | $2.2 \times 10^{-5}$ | $7.8 \times 10^{-4}$ | $2.2 \times 10^{-3}$ | $1.8 \times 10^{-5}$ |
| P(k \| ice)/P(k \| steam) | 8.9 | $8.5 \times 10^{-2}$ | 1.36 | 0.96 |

[4] Polysemy – the coexistence of many possible meanings for a word or phrase

In Table 1, the values plotted indicate the probability the next word is either 'solid', 'gas', 'water' or 'fashion'. In the final row, the ratio of P(k|ice) divided by P(k|steam) is shown. Observing the results it is evident that when the next word is closer to the numerator (ice) the ratio is >1 and when the next word is closer to the denominator (steam) the ratio is <1. The probability increases as the ratio gets further from 1. For completely dissimilar words, the value will be close to 1 as both probabilities should both be very low, evidenced by the ratio for 'fashion'.

### 2.4.2 Embeddings from Language Models (ELMo)

ELMo builds upon GLoVE by using a contextualised embeddings instead of a fixed embedding for each word. This is achieved by using a bi-directional LSTM to analyse an entire sequence of words before assigning each one an embedding. This contextual understanding is gained by training ELMo on a task called language modelling, whereby the objective is to predict the next word in a sequence of words. The implementation of the bi-directional LSTM was integral to the success of ELMo as it enabled the model to be able to observe the previous and next words as well as train by analysing the sequence from **both directions** [9]. Once the forward and backward language models are trained, ELMo concatenates the hidden layer weights together into a single embedding.

### 2.4.3 Universal Language Model Fine-Tuning (ULMFiT)

The goal of ULMFiT was to build one model that could serve any classification task. It was primarily achieved using transfer learning[5] and is regarded as one of the first successful implementations of transfer learning in natural language processing. As the name suggests, ULMFiT is initially pre-trained on a **language model**ling task utilising a large data set. Adaptation to new tasks can be achieved by further fine-

[5] Transfer learning – A machine learning method where a model developed for a task is reused as the starting point for another task

tuning. The ULMFiT paper highlighted two novel techniques that improved the transfer learning process: Slanted Triangular Learning Rates and Discriminative Fine-tuning [9]. These are leveraged in the implementation section where they are explained in more detail.

### 2.4.4 Bidirectional Encoder Representations from Transformers (BERT)

In 2017, Google Research published a paper titled *Attention Is All You Need* [1], in which they proposed a new architecture known as the Transformer, for the purpose of Neural Machine Translation (NMT). The Transformer took inspiration from the observation that the best performing models at the time utilised an encoder and a decoder, commonly connected through a mechanism known as attention. Transformers capitalise on this and implement a connected encoder and decoder while dispensing of recurrence entirely. The result led to a paradigm shift in natural language processing and the vast majority of SOTA performances on NLP tasks at the time of writing this paper are from transformer based models.

The transformer is composed of an encoder and a decoder, shown in Figure 4 [10]:

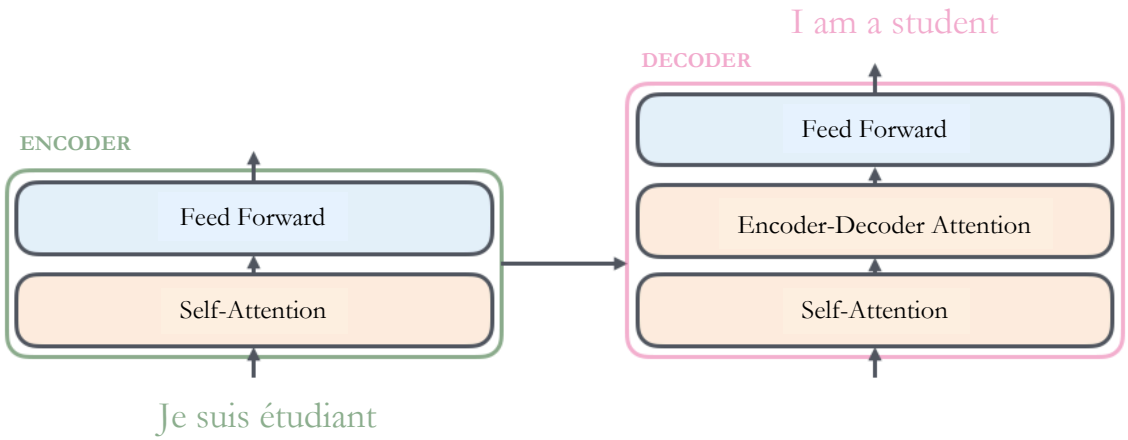



*Figure 4: Transformer architecture. Image reproduced & adapted from* [10].

Unlike LSTMs, Transformers do not need sequential data to be processed in order, as the entire sequence is encoded at once. Additionally, utilisation of attention mechanisms addresses the previously discussed problem of vanishing and exploding gradients in RNN architectures [9]. This is because unlike RNN architectures, the attention layer can access all states and weight them using a learned relevancy with respect to the current token being observed. A self-attention mechanism enables identification of context for any position in the input sequence, allowing Transformers more training parallelisation compared to LSTMs. This drastically reduces training time, which also enables training on larger data sets than before [11].

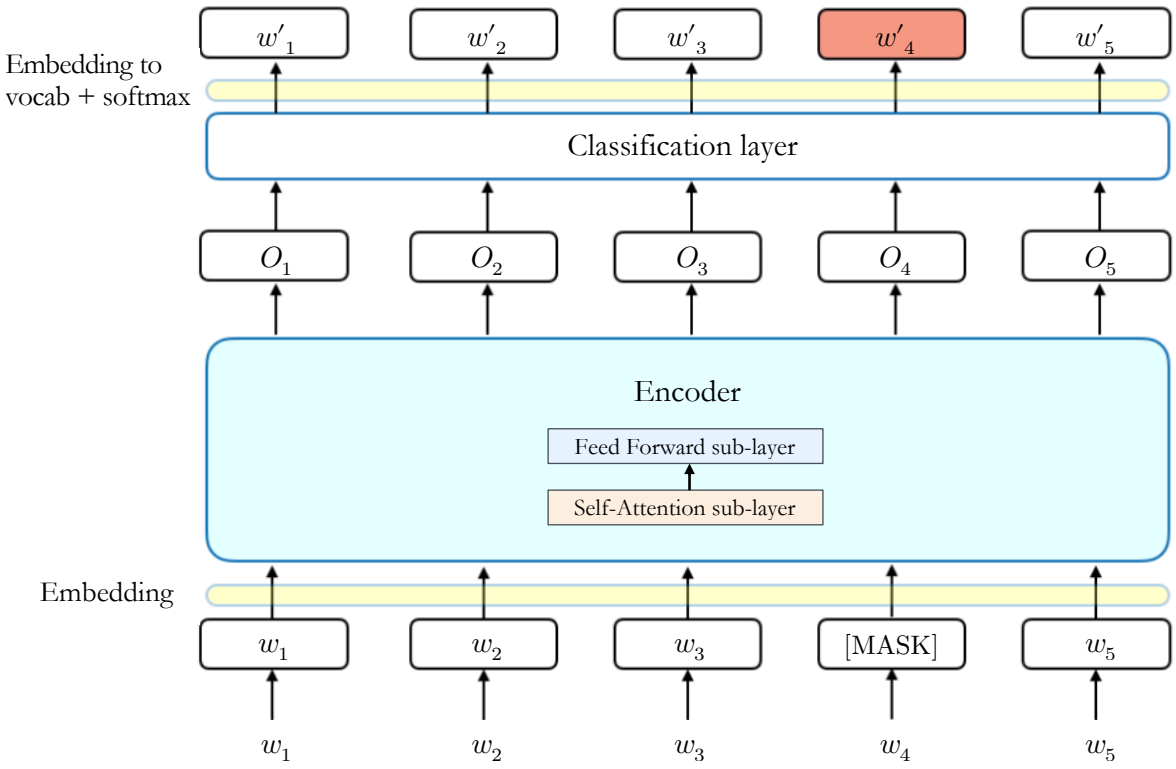



*Figure 5 – BERT architecture. Image reproduced and adapted from:* [11].

The BERT model **utilises the encoder alone**, by stacking multiple layers of them. Each encoder can be broken down into the feed-forward and self-attention sub-layers. In the embedding layer, each word in the input sequence is represented by a vector, which is the sum of the content embedding and the positional embedding. These vector embeddings are then passed to the encoder. The self-attention layer is

where the encoder observes surrounding words in the input sequence to generate a set of **contextually aware** output encodings. The outputs are fed into the feed forward neural network which further processes the encodings.

In a self-attention unit, three weight matrices are learned: query weights $W_Q$, key weights $W_K$, and value weights $W_V$. These matrices are multiplied by each input word embedding to produce the query vectors ($Q$), key vectors ($K$) and value vectors ($V$). The key and query vectors are used to calculate the attention weights. These attention weights are divided by the square root of the dimension of the key vectors $\sqrt{d_k}$ to stabilise the gradients during training [1][12]. They are then normalised by use of a softmax function so that the sum of the weights is equal to 1 [13]. The attention calculation can be expressed as one matrix operation:

$$Self\ Attention\ (Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \qquad (10)$$

Where:

| | |
|---|---|
| $Q$ | The query vectors |
| $K^T$ | The transpose of the key vectors |
| $V$ | The value vectors |
| $d_k$ | Dimension of the key vectors (input sequence length) |

This form of attention is called scaled dot-product attention. One set of ($W_Q$, $W_K$, $W_V$) matrices is referred to as one *attention head*. In practice, each encoder layer in a transformer model has several attention heads. This is advantageous as each head can learn a different form of 'contextual relevance'. In fact, research has empirically demonstrated that with multiple attention heads there are relevance relations that are recognisable to humans such as linking words to the objects they describe [12].

The output of the encoder is fed into a classification layer where during pretraining, the model is trained on two separate tasks: Masked Language Modelling (MLM) and Next Sentence Prediction (NSP).

The **Masked Language Model** task involves replacing 15% of the words at the input stage with a [MASK] token (see Figure 5). The model then uses the context of the words available to it to predict the masked words. The **Next Sentence Prediction** task involves the model receiving two sentences and the task is to predict whether or not the second sentence follows the first sentence in the original input text. The goal of the pretraining stage of BERT is to minimise the combined loss function of the Masked Language Model and Next Sentence Prediction tasks [13].

For the fine tuning stage the model is trained for a different downstream task via supervised learning. the classification layer for pretraining is replaced by a new classification layer whose weights are instantiated randomly. This classification layer and the other encoder layers are then 'fine-tuned' with a low learning rate in order to learn the specific task.

### 2.4.5 Decoding-enhanced BERT with Disentangled Attention (DeBERTa)

DeBERTa is the model that forms the foundations of the FinDeBERTa model introduced in this paper. DeBERTa introduces two core novel improvements over similar encoder based models: a disentangled attention mechanism, and an enhanced mask decoder.

The **disentangled attention mechanism** is implemented by using two individual vectors to encode the content and positional embeddings separately, where BERT uses one vector formed by summing the embeddings. This change was motivated by the observation that the attention weight of a word pair should depend both on

the contents of the words and their relative positions to each other [14]. For example, the word dependency between the words "*mathematical*" and "*proof*" is stronger when they appear beside each other. By separating the content and positional word embeddings into two separate vectors the attention weights can be improved by computing disentangled matrices based on content and relative position.

The **enhanced mask decoder** is an enhancement to the Masked Language Model pretraining task. It consists of incorporating absolute positional word embeddings just before the softmax layer, where the model decodes the masked words. This is because many syntactical nuances depend on absolute positional information of words. This is illustrated by the following example from the DeBERTa paper [15]:

Consider the sentence "a new *store* opened beside the new *mall*", where both the italicised words are masked in the MLM task. The contexts of the words are similar, however, they have differing syntactic roles in the sentence (i.e. *store* is the subject of the sentence). These syntactic differences are largely dependent on absolute word positions within the sentence making it beneficial to capture this additional information.

## 2.5 Sentiment Analysis in the Financial Domain

Sentiment analysis is the mining of text to extract both objective and subjective information, usually to create meaningful insights. There are two main challenges with sentiment analysis in the financial domain: the first is correctly mining the intended sentiment from the text, and the second is having a representative enough data set so that the sentiment being captured is truly representative of the problem itself [16]. Financial sentiment is predominantly done to use as a numeric representation in a trading strategy with the hope that that strategy will yield a

positive Alpha[6] ($\alpha$). Research in financial sentiment analysis commonly focuses on either **(1) correctly identifying financial polarity** or **(2) using polarity indicators successfully in a trading model**. This paper explores both.

**(1)** Many of the early works of deep learning in financial polarity analysis involve the application of an LSTM using a language model. These were mostly built with the intention of aiding traders in decision making [17]. They were shown to be better performing than traditional machine learning approaches, a theme which is prevalent across many papers [18]. Here, pre-training is generally conducted on a labelled data set via supervised learning, which has recently been empirically shown to be a limited approach when compared to pretraining transformers as a self-supervised task [1]. Loughran & McDonald (2015) [19] created a domain specific dictionary of financial based terms and leveraged this to create a more targeted Language Model, offering minor improvements over similar models at the time. LPS introduced a model that focused on the phrasal structure of sentences to better understand contextual and semantic orientations [20]. Here, they also released the Financial Phrase-bank data set for financial based ternary sequence classification tasks. The Financial Phrase-bank data set is more thoroughly explained in the 'Implementation' section, where it is utilised as a model evaluation metric. HSC (2018) adopted a unique approach based on hierarchical association rules to classify sentiment, and through reimplementation of Loughran & McDonald and LPS, proved using the Financial Phrase-bank data set that it was superior to Loughran & Mcdonald and similar in performance to LPS [21]. FinSSLx (2018) combined an LSTM and text simplification methods using clausal and phrasal based simplifications to classify financial polarity [22]. These novel simplification methods saw FinSSLx set a SOTA result on the Financial Phrase Bank at the time. This SOTA performance was beaten

[6] Alpha – The performance of investment against a market index or benchmark return

by FinBERT [23] (2019), which adopted a transformer centric approach by fine-tuning the pre-trained language model BERT [13].

**(2)** This paper selects NASDAQ:FB to generate polarity indicators for the implemented machine learning model. Facebook was chosen primarily due to the abundance of news and social media data concerning the company, as it has frequently been the focus of media attention in recent years. Finding competitive implementable literature reporting success with sentiment polarity in a financial model is a difficult task. This is partly because of the financial incentive of keeping successful methods discrete. Moreover, an architecture and its feature set are very problem dependent, varying greatly between companies or industries being analysed. For this reason, only papers and blogs that involved companies in the technology sector with reproducible code were examined. Across these papers, it was observed that deep time-series models were generally the most favourable as they were more suited to capturing temporal dependencies. Traditional machine learning methods were also observed, however, those that were successful often incorporated time series information by adding them as additional features [24][17].

# Chapter 3

# Implementation

This section will detail the creation of the financial sentiment analyser FinDeBERTa and subsequently its utilisation in a trading strategy outlined in Figure 6.

## 3.1 Proposed Methodology

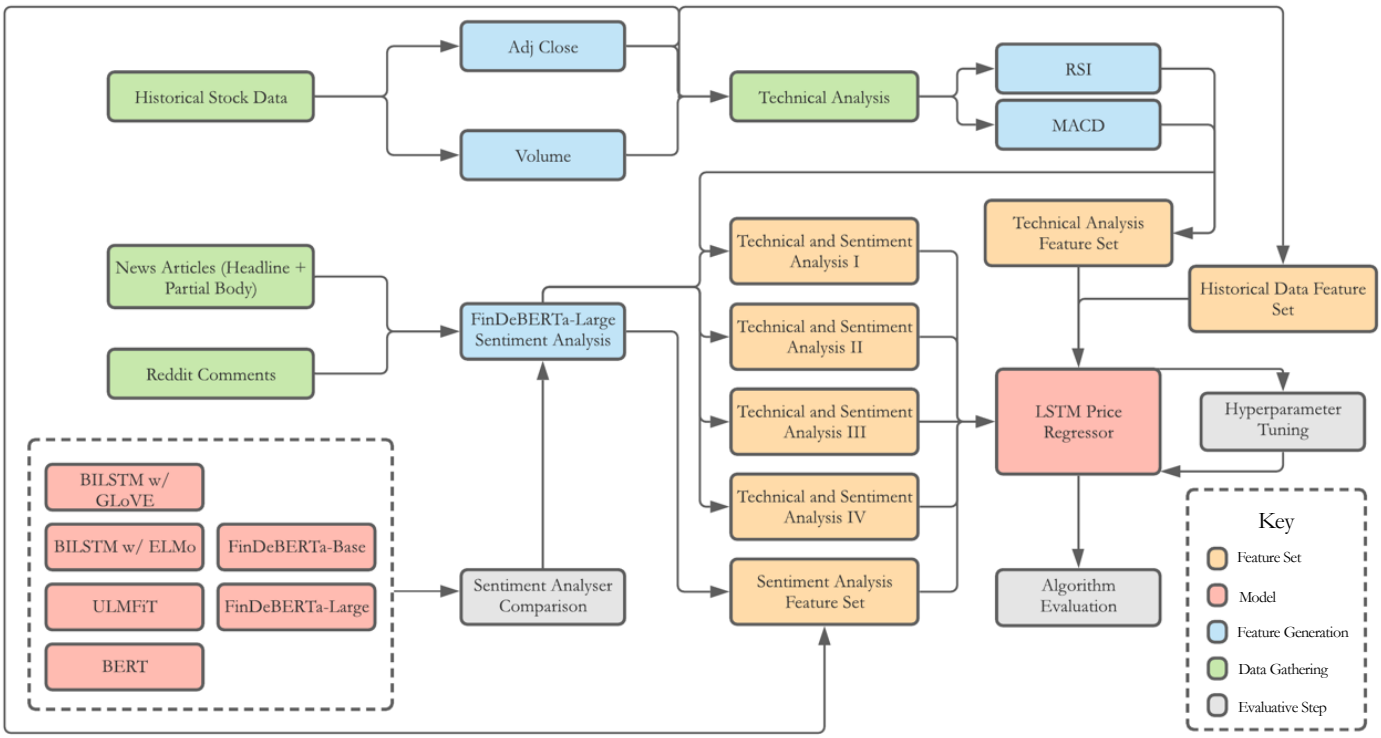



*Figure 6: Proposed methodology flowchart. Image created using Lucidchart.*

## 3.2 FinDeBERTa

This subsection details the full implementation of the FinDeBERTa model including the approach to additional training for downstream classification tasks in the financial domain. It also covers the Bayesian hyperparameter tuning used for training optimisation and the empirical findings within the context of current literature. Consequently, this subsection aims to answer the following **research questions**:

**RQ1.** What is FinDeBERTa's performance on sequence classification tasks in the financial domain when compared to other implemented models?

**RQ2.** How does FinDeBERTa compare to SOTA methods in multiclass financial sentiment sequence classification?

**RQ3.** What is the performance improvement of FinDeBERTa after Bayesian hyperparameter optimisation?

### 3.2.1 The Financial Phrase-bank

The Financial Phrase-bank [20] is an annotated corpus of 4840 labelled financial phrases. The phrases are from English news articles concerning companies listed on OMX Helsinki. This dataset was chosen partly due the text sequences being semantically similar to the target news articles that will be analysed by the stacked encoder model. Moreover, this dataset was selected for its high reliability, as sixteen members, comprising researchers and postgraduate students in financial programs, were responsible for labelling the data. The data set is split into four different groups differentiated by the % agreement on the sequence label: *negative*, *neutral,* or *positive*. The data set is divided into an 80-10-10 train-test-validation split, and all experimental analyses is conducted with 10-fold cross validation.

*Table 2: Financial Phrase-bank data set overview.*

| Data set | Number of Sentences (% Negative, % Neutral, % Positive) |
|---|---|
| 100% agreement | 2259 (25.2, 61.4, 13.4) |
| >75% agreement | 3448 (25.7, 62.1, 12.2) |
| >66% agreement | 4211 (27.7, 60.1, 12.2) |
| >50% agreement | 4840 (28.2, 59.3, 12.5) |

### 3.2.2 Evaluation Metrics

The evaluation metrics used are: Accuracy, Macro-averaged F1 score, Weighted F1 score and Confusion Matrix:

The Confusion Matrix shows the predicted and true labels across all three classes. It can be used to derive the following definitions that have been modified to correspond with this **ternary sequence classification** problem [25]:

- True Positive (TP) = Both the model's prediction and the true class are equal to the class being observed
- True Negative (TN) = Both the model's prediction and the true class are equal to either of the classes **not** being observed
- False Positive (FP) = The model predicted the class being observed, but the true class is **either** of the other classes
- False Negative (FN) = The model predicted **either** of the other classes, but the true class is the class being observed

**Accuracy:**

$$accuracy = \frac{no.\,of\ correct\ predictions}{total\ no.\,of\ predictions} \tag{11}$$

**Macro-averaged F1:**

$$precision = \frac{true\ positives}{true\ positives + false\ positives} \quad recall = \frac{true\ positives}{true\ positives + false\ negatives} \tag{12}$$

$$f1 = 2 \times \frac{precision\ \times\ recall}{precision + recall} \tag{13}$$

$$Macro\ f1 = \frac{f1_{negative} + f1_{neutral} + f1_{positive}}{3} \tag{14}$$

The macro-averaged f1 score involves calculating the f1 score across each class: negative, neutral, and positive, individually. The scores are then summed and divided by 3. Through this, an unweighted (macro) f1 score is produced whereby each class is considered equal in size, meaning class imbalance is not accounted for. Although the Financial Phrase-bank data set is imbalanced, this is likely the most useful f1 metric, as the class distribution of negative, neutral, and positive sentiment in forums and news articles will vary depending on current events at the time.

**Weighted F1:**

$$Weighted\ f1 = f1_{negative} \times 0.282 + f1_{neutral} \times 0.593 + f1_{positive} \times 0.125 \tag{15}$$

The weighted f1 score is calculated similarly to the macro-averaged f1 score. The difference being instead of dividing the summed f1 score by the number of classes, each score is multiplied by the class' ratio in the data set. The class ratios for the full data set of >50% agreement have been inserted into the equation above to illustrate this; 28.2% *negative*, 59.3% *neutral*, 12.5% *positive*.

### 3.2.3 Baseline Methods

In order to adequately assess FinDeBERTa's performance on the Financial Phrase-bank data set, baselines are gathered from four other implementations:

**Bi-LSTM with GLoVE.**

For this implementation, a bi-directional LSTM classifier is implemented with a hidden size of 64 and hidden state size of 128. The last hidden state is mapped to a vector with 3 values by a fully connected feed-forward layer. These 3 values

represent the probability of each of the 3 class labels. A dropout layer is included with a dropout probability of 0.3 to reduce overfitting. The optimiser used is the Adam optimiser. The learning rate is set to 3e-5 with a warmup ratio of 0.1. The model was run for 50 epochs with early stopping exercised when there was no improvement on the validation loss for 3 consecutive iterations. The embeddings used for this implementation are the GLoVe embeddings.

**Bi-LSTM with ELMo.**

This implementation is almost identical to the one above, the only difference being ELMo embeddings are used instead of GLoVe embeddings. This is done to observe any advantage of contextualised word representations, as with ELMo the surrounding words influence the representation of the word.

**ULMFit.**

The pretrained ULMFit language model is fine-tuned for classification with the Financial Phrase-bank data set using the *Fast.ai* library. A fully connected layer is added to the output of ULMFit. Training is conducted for 15 epochs with a learning rate of 4e-3 and an Adam optimiser. The early stopping requirement is the same as the previous two implementations.

**BERT-Base-Cased.**

The BERT-base-cased model is a 12-layer, 768-hidden, 12 head, 109M parameter model pretrained on cased English text. It is implemented using the *Huggingface* library and fine-tuned over 2 epochs with a learning rate of 2e-5 and warmup ratio of 0.06 as recommended by Google Research. The Adam optimiser is also used for this implementation.

### 3.2.4 Implementing FinDeBERTa

Two FinDeBERTa models were created: FinDeBERTa-Base with 12 layers and FinDeBERTa-Large with 24 layers. The full specification of both models is available on the FinDeBERTa Github repository.

Firstly, publicly available SOTA models were assessed based on two factors:

1. Their performance on the GLUE MNLI benchmark task.
2. Their performance by validation loss after further training on the Financial Phrase-bank corpus.

The General Language Understanding Evaluation (GLUE) benchmark is a collection of resources for training, evaluating and analysing natural language understanding systems. Performance on the Multi-Genre Natural Language Inference (MNLI) benchmark was chosen as an evaluation metric, as the task is similar to the Financial Phrase-bank task. The similarity is that both objectives are to correctly sort the text sequences into 3 distinct classes. Specifically for MNLI, "given a premise sentence and a hypothesis sentence, the task is to predict whether the premise entails the hypothesis (entailment), contradicts the hypothesis (contradiction), or neither (neutral)" [26].

After reviewing the scores of various models, 3 were selected and further trained on the Financial Phrase-bank corpus [27]. The results are as follows:

*Table 3: Base model performance comparison on GLUE MNLI and Financial Phrase-bank.*

| Base Model | MNLI Score (acc) | FPB Val Loss (cross-entropy) |
|---|---|---|
| RoBERTa-Base/Large | 87.6/90.2 | 0.408/0.389 |
| ELECTRA-Base/Large | 88.8/90.9 | **0.400**/0.381 |
| DeBERTa-Base/Large | **88.8**/**91.3** | 0.406/**0.377** |

These preliminary results show that DeBERTa is likely to perform marginally better than the other models and as such it was selected as a base architecture.

Due to the aforementioned similarities between the MNLI task and the Financial Phrase-bank task, it was hypothesised that additional training on the MNLI set could be beneficial to the overall performance of FinDeBERTa. Specifically, the large corpus size of 393,000 train and 20,000 test samples, coupled with the high data reliability implied by its use as a classification benchmark, suggested that further training on the data set could improve the model's generalisation to ternary sequence classification problems [28]. The model submitted by Microsoft for assessment on the MNLI test set was accessed and implemented through the *Huggingface* library. The results after further training on the Financial Phrase-bank data set are as follows:

*Table 4: Financial Phrase-bank results using DeBERTa further trained on MNLI task.*

| Base Model | FPB Val Loss (cross-entropy) |
|---|---|
| DeBERTa-MNLI-Base/Large | 0.381/0.354 |

The results support the hypothesis, as evidenced by the Base and Large architectures' cross-entropy loss improving by 0.045 and 0.033, respectively.

#### 3.2.4.1 Mitigation of Catastrophic Interference

Catastrophic Interference is an observed phenomenon in which neural networks have the tendency to completely and abruptly forget previously learned information during training. FinDeBERTa implements two previously mentioned strategies discovered by the authors of the ULMFiT paper that are empirically shown to mitigate the occurrence of the phenomenon in transfer learning models.

The first is **Slanted Triangular Learning Rate (STLR)**, which is a learning rate schedule that initially linearly increases the learning rate, after which the learning rate is linearly decayed. This method of learning is demonstrated in the ULMFiT paper to have a comparably lower incidence of Catastrophic Interference than constant learning rates, partially evidenced by a smoother decline in cross-entropy loss [29].

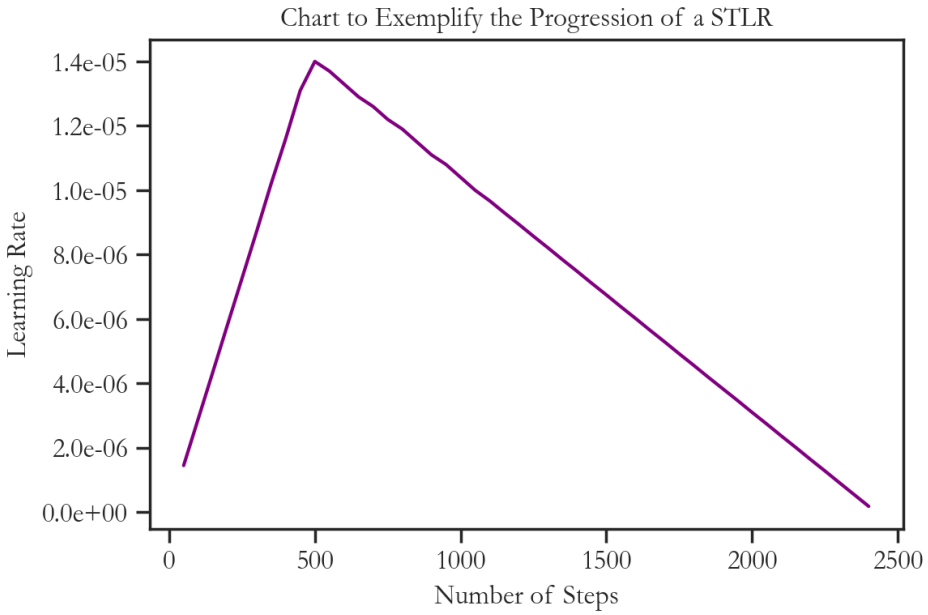



*Figure 7: Slanted Triangular Learning Rate (STLR) plotted from Bayesian Optimisation run data.*

The second is **Discriminative Fine-tuning**, whereby instead of tuning the layers with the same learning rate, the stacked encoder layers are tuned individually. To reduce training time, layers were grouped into 4 groups determined by $n_{layers\ per\ group} = \frac{n_{layers}}{4}$. This meant that the FinDeBERTa-Base architecture's 12 layers were split into 4 groups of 3 encoder layers and the FinDeBERTa-Large architecture was split into 4 groups of 6 encoder layers.

The Slanted Triangular Learning Rate was used in every training iteration of the model, however, the Discriminative Fine-tuning values were discovered empirically in the next subsection, where several hyperparameters are fine-tuned to provide an optimised version of the model.

### 3.2.5 Hyperparameter tuning with Bayesian Optimisation

The objective function $f(x)$ of the optimisation problem for FinDeBERTa is a black box function. As such, there is no analytical expression for $f$ and its derivatives are unknown. Evaluation of the function $f$ is conducted by sampling various hyperparameter values $x$. Bayesian optimisation was deemed to be the most useful method of hyperparameter tuning here. This is due to its incorporation of the *acquisition function* to propose promising sampling points in the search space, thereby markedly reducing search time compared to Grid and Random search [30]. The Base model was used in the search, and architecturally dependent vales such as layer based learning rates were extrapolated for the Large model. The search was run using Southampton University's IRIDIS 5 supercomputer cluster, where 4 independent agents were deployed to run Bayesian searches simultaneously.

Initially, a Bayesian search was done with the learning rate and number of epochs to narrow down the subsequent search space, as these parameters are of primary importance to model performance in related literature [31]. A random sample of the results used to inform the subsequent search is shown below:

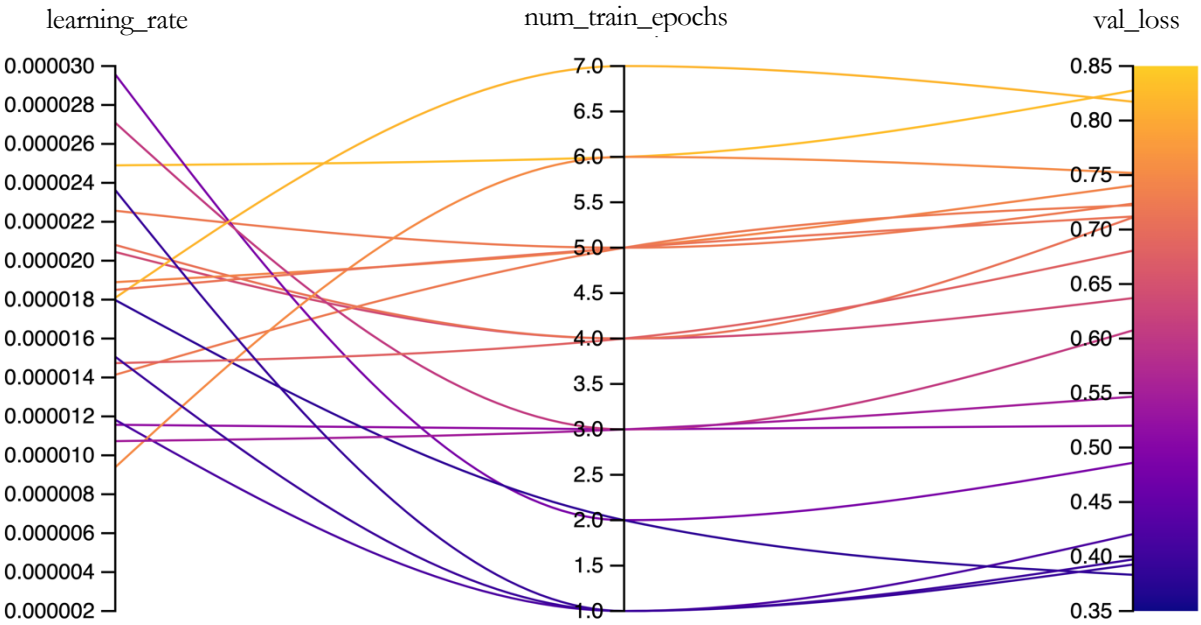



*Figure 8: Preliminary Bayesian Optimisation search results.*

The val_loss label refers to the cross entropy loss on the validation data set. This initial search indicated that the best results generally had a learning rate between 1e-6 and 3e-5 and ran for 1 to 3 epochs. The 'num_train_epochs' parameter was restricted in the subsequent search space. The parameters and their ranges in the subsequent search are outlined below.

```
"parameters": {
    "num_train_epochs": {"min": 1, "max": 3},
    "params_classifier.dense.weight": {"min": 0, "max": 0.5e-3},
    "params_classifier.dense.bias": {"min": 0, "max": 1e-3},
    "params_classifier.out_proj.weight": {"min": 0, "max": 1e-3},
    "params_classifier.out_proj.bias": {"min": 0, "max": 0.5e-3},
    "learning_rate": {"min": 0, "max": 3e-5},
    "warmup_ratio": {"min": 0, "max": 0.3},
    "weight_decay": {"min": 0, "max": 0.3},}
```

*Figure 9: Search space of primary Bayesian Optimisation search.*

The parameters beginning with 'params_classifier' are scalar weights and bias values associated with the fully-connected classification layer. The warmup_ratio refers to the ratio of the total number of steps that is used to progressively increase the learning rate to reduce the primacy effect of the early training examples, as well as form part of the Slanted Triangular Learning Rate. The weight_decay parameter is an L2 Regularisation technique applied to the weights of the fully-connected classification layer to avoid overfitting.

For this Bayesian search a total of 100 runs were successfully completed. Figure 10 displays a sample of 5 of the runs for legibility. A fully iteractive version of the graph with all iterations is available on the FinDeBERTa Github repository.

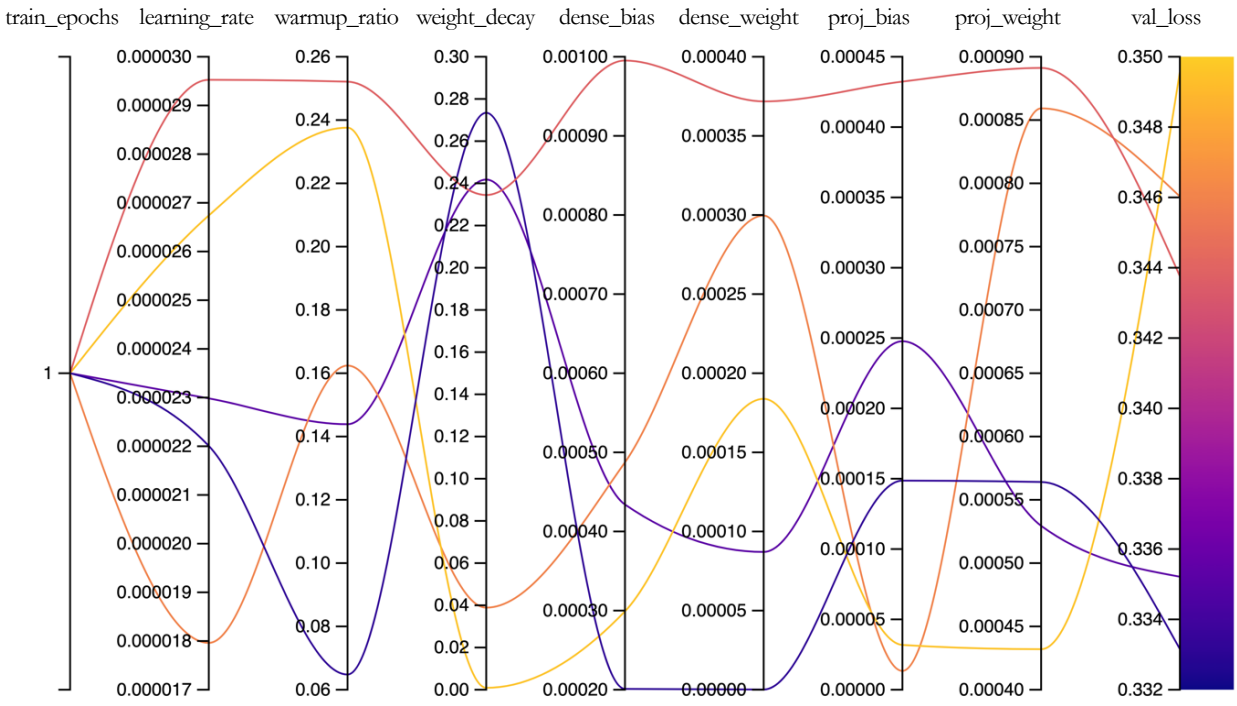



*Figure 10: Results of primary Bayesian Optimisation search.*

The figure shows that there is a definite improvement in validation loss, with all five samples showing better scores than the iterations displayed on the previous Bayesian search. The best performing run had a validation loss of 0.333.

One final Bayesian parameter search was run to find the best layer-wise learning combination for Discriminative Fine-tuning. Each of the 4 groups of 3 layers were assigned a learning rate in the range `"learning_rate": {"min": 0, "max": 3e-5}`. The rest of the parameters were set to the values that scored a validation loss of 0.333 in the previous search.

A technique known as Gradual Unfreezing is implemented. For this process all layers are first frozen meaning the weights cannot be updated [29]. During the training process, layers are iteratively unfrozen starting from the topmost layer group. Here, each layer group is trained on the target task with its respective learning rate. This

has been empirically shown to increase performance by preserving low-level representations and adapting high-level ones [29].

A total of 50 iterations were run and the best combination is as follows:

*Table 5: Best layer-wise learning rates for FinDeBERTa-Base.*

| Layer Group | Learning Rate (3.s.f) |
|---|---|
| 1-3 | $8.37^{-6}$ |
| 4-6 | $1.37^{-5}$ |
| 7-9 | $2.15^{-5}$ |
| 10-12 | $2.54^{-5}$ |

The best combination of those searched indicate that the lower levels of FinDeBERTa favour smaller learning rates. This intuitively makes sense as the higher layers contain task specific information, whereas the lower layers capture general representations. As such, it follows that smaller learning rates in lower levels may reduce incidence of catastrophic interference. The validation loss with these layer-wise learning rate values was 0.310. The four layer groups were expanded to cover 24 layers for the Large model where the validation loss was 0.274. This was the last stage of hyperparameter optimisation which saw the entire process improve validation loss from 0.381 to 0.310 (-0.071) for the Base model and 0.354 to 0.274 (-0.080) for the Large model **(RQ3)**.

Before evaluation was done on the test set, the number of training examples vs validation loss was searched to observe how the validation loss changed with respect to number of training samples. This was done for all models implemented. The models were evaluated on a validation set that was 10% of the full data set regardless of sample size. The data displayed is the mean of 3 runs using different random seeds.

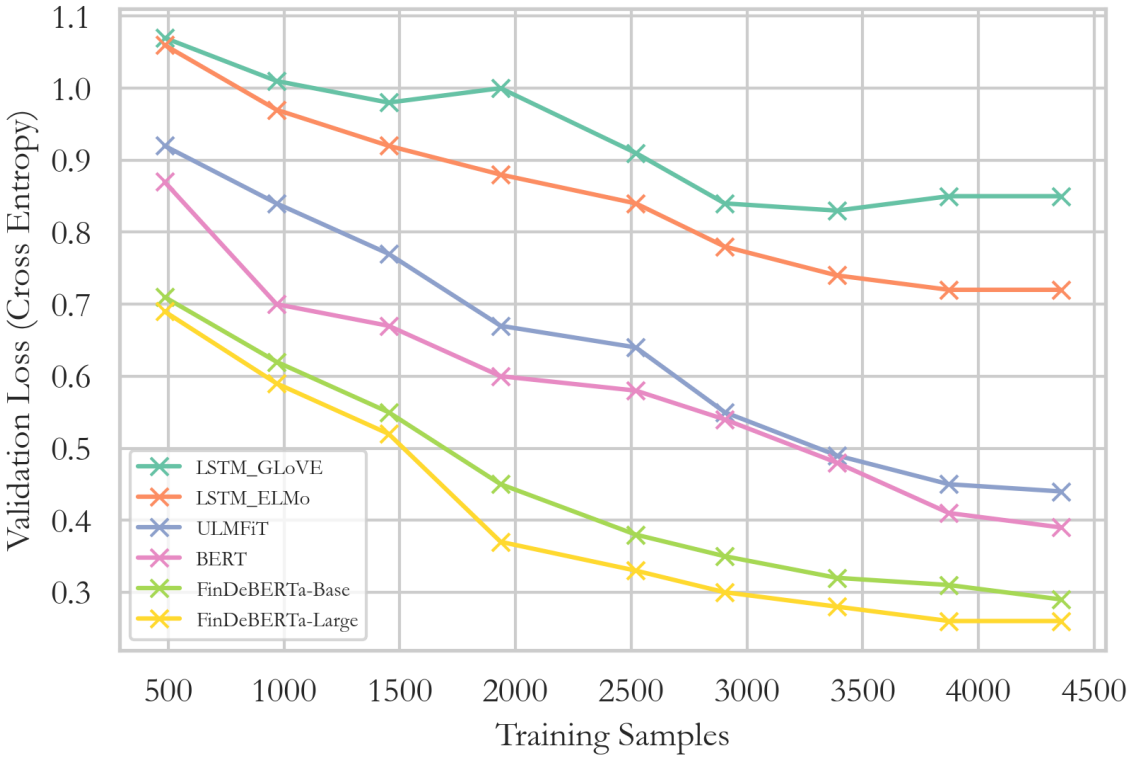



*Figure 11: Validation Loss of different Training Sample sizes.*

The chart indicates that all models improve as the number of training samples increases. Both LSTM based models seem to plateau in performance between 3000 to 4000 training samples. The ULMFiT and BERT models are closer in performance than expected, particularly as the number of training samples increases. This could be attributed to a strong hyperparameter configuration for ULMFiT. The FinDeBERTa models are the best performing by a margin of $\geq 0.1$ loss, with both models outperforming both LSTM implementations with as little as 500 training samples. This demonstrates the effectiveness of stacked encoders. Both FinDeBERTa models also continue to show a downward trend at 80% and 90% of training samples, suggesting that more samples may lead to better performance on the test set.

Consequently, the 10% validation splits previously used to evaluate the hyperparameter searches were added to their respective training sets in all 10 folds. This resulted in the data for experimental analysis becoming 10 folds of a 90-10 train-test split. The experimental results are displayed in Table 6 below.

*Table 6: Experimental Results on Financial Phrase-bank*

| | All Data | | | 100% Agreement | | |
|---|---|---|---|---|---|---|
| Model | Loss | Accuracy | Macro-F1 | Loss | Accuracy | Macro-F1 |
| Bi-LSTM with GLoVE | 0.80 | 0.70 | 0.64 | 0.57 | 0.81 | 0.74 |
| Bi-LSTM with ELMo | 0.69 | 0.76 | 0.71 | 0.50 | 0.84 | 0.77 |
| ULMFit | 0.44 | 0.81 | 0.78 | 0.20 | 0.93 | 0.91 |
| BERT | 0.41 | 0.83 | 0.81 | 0.20 | 0.94 | 0.92 |
| LPS | - | 0.71 | 0.71 | - | 0.79 | 0.80 |
| HSC | - | 0.71 | 0.76 | - | 0.83 | 0.86 |
| FinSSLX | - | - | - | - | 0.91 | 0.88 |
| FinBERT (Prosus AI) | 0.37 | 0.86 | 0.84 | 0.13 | 0.97 | 0.95 |
| FinDeBERTa-Base | 0.29 | 0.90 | 0.89±0.52 | 0.09 | 0.99 | 0.99 |
| FinDeBERTa-Large | **0.26** | **0.91** | **0.90**±0.74 | **0.07** | **1.00** | **1.00** |

**(RQ1)** FinDeBERTa's performance on the Financial Phrase-bank exceeds that of all the other models implemented, as well as models that were found throughout literature. The worst performing of all the models implemented is the Bi-LSTM with GLoVE embeddings. This can likely be attributed to its inability to infer context, as well as it not being robust to polysemy. The LSTM based LPS and HSC models were outperformed by all baseline models implemented except for the one with GLoVE embeddings. This highlights the importance of taking contextual differences into consideration in language modelling tasks.

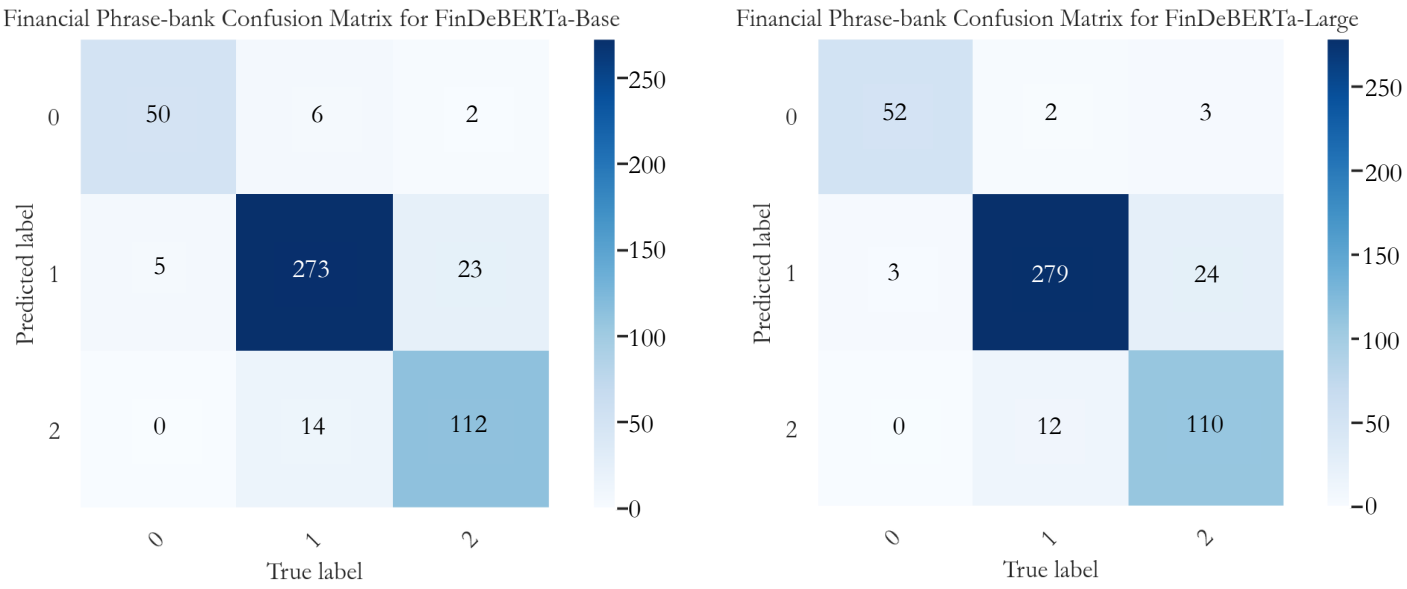
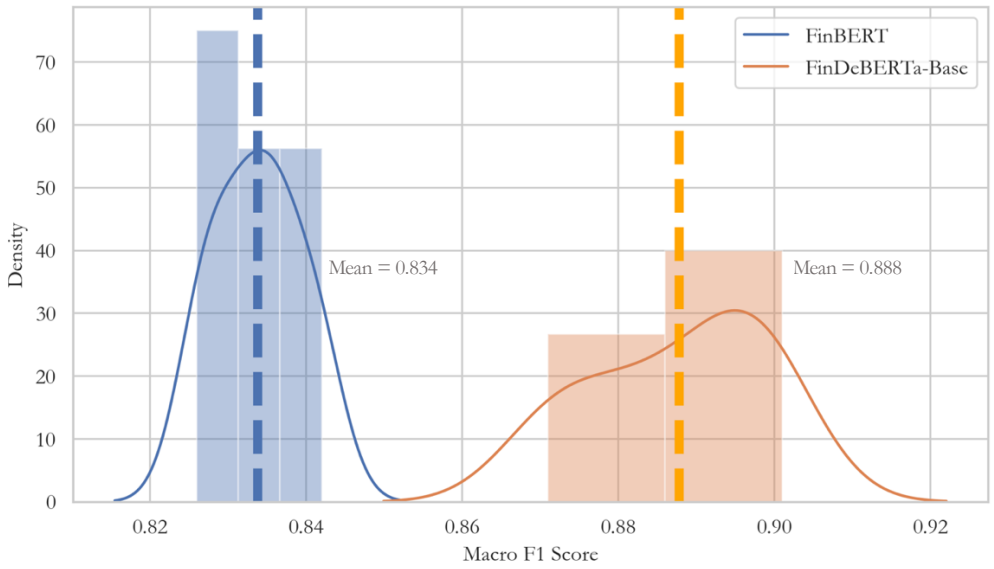



*Figure 12 : FinDeBERTa Confusion Matrices for full Financial Phrase-bank data set. 0 = negative, 1 = neutral, 2 = positive.*

The confusion matrices displayed in Figure 14 are the results from the cross folds that are closest to the mean for both models. The FinDeBERTa-Large confusion matrix shows a better performance across the negative and neutral labels when compared to the Base model. The Base model performs slightly better on positive news. Both models perform best on the neutral class label, which could be attributed to the class imbalance, with 59.3% of the data in the Financial Phrase-bank belonging to the neutral class. The most frequent mis-prediction from both models is a positive sequence being predicted as neutral. This is a trend that is commonly observed through literature including FinBERT where this error rate was more prominent [12].

**(RQ2)** FinDeBERTa-Base outperforms the previous state-of-the-art FinBERT by a healthy margin of 0.08 cross entropy loss and an f1 score that is 5% greater, despite having the same number of stacked encoders. A two-sample t-test was conducted to determine if the FinDeBERTa model had truly reached state-of-the-art performance. This was achieved by implementing the FinBERT model and training and evaluating it on the same 10 data folds that were used in the experimental analysis of FinDeBERTa. The results are as follows:

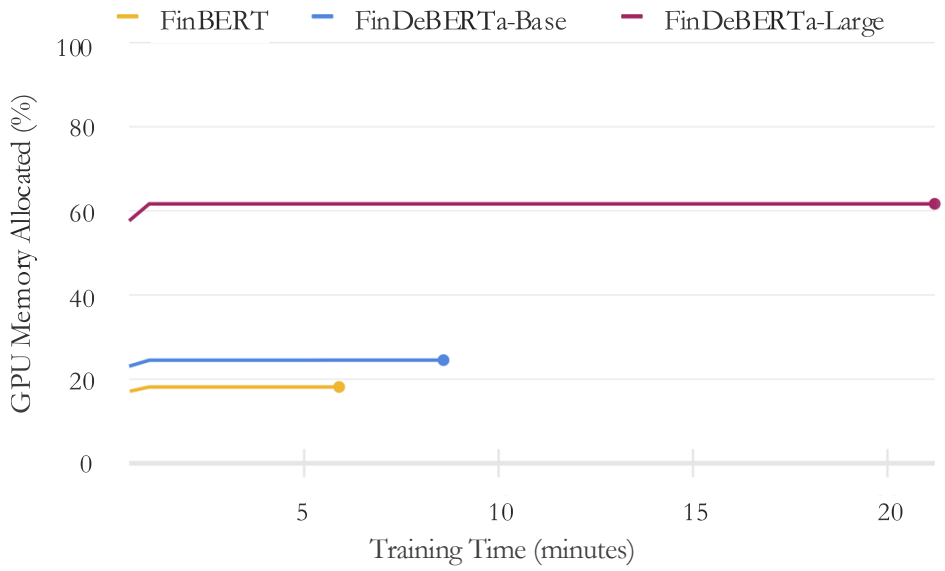
*Figure 13: Student t-test visualisation for FinBERT and FinDeBERTa-Base*




The t-test had a t-statistic of 13.7 and a p value of ~0.00. The large t statistic value of 13.7 suggests that the difference between group means is greater than the pooled standard error, indicating a significant difference between the models. The p value of ~0.00 indicates that it can be stated with near certainty that the FinDeBERTa-Base model outperforms the previous SOTA model FinBERT on the Financial Phrase-bank data set. The t-test for FinBERT against FinDeBERTa-Large is available in Appendix C. The standard errors are included in table 6 and are calculated by dividing the standard deviation by the square root of the number of samples.

The comparison with the Base model is significant as these two models are of similar magnitudes and take up similar VRAM as shown in the figure below.

*Figure 14: Model VRAM usage on a 48gb Quadro RTX 8000.*




Concretely, an industry level integration of FinDeBERTa-Base can provide a large improvement in performance with as little as 6% more VRAM dependency than the previous state-of-the-art. FinDeBERTa-Large requires almost three times the VRAM of the Base model while offering a small improvement with a 0.03 cross entropy loss improvement and a 1% greater f1 score. However, it does attain a 100% f1-score for labels with 100% agreement, reaching the human baseline in performance.

Some improvement from Base to Large is anticipated as the Large model has almost triple the number of parameters and more multi-attention heads, allowing the capture of deeper contextual understanding. However, the improvement from the Base to the Large model (+1.2% accuracy) is comparably small when observing the improvement of the DeBERTa architecture on the MNLI benchmark set (+2.5% accuracy) [14]. This could possibly suggest the upper limit of performance on the Financial Phrase-bank being neared and can be further supported by the perfect f1-

score achieved by the model on labels with 100% agreement. This could be remedied by populating the Financial Phrase-bank with more sequences so that more news in the financial domain is represented. This would also increase the viability of imposing a stricter margin of data agreement for experimental analysis.

## 3.3 Daily Stock Price Prediction Model

This subsection details the creation of the stock prediction model beginning with the data collection step, where data is collected over a 750-day period between 03/03/2019 and 20/03/2021. Sentiment analysis is conducted using FinDeBERTa and technical analysis is conducted using the MACD and RSI metrics. Seven unique feature sets are created for experimental analysis with a LSTM model.

### 3.3.1 Data Collection

The data collection process aimed to gather as much speculative data about general market sentiment and NASDAQ:FB as possible over the 750-day period. In an attempt to capture a fully representative data set containing both entity based and crowd based perception [21], data was sourced from a combination of news providers as well as the discussion forum Reddit.

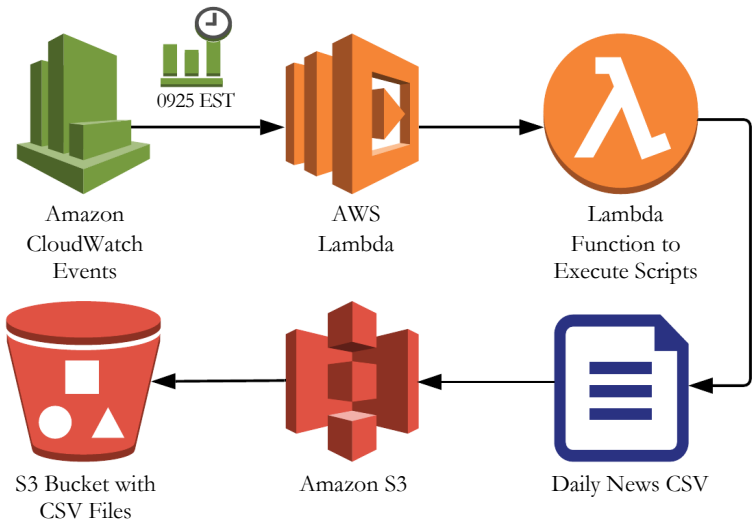



*Figure 15: Amazon Web Services architecture diagram for news data collection. Produced using Lucidchart.*

A total of 57 news providers were web scraped for articles. This included both non-advisory entities [ 'CNBC' , 'Forbes', 'Reuters' ] as well as advisory entities ['InvestorPlace' , 'Seeking Alpha', 'Market Watch' ]. This was done with the intention

that sentiment scores could be separated by source and treated as individual or aggregated hyperparameters. Additionally, general news and company specific news was captured independently in order to capture the general market sentiment as well as the specific sentiment for Facebook. Python scripts utilising the *requests* and *beautifulsoup* libraries were executed once per day at 0925 EST, as shown in Figure 15, five minutes before the NASDAQ opens for trading. A full list of news providers and their grouping is available in Appendix A.

The General and Facebook specific news data set distribution is as follows:

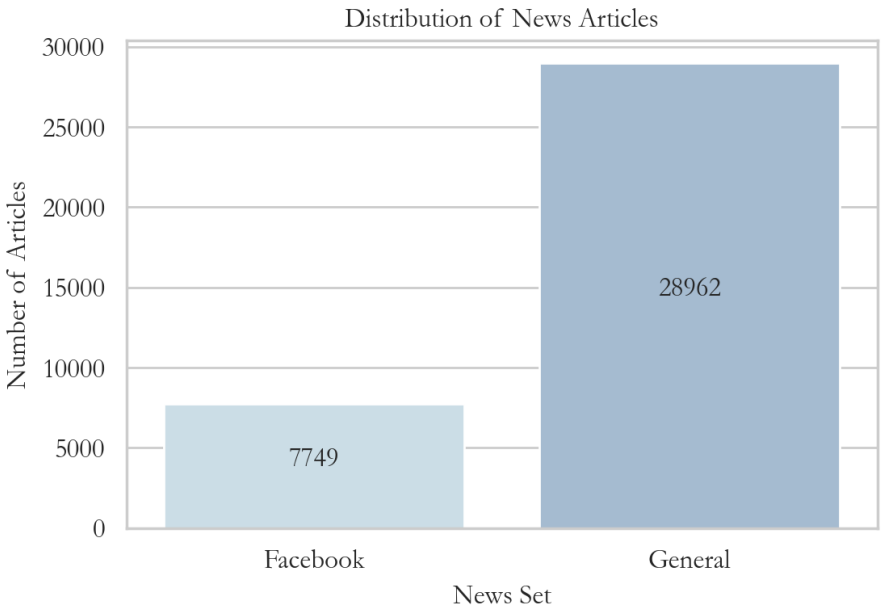
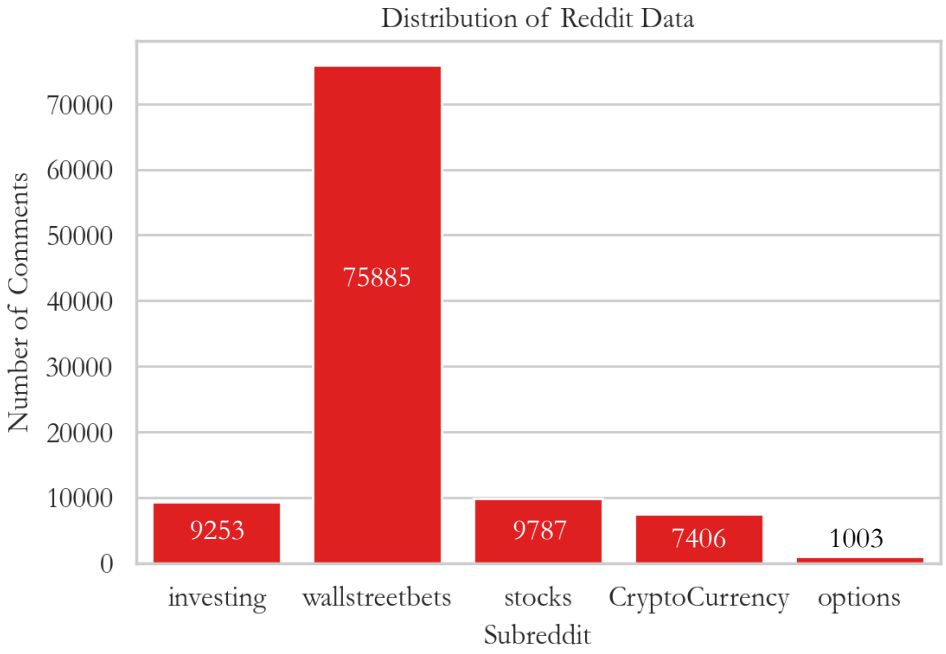



*Figure 16: Subset distribution of collected new articles.*

The Reddit data collection was done using the open source *pushshift* API. Five subforums or 'subreddits' were chosen to collect data from . The five subreddits were chosen based on a selection criteria that considers the size of the subreddit community as well as the mean number of monthly posts returned by regular expression, over the 750 day period**.** The subreddits selected were: /r/investing, /r/wallstreetbets, /r/stocks, /r/options and /r/CryptoCurrency. At the time of collection, the subreddits had a combined number of 17.4 million members and a total of 103,334 comments were captured.

*Figure 17: Count plot of number of comments collected from each subreddit.*




The Reddit data has an uneven distribution, with a large proportion (73.4%) being from the *wallstreetbets* subreddit. This can partially be attributed to *wallstreetbets* containing 10.0 out of the 17.4 million total subscribers. It can further be explained by the media attention garnered during the GameStop (GME) short squeeze [32], leading to a surge in unsubscribed users using *wallstreetbets*.

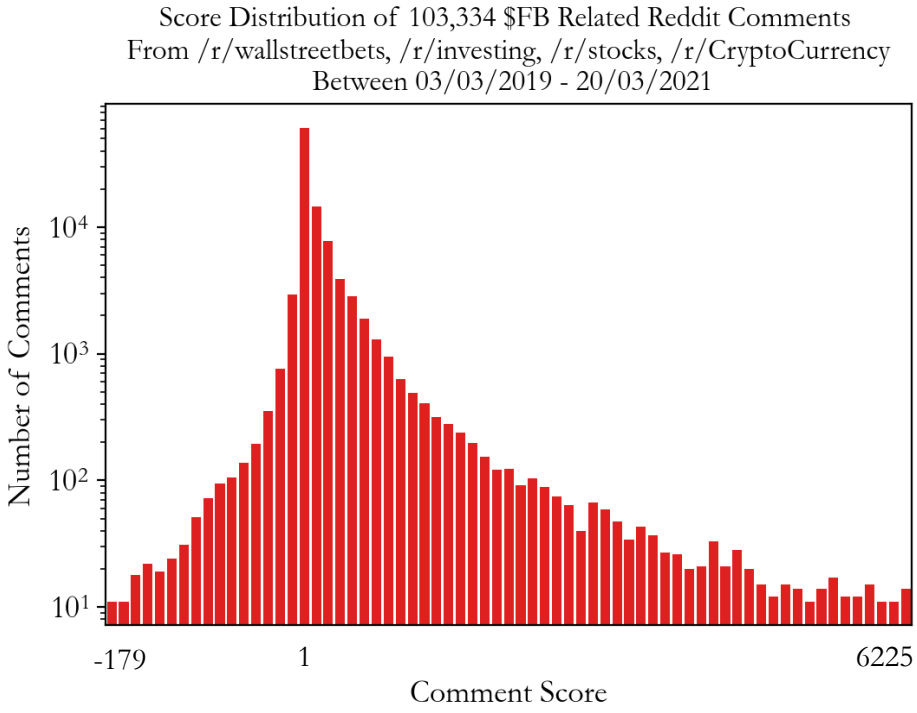



*Figure 18: Distribution of net comment scores for collected Reddit comments.*

Reddit has a voting based comment score system, whereby each unique user can increase or decrease the comment score by 1. This mechanism can be used as a metric to denote consensus about an opinion or position concerning Facebook. In order to exploit this, the data was **treated** as normally distributed. This was done as 58.8% of the comments had a score of 1 (see Figure 18), and the first standard deviation ($\sigma$) either side of the mean ($\mu$) contains 68.2% of the total in normally distributed data. This meant that by scaling the weighting of the data by $5^k \mid -1 > k > 1$ where $k$ is the standard deviation relative to the mean, all comments <-1 standard deviation away from the mean would be exponentially scaled down and similarly all comments >1 standard deviation away from the mean would be exponentially scaled up.

The full data set is comprised of 103,334 Reddit comments, 7,749 Facebook specific articles and 28,962 general market articles. Historical stock price data was collected for the same 750-day period by leveraging the *alphavantage* API.

### 3.3.2 Regression Model Loss Metric

Regression models commonly use the Mean Squared Error (MSE) metric as a loss function, both during training and evaluation. Root Mean Squared Error (RMSE) improves the metric by indicating the mean deviation of the points from the predicted hyperplane. Consequently, RMSE was used as a loss metric for the model, as it commonly is throughout similar literature [21].

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{pred} - y_{actual})^2} \tag{16}$$

### 3.3.3 Regression Model Implementation

The objective of this exercise is to demonstrate how sentiment analysis from FinDeBERTa affects a model's ability to predict the next day's closing price. Several feature set combinations were made to determine which type of analysis yielded the lowest error, as well as provide insight into the types of data that may be useful for this task. Therefore, the significance of the results is in the improvement between feature sets rather than the RMSE score itself. Furthermore, this task is specific to $FB and predicts next day closing prices rather than forecasting as is common in most papers [18]. This makes it difficult to compare the performance of the best optimised model without reimplementing other papers with the appropriate adjustments to fit the task.

An LSTM architecture was employed to create the model. The Tensor-Flow library was utilised for the LSTM architecture by stacking layers. The feature set combinatons are shown in Table 7 below.

*Table 7: Analysis types and corresponding feature combinations for stock price prediction model.*

| Analysis Type | Features |
|---|---|
| Historical Data | Adj. Close, Volume |
| Technical Analysis | Adj. Close, Volume, MACD, RSI |
| Sentiment Analysis | Adj. Close, Volume, Reddit, General News, Facebook News |
| Technical and Sentiment Analysis I. | Adj. Close, Volume, MACD, RSI, Reddit, General News, Facebook News |
| Technical and Sentiment Analysis II. | Adj. Close, Volume, MACD, RSI, Reddit, General News, Facebook News, where news sentiment scores are separated by advisory and non-advisory entities |
| Technical and Sentiment Analysis III. | Adj. Close, Volume, MACD, RSI, Reddit |
| Technical and Sentiment Analysis IV. | Adj. Close, Volume, MACD, RSI, Reddit, Facebook News |

Due to time and space constraints, the Technical and Sentiment analysis feature set was limited to four variations. Variant I. was chosen to observe the effect of the whole data set. Variant II. was chosen to observe the difference when sentiment scores for the news articles in the data set were separated by advisory and non-advisory entities. Variant III. was chosen to observe how social media sentiment analysis performed in isolation. Variant IV. was chosen as through literature review, it was found that the best performing models were predominantly informed by a combination of company specific news and social media data.

As stated, the total data set is comprised of 750 days of data from 03-03-2019 to 20-03-2021. For the model implementation, the data was split into a train and test set with a ratio of 70:30. This amounted to 525 days of training data and 225 for testing. For each analysis type there were two stages of evaluation. The initial stage used hyperparameters that were sourced following background reading of similar works. Next, hyperparameter tuning was conducted, and the best hyperparameters found are used on the test set in the optimised stage.

The model was first fit to the training split, and then the test set was used to iteratively make predictions about the next day's closing price. The RMSE for all analysis types using unoptimized hyperparameters is shown in Table 8 below.

*Table 8: Experimental results for initial models.*

| Analysis Type | RMSE (3.s.f) |
|---|---|
| Historical Data | 5.91 |
| Technical Analysis | 4.78 |
| Sentiment Analysis | 4.40 |
| Technical and Sentiment Analysis I. | 4.88 |
| Technical and Sentiment Analysis II. | 3.95 |
| Technical and Sentiment Analysis III. | 2.89 |
| Technical and Sentiment Analysis IV. | **2.68** |

The initial results show that in general the inclusion of sentiment analysis seems to be the differential for better RMSE scores. T&S Analysis I is an outlier in that its performance is worse than technical analysis alone. This could be attributed to a high level of noise in the data, as this feature set had minimal additional feature selection steps compared to the rest. Additionally, the results seem to suggest that it is favourable to ignore general market news and instead only focus on company specific discourse to inform decisions. The best performing metric T&S Analysis IV follows the trend observed in literature of a combination of company specific news and social media producing the best results.

### 3.3.4 Hyperparameter Optimisation

For the hyperparameter optimisation step, nested validation was used. This utilises the training set by splitting it further into a training and validation set. It is utilised only during the hyperparameter optimisation in order to evaluate the performance of different values while avoiding snooping bias on the test set. Twenty percent of the training set was utilised as the nested validation set, making the reduced training set 420 days and the nested validation set 105 days. As the data is time-series, the nested validation set is taken from the end portion of the training set. Additionally, all feature rescaling was done after the data sets were split to prevent any data leakage into the training data. All hyperparameter search graphs shown are from the search of Technical and Sentiment Analysis IV.

The first parameter searched was the time step. To calculate the best performing value, timestep values between 2 and 30 were tested. The results are shown in Figure 19 below.

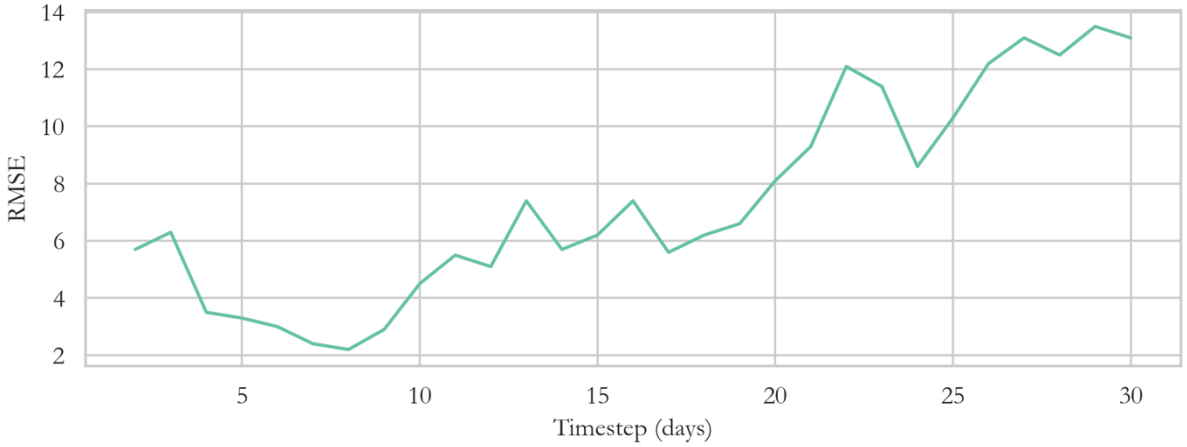



*Figure 19: RMSE for Timestep values between 2 and 30.*

The Training Batch Size and the Number of Epochs were also tested. These were tested together as their values affect each other. The Training Batch Size values tested were [8, 16, 32, 64] and the Number of Epochs values were [5, 10, 20, 40, 60, 80, 100].

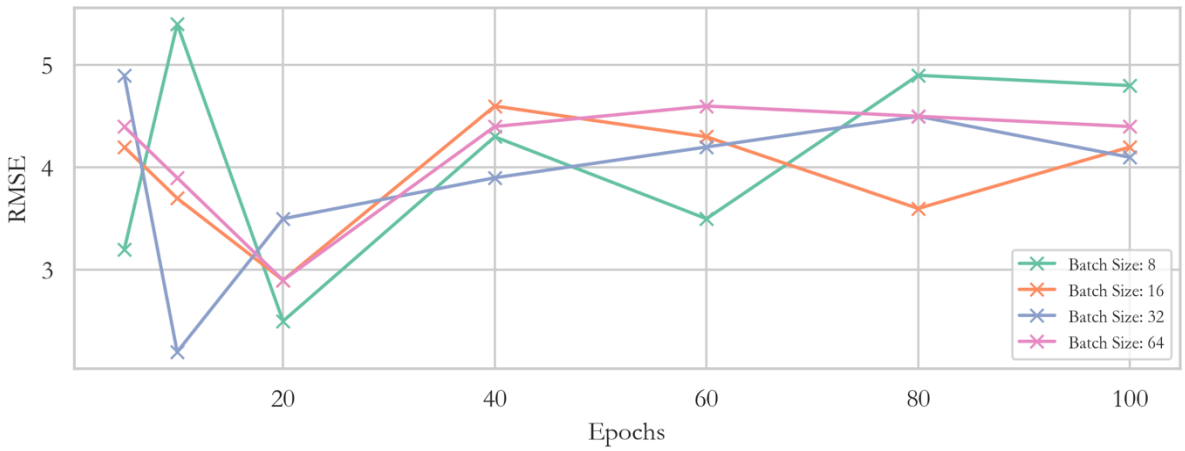



*Figure 20: RMSE for different batch sizes across a range of different epochs.*

Similarly the dropout probability as well as the number of neurons per layer were tested simultaneously. The dropout probabilities tested were [0.3, 0.4, 0.5, 0.6, 0.7] and the number of neurons per layer tested were [5, 10, 15, 20, 25, 30]. The results are shown in Figure 21 below.

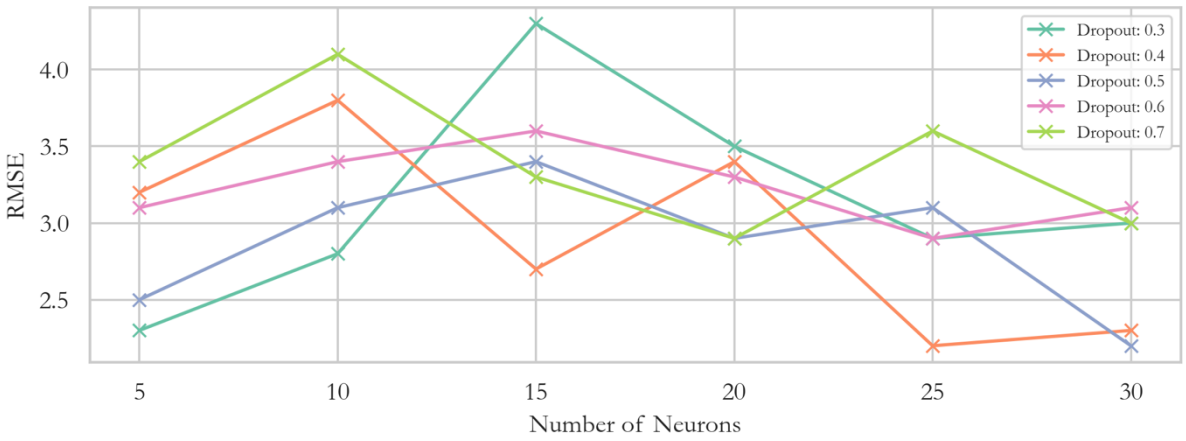



*Figure 21: RMSE for different dropout probabilities across a range of number of neurons.*

Finally, the optimiser used was tested. Those searched were ['adadelta', 'adagrad', 'adam', 'adamax', 'rmsprop', 'sgd']. The results are shown in Figure 22 below.

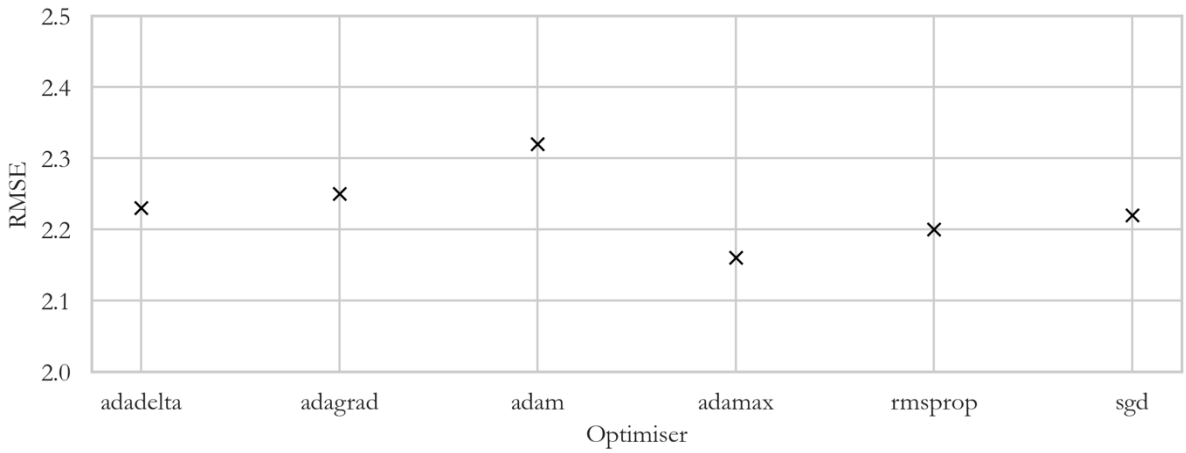



*Figure 22: RMSE with different optimisers.*

*Table 9: Experimental results for optimised model.*

| Analysis Type | RMSE (3.s.f) | Change |
|---|---|---|
| Historical Data | 5.68 | -0.23 |
| Technical Analysis | 4.71 | -0.07 |
| Sentiment Analysis | 4.14 | -0.26 |
| Technical and Sentiment Analysis I. | 4.23 | **-0.65** |
| Technical and Sentiment Analysis II. | 3.68 | -0.27 |
| Technical and Sentiment Analysis III. | 2.50 | -0.39 |
| Technical and Sentiment Analysis IV. | **2.13** | -0.55 |

*Table 10: Parameter comparison between initial best model and optimised best model.*

| Parameter | Initial Best Model | Optimised Best Model |
|---|---|---|
| Analysis Type | T&S Analysis IV. | T&S Analysis IV. |
| Timestep (days) | 4 | 7 |
| Epochs | 40 | 10 |
| Training Batch Size | 16 | 32 |
| Number of Neurons | 64 | 25 |
| Dropout Probability | 1.0 | 0.4 |
| Optimiser | nadam | adamax |
| RMSE | 2.68 | 2.13 |

T&S Analysis IV remained the best performing model after optimisation. The parameter changes are outlined in Table 10 above. The results of the optimised models in Table 9 support the hypothesis that sentiment analysis from FinDeBERTa has a positive impact on the RMSE. Foreseeably, the selection of data used is far more important to the success of the model. This can be evidenced by the improvement from Technical analysis ranging from a 10.2% to 54.7% decrease in RMSE depending on the sources used for sentiment analysis.

# Chapter 4

# Project Management

The following file management tools were used throughout the project:

- Github – Version control for all project files including scraped data sets and Jupyter notebooks
- Google Drive – Cloud storage system used as secondary backup for all project files
- Southampton Microsoft One Drive – Cloud storage system used for tertiary backup for all project file

## 4.1 Risk Mitigation Plan

*Table 11: Risk Mitigation Plan* $Risk\ Exposure = Probability\ \times\ Severity$

| Risk | Probability | Severity | Risk Exposure | Mitigation Strategy |
|---|---|---|---|---|
| Loss of data | 2 | 7 | 14 | Automatic backup measure implemented for all 3 backup tools outlined above. |
| Data set for sentiment analysis not large enough for deep learning | 5 | 3 | 15 | Restrict data sources to those that can be filtered by date and collect a backdated data set. |
| Initial and backdated data set both provided very limited signal data | 3 | 4 | 12 | Use DJI data set used for preliminary testing from Kaggle. |
| Tasks taken longer than anticipated / unexpected additional tasks | 3 | 3 | 9 | Weekly progress reflections that consider the GANTT chart to implement any contingencies necessary. |
| Hardware not suitable for intensity of deep learning experiments | 4 | 2 | 8 | 1) Make use of Google Cloud Platform (GCP) Compute Engine. 2) Make use of ECS GPU service. |
| Severe Illness | 2 | 4 | 8 | Apply for special considerations or deadline extension. Additionally, ensure sufficient time in the GANTT chart between completed implementation and project report deadline. |

# Chapter 5

# Conclusion

This project had a positive outcome overall and all the objectives established were completed. The core focus of the paper, 'FinDeBERTa', exceeded expectations by achieving a new state-of-the-art performance on the Financial Phrase-bank data set with an F1 score improvement of 6%. Furthermore, the daily stock price prediction model successfully demonstrated how using sentiment analysis predictions from FinDeBERTa can improve the performance of stock price prediction models.

## 5.1 Further Advancements

Arguably the biggest improvement that could be made to FinDeBERTa would be providing the model with more training data to cover a wider range of nuances in financial news. This is of particular importance as although the data set is highly reliable, it is quite small. The necessity for additional data is supported by indicators found during evaluation suggesting that the performance on the data set was nearing its peak. Namely, the perfect F1 score on labels with 100% agreement.

One area of interest that the stock price prediction model does not explore is how the RMSE changes when using a range of different sentiment analysers. This would highlight the model to model impact of sentiment analysis, demonstrating to what degree, if any, a more accurate sentiment analyser such as FinDeBERTa has on price prediction performance. Additionally, it would be interesting to deeper explore the possible feature set combinations for sentiment analysis. An observation made was that several text sequences that were analysed were not as focused about the Facebook stock as they should have been. So additional sequence filtering by use of

regular expressions is also another area that could be explored. The full data set used will be made publicly available on the FinDeBERTa Github repository.

**Word Count: 9955**

# Bibliography


[1] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, Jun. 2017, vol. 2017-December, pp. 5999–6009, Accessed: Dec. 17, 2020. [Online]. Available: https://arxiv.org/abs/1706.03762v5.

[2] "Forces That Move Stock Prices." https://www.investopedia.com/articles/basics/04/100804.asp (accessed Dec. 05, 2020).

[3] "Market Sentiment Definition." https://www.investopedia.com/terms/m/marketsentiment.asp (accessed Dec. 05, 2020).

[4] "Moving Average Convergence Divergence (MACD) Definition." https://www.investopedia.com/terms/m/macd.asp (accessed Feb. 26, 2021).

[5] "Relative Strength Index (RSI) Definition & Calculation." https://www.investopedia.com/terms/r/rsi.asp (accessed Mar. 01, 2021).

[6] "Artificial Intelligence: Definition, Types, Examples, Technologies | by Chethan Kumar GN | Medium." https://chethankumargn.medium.com/artificial-intelligence-definition-types-examples-technologies-962ea75c7b9b (accessed Apr. 26, 2021).

[7] "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition [Book]." https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/ (accessed Dec. 21, 2020).

[8] "Understanding LSTM and its diagrams | by Shi Yan | ML Review." https://blog.mlreview.com/understanding-lstm-and-its-diagrams-37e2f46f1714 (accessed May 02, 2021).

[9] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation." Accessed: Feb. 27, 2021. [Online]. Available: http://nlp.

[10] "The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time." http://jalammar.github.io/illustrated-transformer/ (accessed May 03, 2021).

[11] "BERT Explained: State of the art language model for NLP | by Rani Horev | Towards Data Science." https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270 (accessed May 04, 2021).

[12] "Transformer (machine learning model) - Wikipedia." https://en.wikipedia.org/wiki/Transformer_(machine_learning_model)#Multi-head_attention (accessed Dec. 28, 2020).

[13] J. Devlin, M.-W. Chang, K. Lee, K. T. Google, and A. I. Language, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." Accessed: Oct. 12, 2020. [Online]. Available: https://github.com/tensorflow/tensor2tensor.

[14] "Microsoft DeBERTa surpasses human performance on the SuperGLUE benchmark - Microsoft Research." https://www.microsoft.com/en-us/research/blog/microsoft-deberta-surpasses-human-performance-on-the-superglue-benchmark/ (accessed May 04, 2021).

[15] P. He, X. Liu, J. Gao, and W. Chen, "DeBERTa: Decoding-enhanced BERT with Disentangled Attention," *arXiv*, Jun. 2020, Accessed: Jan. 05, 2021. [Online]. Available: http://arxiv.org/abs/2006.03654.

[16] "Market Sentiment - Overview, Impact on Prices, Trading Strategies." https://corporatefinanceinstitute.com/resources/knowledge/trading-investing/market-sentiment/ (accessed Dec. 05, 2020).

[17] M. Kraus and S. Feuerriegel, "Decision support from financial disclosures with deep neural networks and transfer learning," *Decis. Support Syst.*, vol. 104, pp. 38–48, Oct. 2017, doi: 10.1016/j.dss.2017.10.001.

[18] K. Mishev, A. Gjorgjevikj, I. Vodenska, L. T. Chitkushev, and D. Trajanov, "Evaluation of Sentiment Analysis in Finance: From Lexicons to Transformers," *IEEE Access*, vol. 8, pp. 131662–131682, 2020, doi: 10.1109/ACCESS.2020.3009626.

[19] T. Loughran and B. Mcdonald, "The Use of Word Lists in Textual Analysis," *J. Behav. Financ.*, vol. 16, no. 1, pp. 1–11, 2015, doi: 10.1080/15427560.2015.1000335.

[20] P. Malo, A. Sinha, P. Takala, P. Korhonen, and J. Wallenius, "Good Debt or Bad Debt: Detecting Semantic Orientations in Economic Texts," 2013.

[21] S. Krishnamoorthy, "Sentiment Analysis of Financial News Articles using Performance Indicators," *Knowl. Inf. Syst.*, vol. 56, no. 2, pp. 373–394, Nov. 2018, doi: 10.1007/s10115-017-1134-1.

[22] M. Maia, A. Freitas, and S. Handschuh, "FinSSLx: A Sentiment Analysis Model for the Financial Domain Using Text Simplification," in *Proceedings - 12th IEEE International Conference on Semantic Computing, ICSC 2018*, Apr. 2018, vol. 2018-Janua, pp. 318–319, doi: 10.1109/ICSC.2018.00065.

[23] D. T. Araci, "FinBERT: Financial Sentiment Analysis with Pre-trained Language Models," 2019.

[24] "Stock Price Prediction Using Machine Learning | Deep Learning." https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/ (accessed Jan. 07, 2021).

[25] "Multi-Class Metrics Made Simple, Part II: the F1-score | by Boaz Shmueli | Towards Data Science." https://towardsdatascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1 (accessed Jan. 10, 2021).

[26] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A MULTI-TASK BENCHMARK AND ANALYSIS PLATFORM FOR NATURAL LANGUAGE UNDERSTAND-ING."

[27] Y. Liu *et al.*, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," *arXiv*, Jul. 2019, Accessed: Dec. 17, 2020. [Online]. Available: http://arxiv.org/abs/1907.11692.

[28] A. Conneau *et al.*, "Unsupervised Cross-lingual Representation Learning at Scale." Accessed: Jan. 11, 2021. [Online]. Available: https://github.com/facebookresearch/cc.

[29] J. Howard and S. Ruder, "Universal Language Model Fine-tuning for Text Classification." Accessed: Jan. 15, 2021. [Online]. Available: http://nlp.fast.ai/ulmfit.

[30] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient Global Optimization of Expensive Black-Box Functions," *J. Glob. Optim.*, vol. 13, no. 4, pp. 455–492, 1998, doi: 10.1023/A:1008306431147.

[31] J. Mockus, "Application of Bayesian approach to numerical methods of global and stochastic optimization," *J. Glob. Optim.*, vol. 4, no. 4, pp. 347–365, Jun. 1994, doi: 10.1007/BF01099263.

[32] "WallStreetBets: The Big Short-Squeeze | Nasdaq." https://www.nasdaq.com/articles/wallstreetbets%3A-the-big-short-squeeze-2021-01-26 (accessed Feb. 02, 2021).

# Appendices

# A News Sources

24/7 Wall Street
**Accesswire**
Barrons
**Benzinga**
Bloomberg Markets and Finance (video)
Bloomberg Technology (video)
Business Insider
Business Wire
**Cheddar Videos**
CNBC
CNBC International TV
CNBC Television
CNET
CNN
CNN Business
**Deadline**
**Digital Trends (video)**
**Discount The Obvious**
**Engadget (video)**
**ETF Trends**
ETF.com
**Fast Company**
Forbes
Fox Business
**FreightWaves**
**GeekWire**
**Globe News Wire**
**GuruFocus**
Huffington Post
**Investopedia**
**Investor Place**
**Investors Business Daily**
**Invezz**
**Kiplinger**
**Market Watch**
**Morningstar Inc.**
New York Post
New York Times
**Newsfile Corp**
**OTC PR Wire**
PR Newswire
**Pulse2**
PYMNTS.com
Reuters
**Schaeffers Research**
**See It Market**
**Seeking Alpha**
Skynews
**TechCrunch**
**The Dog of Wall Street**
The Guardian
**The Motley Fool**
**The Street (video)**
**The Verge (video)**
VentureBeat
Wall Street Journal (video)
Yahoo Finance (video)
**Zacks Investment Research**

*Bold entries belong to the *advisory entity* group.

# B FinDeBERTa Test Results

FinDeBERTa-Base

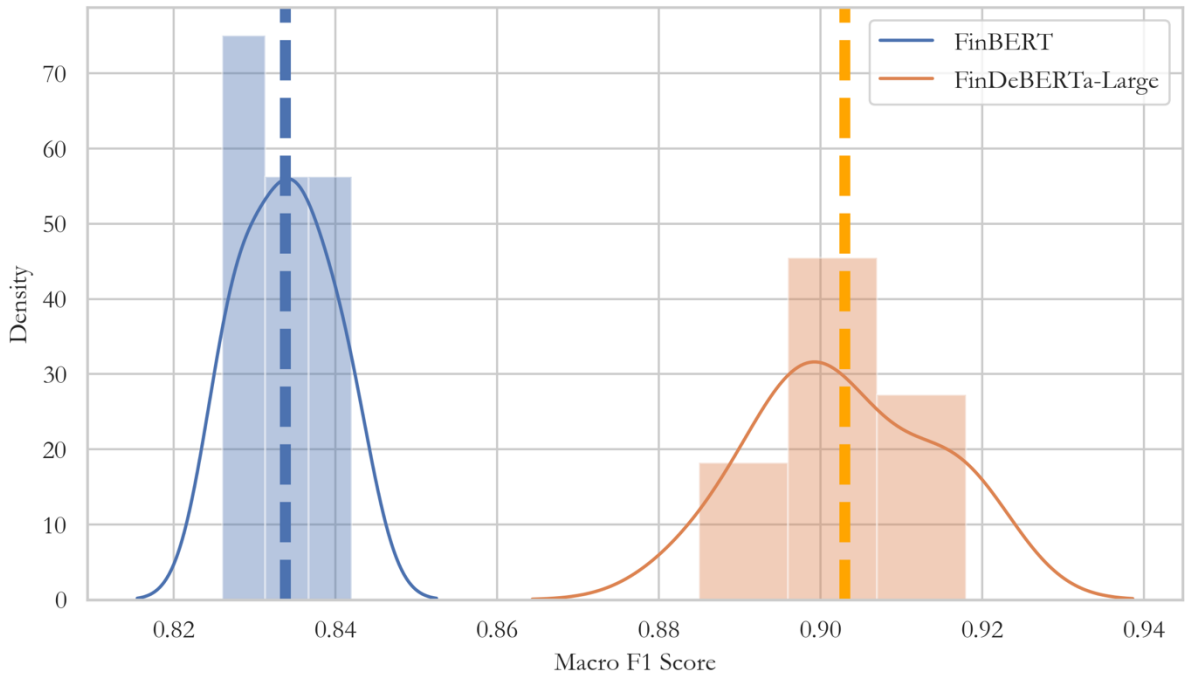
acc = 0.8969072164948454
eval_loss = 0.29301910742305097
f1_macro = 0.8852862326734261
f1_weighted = 0.896247854311962
mcc = 0.8090720732032607

—

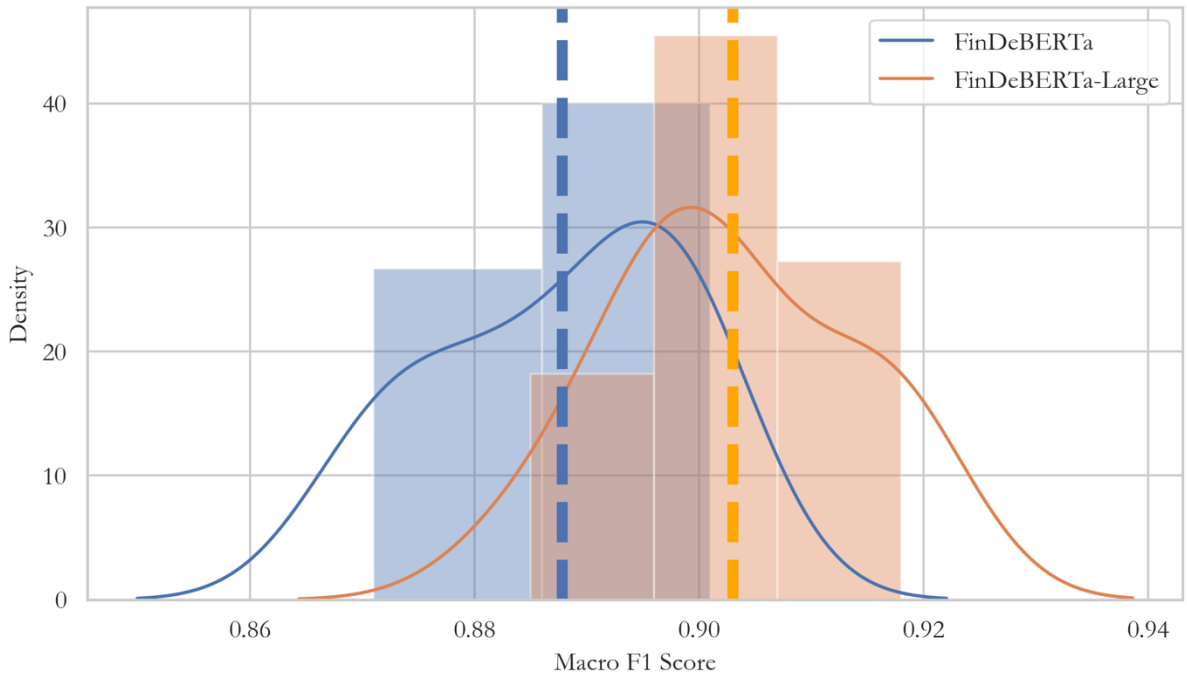
FinDeBERTa-Large

acc = 0.9092783505154639
eval_loss = 0.26176932497288205
f1_macro = 0.9031816218794515
f1_weighted = 0.908014418809042
mcc = 0.8316836565687391

# C T-Tests



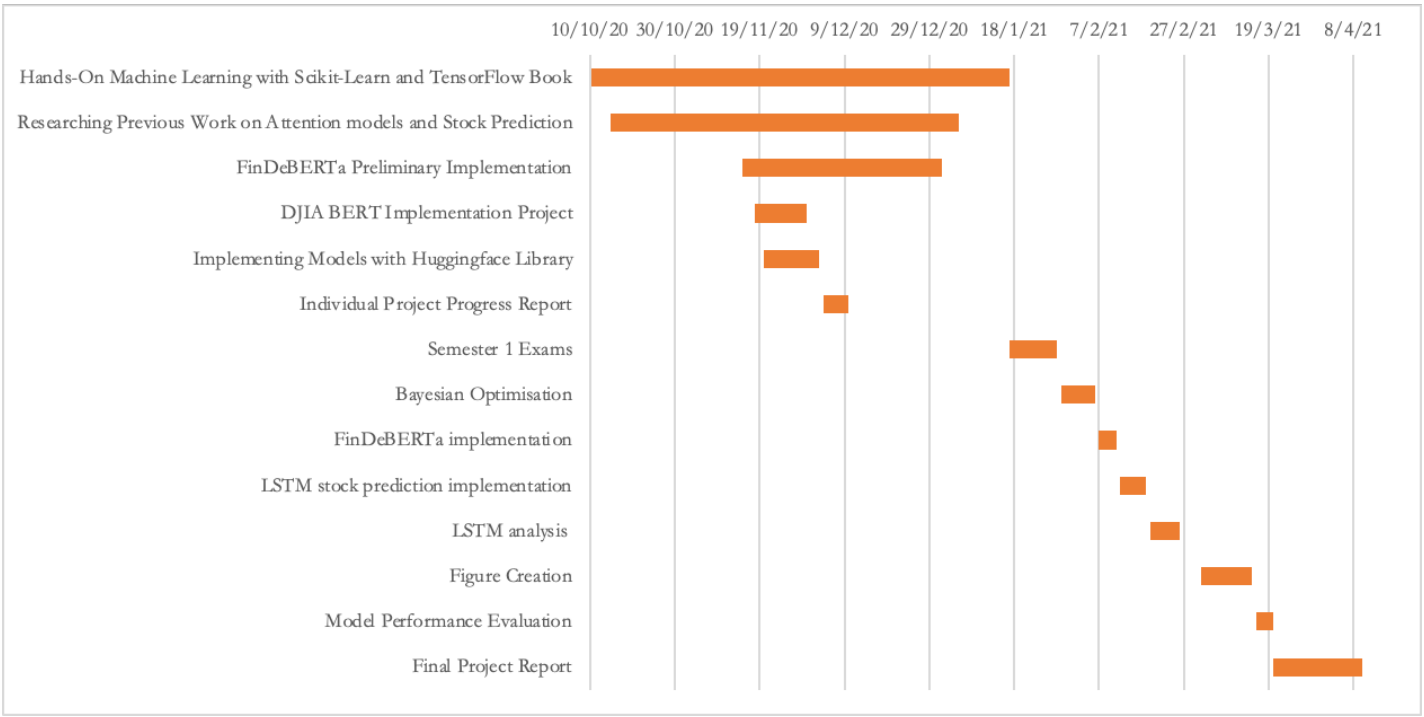

t-statistic = 17.9 , P = ~0.000




t-statistic = 3.09 , P = 0.006

# D GANTT Charts

Planned GANTT Chart




Actual GANTT Chart