University of Glasgow | School of Computing Science

# Adapting Natural Language Processing and Deep Learning Tools to an end-to-end Day Trading System

Enrique Alejandro González Amador

School of Computing Science
Sir Alwyn Williams Building
University of Glasgow
G12 8QQ

A dissertation presented in part fulfilment of the requirements of the Degree of Master of Science at The University of Glasgow

5th September 2025

## Abstract

Stock prediction has been extensively studied, and researchers and investors continually leverage state-of-the-art techniques to gain a competitive edge. Day trading remains especially challenging due to market volatility and the noise that must be filtered to extract predictive signals for trading decisions. This project develops an end-to-end system combining deep learning and natural language processing techniques with a robust staking and risk management strategy. Experiments, including evaluations and investment simulations, showed that significant feature engineering would be required to integrate tweet-derived market data with stock technical indicators. Moreover, since tweet data were not available for every trading day, models using only technical indicators consistently outperformed those including tweet features. Nevertheless, the end-to-end system achieved returns of 1.61% over one hundred trading days, providing a foundation for further development into a competitive day trading approach.

## Education Use Consent

I hereby give my permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic format. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Name: Enrique Alejandro Gonzalez Amador Signature: Alex G.

# AI Usage Declaration

In preparing this dissertation, I made use of Artificial Intelligence (AI) tools in the following ways:

- For code augmentation and writing aid with respect to style and correctness.
- For generating summaries and supporting the initial exploration of topic literature.

All ideas, arguments, analyses, and conclusions presented in this dissertation are entirely my own. I have complete knowledge and understanding of all source code and text included.

My use of AI has been in full compliance with the University of Glasgow's policy on the ethical and transparent use of AI in academic work. At no stage has AI been used as a substitute for my own critical thinking, evaluation of sources, or the formation of arguments and conclusions.

# Acknowledgements

My sincere gratitude to my supervisor, Joemon Jose, for their guidance and support throughout this project.

Thanks also to the Ministry of Science, Humanities, Technology and Innovation of Mexico, which continues to create opportunities for Mexicans to go abroad to the top academic circles worldwide. Without their funding, none of this would have been possible.

To the School of Computing Science and the department in charge of the University's cluster, for graciously allowing me to use the hardware and infrastructure which allowed me to produce the results showcased on this report.

Lastly, to my friends, family and loved ones, who always cheer me on to keep going.

# Contents

# Chapter 1: Introduction

## 1.1 Motivation

Stock market investment has long been a research area of big interest, attracting both academia for its complex dynamics and the financial industry for its potential for profit. The central problem within this domain, particularly for short-term strategies like day trading—where positions must be closed on the same trading day—is the accurate prediction of stock price movements. This is a formidable challenge, as day trading environments are characterized by high volatility and a low signal-to-noise ratio, making it incredibly difficult to extract clear, predictive signals from the vast amount of available data. To tackle this problem, researchers and practitioners have traditionally relied on two primary sources of information:

1. Quantitative Historical Data: This includes historical stock prices (open, high, low, close) and trading volumes. This structured data forms the basis of technical analysis, where indicators like moving averages and the Relative Strength Index (RSI) are used to identify patterns and trends. The rationale for using this data is that historical price action can often be indicative of future movements. However, a significant limitation is that these indicators are inherently backward-looking and may fail to capture sudden shifts in market dynamics driven by new information.

2. Unstructured Textual Data: With the rise of social media, platforms like Twitter have become a vast, real-time repository of public opinion and sentiment. This alternative data source offers the potential to capture the forward-looking ”market mood” that often precedes price movements. The intuition is that collective sentiment can be a powerful driver of short-term market behavior. The challenge, however, lies in a different kind of noise: extracting meaningful investment signals from millions of informal, often irrelevant, tweets is a complex Natural Language Processing (NLP) task.

This project is motivated by the hypothesis that a more robust and accurate prediction model can be built by synergistically combining these two disparate data types. By integrating traditional, backward-looking technical indicators with real-time, forward-looking sentiment signals from social media, we aim to develop an end-to-end trading system that can better navigate the noisy environment of day trading. The development of such a system requires not only sophisticated predictive models but also robust risk management strategies to create a consistently profitable approach.

## 1.2 Purpose

The primary aim of this project is to develop and evaluate an end-to-end day trading system that integrates deep learning models with Natural Language

Processing (NLP) techniques. This project uses the work of ”Stock movement prediction from tweets and historical prices” from Xu and Cohen [2018] as a starting point. However, this project is focused on having a more basic predicting model and embedding it on a complete, automated pipeline that includes a robust risk and capital allocation strategy to determine the viability of such an approach.

Specifically, this report details the investigation into whether sentiment, emotion, and stance signals—extracted from tweets using BERT-based models—can be effectively leveraged alongside traditional technical indicators to improve the predictive accuracy of an LSTM-based model. The ultimate goal is to determine if this integrated system can achieve positive returns in a realistic, simulated trading environment by using the Kelly Criterion for position sizing and a sector-based portfolio construction method.

To address this, the system incorporates two key principles:

- Optimal Position Sizing using the Kelly Criterion: Rather than investing a fixed amount, the system uses the Kelly Criterion to dynamically size each trade. The rationale for this choice is that the Kelly Criterion provides a mathematically rigorous framework for maximizing growth rate of capital. By basing investment size on the model’s predicted probability of success and the historical payout ratio, it balances risk and reward, helping to prevent losses associated with over-leveraging. A conservative ”Fractional Kelly” approach is used to further mitigate risk.
- Diversification through Sector-Based Selection: To avoid concentrating risk in a single industry, the system employs a sector-based portfolio construction method. This strategy involves selecting the most promising trade from different market sectors each day. This enforces diversification, reducing the portfolio’s vulnerability to adverse events affecting a specific sector and leading to more stability and resilience to market changes.

To give structure to the investigation, the project is guided by the following objectives:

- Objective 1: Comprehensive Feature Engineering
- Objective 2: Predictive Model Development and Evaluation
- Objective 3: End-to-End System Implementation

The complete discussion of the research objectives is continued on chapter 2.

## 1.3 Report Structure

This dissertation is organized into five chapters. Chapter 2, Background, reviews the core technologies and theoretical concepts underpinning this project, including Long Short-Term Memory (LSTM) networks, BERT, the Kelly Criterion, and Isotonic Regression. It also formally states the project’s research objectives and questions. Chapter 3, Design & Implementation, provides a detailed description of the system architecture, the data processing pipeline, the feature engineering process for both stock and tweet data, and the design of the investment simulation framework. Chapter 4, Experiments & Results, presents the empirical findings of the study. It includes a comparison of different model architectures, an analysis of the impact of various feature sets on predictive performance, and a detailed report on the profitability of the end-to-end simulation. Chapter 5, Conclusions, summarizes the key achievements of the project in relation to the initial research questions, discusses the limitations encountered during the study, and proposes directions for future work.

# Chapter 2: Background

This chapter will go over the multidisciplinary approach of this project. The following sections will cover fundamental knowledge required to understand the core technological and theoretical components . These components are drawn from the fields of deep learning, natural language processing, quantitative finance, and statistical inference.

After laying the groundwork on which this project is built, the later sections will explain the main objectives and the research questions answered throughout the implementation and experimentation of this dissertation.

## 2.1 Framing the Stock Prediction Task

Predicting stock market movements can be framed in several ways, each with distinct practical implications for a trading system. The choice of task fundamentally dictates the model architecture, evaluation metrics, and how the model's output is translated into an actionable trading decision. The primary prediction tasks in this domain are:

- **Price Prediction (Regression):** The approach here is to predict the exact closing price of a stock for the next trading day. While conceptually simple, this is an extremely difficult regression task due to the market's high volatility and non-stationary nature. Models are optimized to minimize price-distance errors (e.g., Mean Squared Error), which does not always align with maximizing profitability, as a small price error can still lead to a wrong directional decision.
- **Movement Prediction (Classification):** A model can also be constructed as a binary classification task: will the stock price go up or down? This simplifies the problem by focusing on directional accuracy, which is the most critical factor for profitability in many trading strategies.
- **Return Prediction (Regression):** A variation of price prediction is to predict the percentage return or the log magnitude of the change of a stock. This task is often preferred as returns are more stationary than raw prices. The output, a continuous value, can be used to estimate both the direction and the potential magnitude of a move.

For this project, we focus on the **Movement Prediction (Classification)** and **Return Prediction (Regression)** tasks. The primary goal is classification, as it directly informs the binary decision to buy or abstain from trading. The regression task serves as an alternative approach where the predicted return is converted into a directional signal, allowing for a direct comparison of which framing yields better practical results. This dual-task evaluation allows us to determine whether predicting the magnitude of a move adds valuable information or simply introduces unnecessary complexity.

### 2.1.1 Data Modalities and Model Selection

The data used to tackle these prediction tasks comes in two distinct forms, each requiring a specialized type of deep learning model:

1. **Sequential Time-Series Data:** The historical stock prices and their derived technical indicators form a sequence where the order of data points is critical. Past values are used to predict future ones, making this a classic time-series forecasting problem. **Long Short-Term Memory (LSTM) networks** are exceptionally well-suited for this type of data. Their architecture is explicitly designed with memory cells and gates to capture long-range dependencies and temporal patterns, which are fundamental to technical analysis.

2. **Unstructured Textual Data:** Social media posts, such as tweets, are unstructured sequences of words. The primary challenge is to extract semantic meaning, sentiment, and context from this raw text. **Bidirectional Encoder Representations from Transformers (BERT)** and similar transformer-based models are the state-of-the-art for such Natural Language Processing (NLP) tasks. Pre-trained on vast text corpora, BERT excels at understanding linguistic nuances and context, allowing it to transform a noisy tweet into a structured set of features (e.g., sentiment scores or stance classification) that can be integrated with the time-series data.

This project will take advantage of LSTMs for temporal data and BERT for the semantic, textual data. By using this dual approach, we aim to create a more comprehensive view of the market than either data source could provide alone.

## 2.2 Deep Learning: long short-term memory (LSTM)

Time series prediction is a challenging area of Machine Learning and Deep Learning ([Lai et al., 2018]; [Lim and Zohren, 2021]). Traditional recurrent neural networks (RNNs) can theoretically keep track of dependencies and relationships during training trough the use of backpropagation, where the RNN calculates the gradient of the loss function with respect to its weights one layer at a time, iterating backwards till the calculation reaches the first layer; the RNN would use this calculation to further adjust its weights and achieve a lower loss. By understanding 'learning' as an optimization problem ([Rumelhart et al., 1986]), backpropagation became a commonly used tool for traditional RNNs in deep learning.

However, as neural networks grew in size and depth, the nature of the backpropagation process introduced a significant hurdle when training recurrent neural networks([Hochreiter, 1991]). In the vanishing gradient problem, small number multiplication makes the gradients in the backpropagation calculations increasingly small, exponentially approaching zero, causing the training to slow down or even halt completely. This problem of RNNs became a research area that led to several advancements([Hochreiter and Schmidhuber, 1997]; [Kolen and Kremer, 2001]).

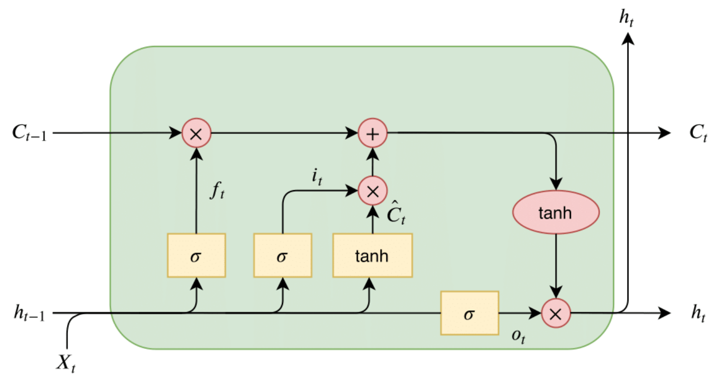

Figure 2.1: LSTM Cell Diagram

Introduced in 1997 ([Hochreiter and Schmidhuber, 1997]), long short-term memory (LSTM) neural networks were proposed as a solution to the vanishing gradient problem. By employing a sophisticated system of cell states and gates, LSTM networks can keep track of dependencies and relationships in long sequences of data without running into the vanishing gradient problem, which makes them effective for time series forecasting and prediction.

LSTM neural networks are at the core of the system developed for this project. The prediction models used in both the testing and the final simulation pipeline are used to run either classification or regression models. The models are trained on a large section of the composite dataset and finetuned to create predictions for the remaining of the data, which will be used for evaluation of performance and the investment simulations.

## 2.3 Natural Language Processing: BERT

Since this project involves the analysis of large datasets containing text messages (the tweet dataset), there is a need to understand how the text classification tools that will be used on this project came about, and how the field of Natural Language Processing (NLP) has progressed up to this point.

Early work in the language processing field was heavily based on pattern and rule matching, trying to encode linguistic rules into early computing systems. Works in machine intelligence ([Turing, 1950]) and linguistics ([Chomsky, 1957]) defined the approaches taken to try an emulate language tasks such as conversational agents and machine translation. An example that displays the culmination of this approaches into a computer chat-bot is ELIZA ([Weizenbaum, 1966]), groundbreaking for the time but is considered rudimentary by today's standards.

When the limitations of rule based systems became apparent, the field moved on into statistical approaches([Brown et al., 1990]; [Manning and Schütze, 1999]). N-gram and Hidden Markov models (HMVs) became the state of the art techniques for tasks such as part-of-speech tagging and machine translation tasks.

As the field continued to advance the introduction of early deep learning techniques allowed for further advancement on the field of NLP. The introduction of the first neural language models ([Bengio et al., 2003]), the introduction and

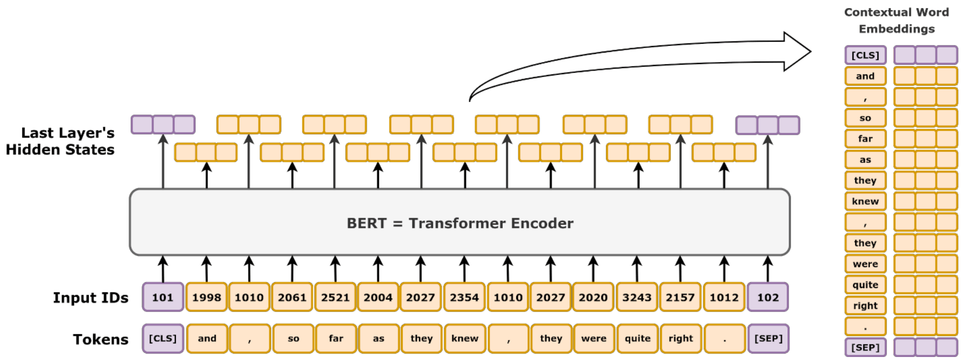
Contextual Word Embeddings
Last Layer's Hidden States
BERT = Transformer Encoder
Input IDs
Tokens
101 1998 1010 2061 2521 2004 2027 2354 1010 2027 2020 3243 2157 1012 102
[CLS] and , so far as they knew , they were quite right . [SEP]

Figure 2.2: High-level schematic diagram of BERT ([Godoy, 2022]).

popularization of word embeddings ([Mikolov et al., 2013]) and the usage of RNN and LSTM for tasks such as machine translation ([Sutskever et al., 2014]) as it became the state-of-the-art before the introduction of transformers.

This historical overview leads to the introduction of BERT. Shortly after the introduction of the transformer architecture in 2017([Vaswani et al., 2017]), a new model that built upon this architecture was introduced. BERT, which stands for Bidirectional Encoder Representations from Transformers, significantly advanced the ability of computers to process human language and generate text. Introduced in 2018 ([Devlin et al., 2018]), BERT brought about significant improvements in a variety of NLP tasks such as text classification. As a pre-trained model, it has already been trained on a large corpus of text. This allows the model to be used directly or to be fine-tuned on smaller, task-specific datasets.

To capture semantic and contextual information from tweets, the system employs several pre-trained BERT-based language models. Each model is fine-tuned for text classification (e.g., sentiment, stance, or emotion). The resulting classification outputs (such as probabilities or embeddings) are then used as features that serve as inputs to subsequent deep learning tasks.

This approach leverages the linguistic knowledge already encoded in large pre-trained models, while transforming raw text into structured, high-level representations suitable for downstream models. Combining multiple BERT variants enables the generation of diverse feature sets, which can improve robustness and performance compared to relying on a single model.

## 2.4 Staking Strategies: Kelly Criterion

For any investor or gambler, determining the optimal amount of capital to allocate to an opportunity is a fundamental aspect of a successful strategy. Go too small, and potential returns are needlessly diminished; risk too much, and the threat of ruin looms large. The Kelly Criterion offers a mathematical solution to this dilemma, providing a formula for sizing positions to maximize the long-term growth rate of assets. Developed by John L. Kelly Jr. in 1956, the criterion provides a disciplined framework that balances the probability of success with the potential returns, a tool that has been embraced in fields ranging from sports betting to sophisticated investment management.

Developed originally as a way to analyze long-distance telephone signal noise

([Kelly, 1956]), the Kelly Criterion's potential applications for gambling and investment were quickly recognized. The general form for the Kelly Criterion, assuming a loss of the full wager with every bet lost, is defined as:

$$f^* = p - \frac{q}{b} = p - \frac{1-p}{b} \tag{2.1}$$

where:

- $f^*$ is the fraction of the current bankroll to wager.
- $p$ is the probability of a win.
- $q = 1 - p$ is the probability of a loss.
- $b$ is the proportion of the bet gained with a win.

This general form of the formula is intuitive to understand and was rigorously proven in its original paper. However, the original formulation did not account for the specifics of investment returns; to address this, different formulations were made to account for continuous returns and other factors ([Thorp, 2008]; [Rotando and Thorp, 1992]). For the purpose of investment, the formula for the Kelly Criterion can be adapted as:

$$f^* = \frac{p}{l} - \frac{q}{g} \tag{2.2}$$

where:

- $f^*$ is the fraction of the assets to apply to the stock or security.
- $p$ is the probability that the investment increases in value.
- $q$ is the probability that the investment decreases in value.
- $g$ is the fraction that is gained in a positive outcome.

There are two main problems with using the Kelly Criterion for investments with probabilities informed by a deep learning model. First, even if we tune our LSTM network to output a value between zero and one (which would mimic traditional probability), this is not an actual probability, just the confidence estimation from the model of wether of not the stock will go up or down based on the patterns learned from training data and the input of today. Second, for the Kelly Criterion to hold true in investment scenarios, we must know how much the stock being traded will go up or down. This information is almost always unknown and it can only be estimated based on historical data. In later sections, the strategies and tools chosen to mitigate this limitations will be further explained.

## 2.5 Statistical Inference: Isotonic Regression

As mentioned in the deep learning overview of this report, even if we tune a deep learning model to output results of model confidence as a number between zero and one in accordance with Kolmogorov Axioms of probability ([von Plato,

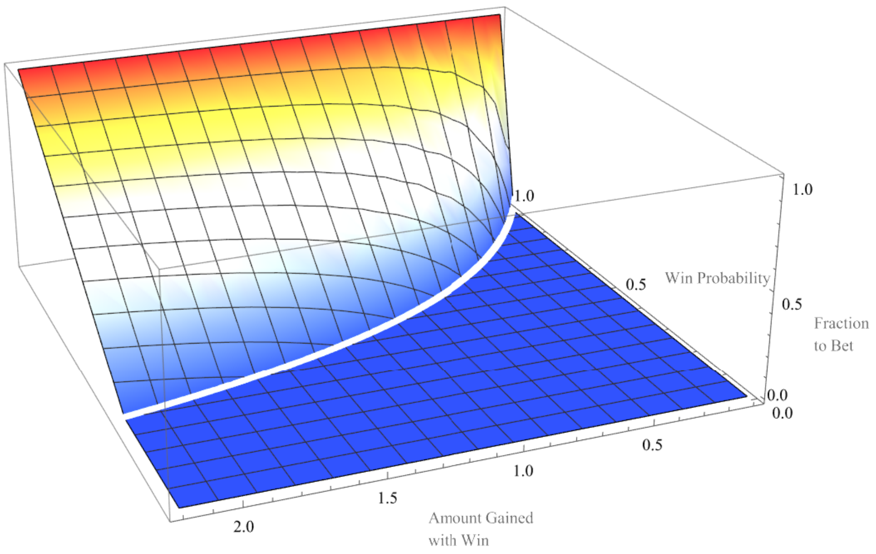

Figure 2.3: 3D figure representing the optimal Kelly bet size on the vertical axis ([Commons, 2025]).

1933]), we are not actually getting the probability of the predicted stock to go up or down, the model is simply giving its degree of confidence based on its training data and the learned patterns encoded in its weights. This becomes problematic since the Kelly Criterion requires the probability to be known in order to maximize the expected returns in the long run. Even assuming a clean and sufficiently large dataset and an optimal model, it is unrealistic to expect its predictions to be analogous to true probability.

In order to address this problem, this project makes use of isotonic regression as an intermediate step between the classifier score and the staking calculation. Isotonic Regression was introduced alongside its cornerstone algorithm, the Pool Adjacent Violators Algorithm (PAVA). Both introduced in the same paper ([Ayer et al., 1955]), they became a tool used in many fields for their flexibility.

Functionally, isotonic regression fits a line into a sequence of data points such that the fitted line is non-decreasing (or non-increasing) at all points, while remaining as close to the observations as possible ([Fielding, 1974]). The strength of the technique lies in its ability to correct monotonic distortions as well as its non-linearity.

In 2002, it was first shown that probability calibration using isotonic regression could be used for classification problems in the context of machine learning with satisfactory results ([Zadrozny and Elkan, 2002]). The paper clearly articulates how isotonic regression works well for both binary and multiclass classification problems as well as showing several advantages compared to other calibration methods, particularly its non-parametric nature. The experimental results ultimately showed the benefits of using calibrated probabilities instead of taking the classifiers scores at face value, since these are more often than not, uncalibrated.

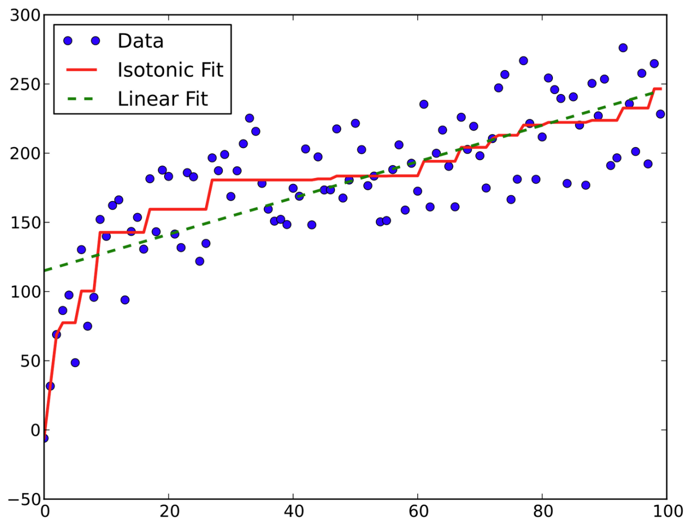

Figure 2.4: Isotonic regression (solid red line) compared to linear regression on the same data, both fit to minimize the mean squared error ([Commons, 2020]).

## 2.6 Project Aims

This chapter outlines the primary aims of the research project. At its core, this project seeks to investigate whether integrating Natural Language Processing (NLP) features derived from social media with traditional financial data can enhance the accuracy of stock movement prediction. To provide a clear path for this investigation, the project is structured around a set of specific research objectives and seeks to provide an answer to a set of research questions. The objectives detail the technical steps from feature engineering and model evaluation to the development of an investment simulation pipeline, while the research questions target the central hypotheses regarding the utility of NLP features and the practical viability of the proposed system.

### 2.6.1 Research Objectives

- Objective 1: Stocks and tweets feature engineering
  - Extract technical indicators (e.g., moving averages, RSI, MACD, volatility) from stock price data.
  - Perform sentiment analysis, stance detection and emotion analysis on tweets related to the stocks.
  - Align and aggregate stock features with tweet-derived features on a common time scale.
  - Run tests using combinations of features to understand the impact of the engineered features
- Objective 2: Model and features testing and evaluation
  - Design an environment to test different deep learning architectures.
  - Compare results to baselines such as ARIMA and previous other relevant stock prediction papers ([Xu and Cohen, 2018]; [Hu et al., 2018])
  - Evaluate predictive performance using metrics such as accuracy, F1-score, RMSE, and Matthews coefficient.
- Objective 3: Creation of an investment simulation pipeline

  - Design a framework to simulate trading strategies based on model predictions.
  - Implement a staking strategy using the Kelly Criterion and other risk management techniques.
  - Assess profitability and risk through performance metrics (e.g., cumulative returns, Sharpe ratio).

### 2.6.2 Research Questions

- How does the integration of NLP-derived features—specifically financial stance, general sentiment, and emotion—impact the directional accuracy of a BiLSTM model for next-day stock price prediction compared to a model using only historical price and volume data?
- Among different NLP features, which provides the most utility for stock movement prediction? Specifically, how does a domain-specific ”stance” feature compare to general-purpose ”sentiment” and multi-class ”emotion” features when combined with technical indicators?
- Can an end-to-end trading system, leveraging calibrated BiLSTM predictions and a Fractional Kelly Criterion for capital allocation, achieve positive returns in a simulated environment? How does its performance compare across models with and without NLP features?
- Is the predictive performance of the BiLSTM models consistent across different market sectors and individual stocks? Are there specific tickers or sectors where the models consistently succeed or fail?

# Chapter 3: Design & Implementation

This chapter covers the design decisions, methodological approach and implementation of the end-to-end trading simulator, as well as the necessary pre-processing, feature engineering and model evaluation stages. First, we will describe the design behind its modular components. Then, we will cover the data acquisition, pre-processing and feature engineering stages. Lastly, this chapter will describe the implementation of the evaluation program and the investment simulation program that will test the real-world viability of the proposed strategy, including specific implementation details where necessary.

## 3.1 Design Rationale & Architecture

All the sections developed for the project are designed with a modular, multi-stage approach to allow flexibility for scenarios where multiple models with different configurations need to be tested in succession. In principle, each component should take a given input, do its independent operations and produce an output that can be taken as is by the next module in the chain.

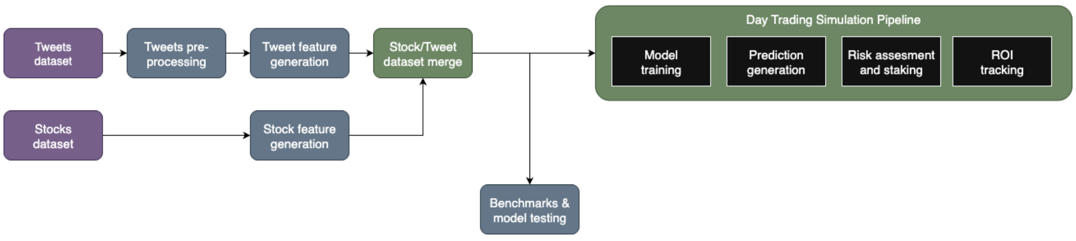

Figure 3.1: High level logic of the System's architecture.

This separation is a deliberate design choice. The **benchmarking and model testing pipeline** is an offline framework designed for rapid, systematic experimentation. It allows for the evaluation of different feature sets, data processing techniques, and model architectures without having to make any significant changes for each experiment. This is crucial for iterating on the predictive component of the system efficiently.

Conversely, the **Day Trading Simulation Pipeline** is a system designed to closely mimic a real-world trading environment. It operates chronologically, using only the data available up to a given day to make decisions, thereby avoiding lookahead bias. This pipeline integrates the best-performing model from the benchmarking stage with risk management and portfolio construction logic to provide a robust assessment of the strategy's profitability. The modularity ensures that the predictive model can be updated or replaced without altering the core simulation and risk management logic.

Both modules (evaluation and simulation) are designed to take the same parquet files containing the stock and tweet data already pre-processed. And all configuration are declared on global parameters declared on a single space in the code, making experimentation and iteration a straightforward process.

## 3.2 Data Acquisition and Pre-processing

### 3.2.1 Dataset Selection

All the data used in both the evaluation and simulation stages of the project comes from the Stocknet dataset, originally introduced in the work of Xu and Cohen [2018].

The Stocknet dataset provides a comprehensive benchmark for the task of stock movement prediction by integrating both financial market data and social media signals. It covers a two-year period from January 1, 2014 to January 1, 2016, and includes a total of 88 target stocks. The selection procedure follows the approach outlined by Xu and Cohen [2018]: all eight stocks in the Conglomerates sector are included, while in each of the remaining eight sectors the ten largest firms by market capitalization are selected. This ensures both sectoral diversity and sufficient representation of highly traded firms. The complete list of stocks is reported in the appendix of the original paper and on the github repository where the dataset is hosted.

The dataset consists of two main components: *tweet data*, collected from Twitter, and *price data*, retrieved from Yahoo Finance. For both components, the repository provides raw and preprocessed versions, organized by stock ticker.

- **Tweet data:**
  - Raw format: JSON objects following the standard Twitter API structure.
  - Preprocessed format: JSON files containing the keys `text`, `user_id_str`, and `created_at`.
- **Price data:**
  - Raw format: CSV files with daily entries including date, open, high, low, close, adjusted close prices, and volume.
  - Preprocessed format: TXT files with daily entries containing the date, percentage movement, normalized open, high, low, and close prices, as well as volume.

By combining market signals with large-scale social media data, the Stocknet dataset enables research on multimodal approaches to financial prediction. Its design facilitates the study of temporally-dependent relationships between unstructured text and structured price series, making it particularly suitable for the purposes of this project.

### 3.2.2 Dataset Pre-processing

The initial Stocknet dataset, while comprehensive, requires significant preprocessing to transform its raw components into a format suitable for machine learning models. This phase involves loading heterogeneous data sources, cleaning textual data to remove noise, and structuring the final datasets for subsequent feature engineering and model training stages. The process is executed across several sequential notebooks.

First, the raw price and tweet data are loaded and consolidated. The price data, originally stored in separate CSV files for each of the 88 tickers, is read into memory, combined into a single tidy pandas DataFrame, and then saved in the efficient Parquet format as `stock_prices.parquet`. Similarly, the tweet data, distributed across a nested directory structure of JSON files, is parsed to extract key fields—namely the ticker, tweet text, creation timestamp, and user ID. This consolidated tweet data is then saved as `stock_tweets.parquet`.

The core of the pre-processing work focuses on sanitizing the tweet text. An initial cleaning function is applied to each tweet to eliminate common social media artifacts that provide little semantic value for sentiment analysis. This step removes URLs, user mentions (e.g., `@username`), hashtags, stock cashtags (e.g., `$TICKER`), and various punctuation marks. The regular expression pattern used for this initial sanitization is shown below.

Listing 3.1: Initial Regex Pattern for Tweet Sanitization

```python
char_patterns = re.compile(
    'http[s]?://(?:[a-zA-Z]|[0-9]|[[a-zA-Z]+|
    @[a-zA-Z]+|[,.^_$*%-;!?:]'
    )

for i in range(len(tweet_df["text"])):
    tweet_df["text"][i] = char_patterns.sub('',
        tweet_df["text"][i])
```

Following the initial cleaning, two distinct data structures are prepared for different modeling approaches. The first approach involves creating a daily summary of market sentiment. To achieve this, all individual tweets for a given stock on a specific day are aggregated by concatenating their text into a single document. This produces a DataFrame where each row corresponds to one stock on one trading day, containing all associated tweet text.

A second, more refined cleaning step is then applied to both the aggregated and non-aggregated tweet data. This step removes words and abbreviations that could act as trivial predictors or introduce noise, such as the stock's own ticker symbol, retweet indicators ('rt'), and common filler words like 'inc' and 'ie'. This ensures that the model focuses on the semantic content of the tweets rather than superficial patterns.

Listing 3.2: Refined Regex for Removing Tickers and Noise

```python
for i in range(len(tweet_df["text"])):
    # The ticker is dynamically inserted into the pattern
    ticker = tweet_df["ticker"][i]

    more_patterns = re.compile(
        r'(\s*)({})(\s*)|(\s*)ATUSER(\s*)|(\s*)AT_USER(\s*)|
        (\s*)URL(\s*)|(\s*)rt(\s*)|(\s*)inc(\s*)|
        (\s*)ie(\s*)'.format(ticker), flags=re.IGNORECASE
        )

    tweet_df["text"][i] = more_patterns.sub('',
        tweet_df["text"][i])
```

This multi-stage pre-processing pipeline results in two final, cleaned datasets ready for analysis:

- **`merged_stock_tweet.parquet`**: Contains tweets aggregated on a daily per-stock basis.
- **`stock_tweet_no_merge.parquet`**: Contains individual tweets, cleaned but not aggregated.

These clean and structured datasets serve as the foundation for all subsequent feature extraction and model training tasks.

## 3.3 Feature Engineering

### 3.3.1 Technical Indicator Generation from Stock Data

In addition to social media signals, a comprehensive set of technical indicators is engineered from the historical price data. These features are widely used in quantitative trading to capture market dynamics such as trend, momentum, volatility, and volume, providing the models with a quantitative basis for understanding price action ([Murphy, 1999]).

The process leverages the `pandas_ta` library. To ensure chronological integrity and prevent data leakage across assets, the dataset is grouped by each unique ticker, and the feature engineering functions are applied independently to each stock's time series. The selection of indicators is designed to provide a multi-faceted view of the market:

- **Trend Indicators:** Exponential Moving Averages (EMAs) with periods of 12, 26, and 50 days are calculated to identify short, medium, and long-term trends. The Moving Average Convergence Divergence (MACD) indicator is also included, providing the MACD line, signal line, and histogram.
- **Momentum Indicators:** The Relative Strength Index (RSI) with a 14-day period is used to measure the speed and change of price movements, helping to identify overbought or oversold conditions. The Stochastic RSI is also computed to provide a more sensitive momentum reading.
- **Volatility Indicators:** The Average True Range (ATR) measures market volatility, while Bollinger Bands (with a 20-day period and 2 standard deviations) create a dynamic envelope around the price, indicating periods of high or low volatility.
- **Volume Indicators:** On-Balance Volume (OBV) is used to relate price and volume, providing insights into the strength of a price trend based on trading volume.

Listing 3.3: Function for Calculating Technical Indicators

```python
def apply_ta_indicators(df_group):
    df_group.set_index(pd.DatetimeIndex(df_group['date']),
        inplace=True)
    # Trend indicators
    df_group.ta.ema(length=12, append=True)
    df_group.ta.ema(length=26, append=True)
    df_group.ta.ema(length=50, append=True)
    df_group.ta.macd(fast=12, slow=26, signal=9, append=True)


    # Momentum indicators
    df_group.ta.rsi(length=14, append=True)
    df_group.ta.stochrsi(length=14, append=True)

    # Volatility indicators
    df_group.ta.atr(length=14, append=True)
    bb = ta.bbands(df_group['close'], length=20, std=2)
    df_group['BB_upper'] = bb['BBU_20_2.0']
    df_group['BB_middle'] = bb['BBM_20_2.0']
    df_group['BB_lower'] = bb['BBL_20_2.0']

    # Volume indicators
    df_group.ta.obv(append=True)
    return df_group.reset_index(drop=True)

master_df =
   master_df.groupby('ticker').apply(apply_ta_indicators)
```

### 3.3.2 NLP Feature Extraction from Social Media

After pre-processing, the raw text of the tweets is transformed into a set of quantitative features suitable for machine learning. This process leverages several pre-trained transformer models from the HuggingFace library to extract nuanced signals related to sentiment, emotion, and financial stance. Each feature is calculated on a per-tweet basis before being aggregated to a daily feature.

First, a general sentiment score is assigned to each tweet using the `nlptown/bert-base-multilingual-uncased-sentiment` model. This model classifies text on an ordinal scale from 1 (very negative) to 5 (very positive), with the score determined by taking the argmax of the model's output logits. The function in Listing 3.4 demonstrates this per-tweet calculation. To create a daily signal, the final sentiment score for a given stock is the **average** of all individual tweet scores from that day.

Listing 3.4: Function for Calculating Sentiment Score

```
def cal_sentiment_score(text):
    # Tokenize and move tensors to GPU
    inputs = tokenizer.encode(text, return_tensors='pt',
       truncation=True, max_length=512).to(device)

    with torch.no_grad():
        outputs = bert_model(inputs)

    # The sentiment is the index of the highest logit, plus one
    sentiment = int(torch.argmax(outputs.logits)) + 1
    return sentiment
```

To capture a more detailed emotional context, a second model, `j-hartmann/emotion-english-distilroberta-base`, is employed. This classifier analyzes each tweet and returns a probability distribution across seven distinct emotional categories. These categories—anger, disgust, fear, joy, neutral, sadness, and surprise—are based on established emotion recognition frameworks in NLP research, such as that proposed by Saravia et al. [2018]. This process generates seven feature columns where the daily value for each emotion is the **sum** of all corresponding confidence scores from tweets on that day, aggregating the total emotional "charge" for a stock.

Finally, to incorporate a domain-specific signal, a financial stance detection model, `zhayunduo/roberta-base-stocktwits-finetuned`, is used. This is crucial for distinguishing general sentiment from a direct opinion on a stock's future performance. The model, specifically fine-tuned on data from the StockTwits social network, classifies each tweet as either 'Positive' (bullish) or 'Negative' (bearish) and provides a confidence score, as shown in Listing 3.5. The daily stance is then captured in two columns representing the total **count** of tweets classified as 'Positive' and 'Negative', respectively.

Listing 3.5: Function for Financial Stance Detection

```python
def stance_score(text):
    # The pipeline handles tokenization and model inference
    result = nlp(text)
    label = result[0]['label']
    score = result[0]['score']
    return label, score
```

This multi-faceted feature engineering approach produces a rich daily summary of social media signals, which serves as a primary input for the stock movement prediction models. The final enriched dataset is saved as `stock_tweets_withsentiment_withemotion_withstance_nomerge.parquet`.

## 3.4 Final Dataset

The final step in the data preparation pipeline is the construction of a master DataFrame that consolidates all engineered features from the different data sources: historical stock prices, aggregated daily tweet metrics, and company metadata. This unified dataset serves as the single source of truth for training and evaluating all predictive models.

The process begins by merging the daily stock price data with the daily aggregated tweet features. A left merge is performed on the stock price table, ensuring that all trading days are preserved, even those for which no tweet data is available. For such days, the tweet-derived features (e.g., sentiment, emotion scores, stance counts) are imputed with a value of zero, representing a neutral or non-existent social media signal. Subsequently, static company information, such as sector and company name, is merged in based on the stock ticker.

The resulting master DataFrame is a rich, high-dimensional dataset where each row corresponds to a single trading day for a specific stock. It contains all the necessary information for building multimodal prediction models, combining fundamental company attributes, market-based technical indicators, and social media sentiment.

The columns of the final master DataFrame are described below:

**`date`**: The date of the trading session.

**`open, high, low, close, adj_close`:** Standard daily Open, High, Low, Close, and Adjusted Close prices.

**`volume`:** The number of shares traded on that day.

**`ticker`:** The stock's ticker symbol.

**`company`:** The stock's company name.

**`sector`:** The industrial sector to which the company belongs.

**`stance_positive, stance_negative`:** The total daily count of tweets classified as bullish and bearish, respectively.

**`sentiment`:** The average sentiment score (from 1 to 5) of all tweets for that day.

**`emotion_`:** Seven columns (`emotion_anger`, `emotion_disgust`, etc.) containing the summed daily scores for each emotion.

**`MACD, MACDh*, MACDs*`:** The MACD line, histogram, and signal line.

**`RSI_14`:** The 14-day Relative Strength Index.

**`STOCHRSIk*, STOCHRSId*`:** The K and D lines of the Stochastic RSI.

**`ATRr_14`:** The 14-day Average True Range.

**`BB_upper, BB_middle, BB_lower`:** The upper, middle, and lower Bollinger Bands.

**`OBV`:** The On-Balance Volume indicator.

**`EMA_12, EMA_26, EMA_50`:** Exponential Moving Averages for 12, 26, and 50-day periods.

## 3.5 Model Evaluation and Metric Tracking

To facilitate a rigorous and reproducible evaluation of predictive models, a custom Python class, StockPredictionPipeline, was created. This class serves as a comprehensive framework that encapsulates the entire experimental workflow, from data processing to model training, evaluation, and results aggregation. Its primary design goal is to enable the systematic benchmarking of different model architectures and problem formulations across a diverse portfolio of stocks.

The pipeline's flexibility is centered around its initialization, where the user defines the core parameters of the experiment. These inputs include the dataset, the feature set, the model architecture (e.g., LSTM, BiLSTM), the time-series sequence length, and, most importantly, the problem_type. This parameter explicitly configures the pipeline to perform either regression (predicting the magnitude of the next day's price change) or classification (predicting the direction of the next day's price change).

Listing 3.6: Feature Set and Pipeline Initialization

```python
feature_columns = [
        'open', 'high', 'low', 'close', 'volume',
        'stance_positive', 'stance_negative',
        'EMA_12', 'EMA_26', 'EMA_50', 'MACD_12_26_9',
           'MACDh_12_26_9',
        'MACDs_12_26_9', 'RSI_14', 'ATRr_14',
           'STOCHRSIk_14_14_3_3',
        'STOCHRSId_14_14_3_3', 'BB_upper', 'BB_middle',
           'BB_lower', 'OBV'
        ]

#The pipeline can be initialized for either regression or
   classification

pipeline = StockPredictionPipeline(
                                df=master_df,
                                feature_columns=feature_columns,
                                model_type='BiLSTM',
                                sequence_length=12,
                                problem_type='regression' # or
                                   'classification'
                                )
```

A distinct model is trained for each company, allowing the network to capture the unique dynamics of individual stocks. The model architecture is constructed dynamically based on the chosen problem_type. While the core recurrent layers (e.g., BiLSTM) and regularization layers (Batch Normalization, Dropout) remain consistent, the output layer and compilation settings are tailored specifically to the task.

For regression, the model employs a final Dense layer with a linear activation function and is compiled with the Huber loss function to robustly predict continuous log returns.

For classification, the model uses a Dense layer with a sigmoid activation function to output a probability and is compiled with binary cross-entropy loss for the directional prediction task.

Listing 3.7: Task-Specific Model Architecture

```python
def build_model(self, input_shape):
    # ... (Shared RNN, Batch Norm, and Dense layers) ...

    # Problem-specific output layer and compilation
    if self.problem_type == 'regression':
        model.add(layers.Dense(1, activation='linear'))
        model.compile(
            optimizer=keras.optimizers.Adam(learning_rate=0.001),
            loss='huber',
            metrics=['mae', 'mse']
            )
    else:  # classification
        model.add(layers.Dense(1, activation='sigmoid'))
        model.compile(
            optimizer=keras.optimizers.Adam(learning_rate=0.001),
            loss='binary_crossentropy',
            metrics=['accuracy', 'precision', 'recall']
            )
return model
```

The evaluation logic is equally task-aware. The pipeline calculates a comprehensive suite of metrics appropriate for the chosen problem type, ensuring a complete assessment of performance.

When a regression model is trained, it calculates standard regression metrics (MSE, MAE, $R^2$). Crucially, it also evaluates the model's directional capability by converting the predicted log returns into binary up/down signals and computing corresponding classification metrics (Accuracy, MCC, F1 Score).

When a classification model is trained, it directly calculates the full set of classification metrics from the model's probabilistic output. In this mode, the regression-specific metrics are recorded as not applicable (NaN).

This dual-metric system ensures that regardless of the primary task, the model's ability to predict market direction—a key indicator of practical utility—is always evaluated. All performance metrics, along with metadata like company, sector, and sample sizes, are captured for each individual stock.

Listing 3.8: Comprehensive Metric Collection

```
result = {
        'company': company_name,
        'sector': sector,
        'model_type': self.model_type,
        'problem_type': self.problem_type,
        # Regression metrics (NaN if classification)
        'mse': mse,
        'mae': mae,
        'r2': r2,
        # Classification / Directional metrics
        'mcc': mcc,
        'f1': f1,
        'precision': precision,
        'recall': recall,
        'directional_accuracy': directional_accuracy,
        # Data info
        'n_samples': len(X),
        'test_samples': len(X_test)
        }
```

Upon processing all companies, the pipeline aggregates the individual results into a final Pandas DataFrame. An analysis function then computes overall performance statistics (mean and standard deviation) for each metric, provides performance breakdowns by market sector, and identifies top-performing models. This aggregated data is saved to a CSV file, creating a persistent and detailed record of the experiment. This structured methodology allows for a direct and fair comparison between the two distinct approaches: predicting the magnitude of price changes versus predicting the direction.

## 3.6 Investment Simulation Pipeline

The core of this project is an event-driven simulation system designed to recreate a realistic trading environment. This pipeline moves forward day-by-day, making dynamic decisions about model training, stock selection, and capital allocation based on historical data available up to that point. This approach avoids lookahead bias and provides a robust evaluation of the strategy's real-world viability. The entire system is governed by a set of key configurable parameters that define its behavior.

Listing 3.9: Global Configuration for the Simulation

```
MODEL_SAVE_PATH = "trained_models/"
INITIAL_TRAINING_DAYS = 1100 # Days of data before simulation
    begins
KELLY_FRACTION = 0.05 # Fraction of the Kelly bet to take
SECTOR_CONFIDENCE_THRESHOLD = 0.40 # Min avg sector
    probability to consider
RETRAIN_INTERVAL = 200 # Days before a model is flagged for
    retraining
MAX_DAY_GAP = 5 # Max gap to consider a data period contiguous
```

The pipeline is architected around several key functions that handle data preparation, model training, prediction, and portfolio management.

### 3.6.1 Dynamic Data Handling and Robustness

To handle the non-continuous nature of real-world financial data (due to weekends, holidays, or data gaps), the system first identifies contiguous trading periods for each stock. This ensures that the time-series sequences fed into the models represent uninterrupted data, minimizing the chances of models learning from artificial jumps. Furthermore, the sequence length for each model is not fixed but is calculated dynamically based on the amount of historical data available for each specific company. This adaptive approach allows the system to build more robust models for stocks with longer histories while still being able to model those with less available data.

Listing 3.10: Handling of Contiguous Data Periods

```
def create_contiguous_sequences(data: np.ndarray,
                                targets: np.ndarray,
                                contiguous_periods: list,
                                sequence_length: int):

    X, y = [], []

    for start_idx, end_idx in contiguous_periods:
        # ... logic to skip periods shorter than
            sequence_length ...

        for i in range(start_idx + sequence_length, end_idx +
            1):
             X.append(data[i-sequence_length:i])
             y.append(targets[i])

    return np.array(X), np.array(y)
```

### 3.6.2 Per-Asset Modeling and Probability Calibration

A separate LSTM model is trained for each individual stock. The task is framed as a binary classification problem: predicting whether the next day's closing price will be higher than the current day's close. A critical step in the training process is the calibration of model outputs. The raw output of a neural network's sigmoid function, while bounded between 0 and 1, does not represent a true, reliable probability. To address this, an Isotonic Regression model is fitted on the validation set predictions. This post-processing step calibrates the model's output, transforming it into more accurate probabilities that can be confidently used for quantitative decision-making, such as in the Kelly Criterion.

For each asset, the system saves four essential artifacts: the trained Keras model, the fitted data scaler, the Isotonic Regression calibrator, and the dynamically calculated sequence length.

Listing 3.11: Model Training and Isotonic Calibration

```python
def train_company_models(company_data_df: pd.DataFrame,
                         ticker: str, ...):

    # ... data splitting and scaling ...

    # Define and train the LSTM model
    model = keras.Sequential([
        layers.Input(shape=(X_train_scaled.shape[1],
           X_train_scaled.shape[2])),
        layers.Bidirectional(layers.LSTM(128,
           return_sequences=True, ...)),
        layers.Bidirectional(layers.LSTM(64,
           return_sequences=False, ...)),
        layers.BatchNormalization(),
        layers.Dense(32, activation='relu'),
        layers.Dropout(0.3),
        layers.Dense(1, activation='sigmoid')
        ])

    model.compile(loss='binary_crossentropy', ...)
    model.fit(X_train_scaled, y_train, ...)

    # Train the Isotonic Regression calibrator
    validation_predictions =
       model.predict(X_val_scaled).flatten()
    calibrator = IsotonicRegression(y_min=0.0, y_max=1.0,
       out_of_bounds='clip')
    calibrator.fit(validation_predictions, y_val)

    # Save all components
    model.save(f"{ticker}_lstm.keras")
    joblib.dump(calibrator, f"{ticker}_calibrator.pkl")
    joblib.dump(scaler, f"{ticker}_scaler.pkl")
    joblib.dump(sequence_length, f"{ticker}_seq_length.pkl")
```

### 3.6.3 Portfolio Construction and Capital Allocation

Each day within the simulation, the system generates a calibrated probability prediction for every stock using the trained model. These predictions are then fed into a portfolio construction module that implements a sector-based investment strategy combined with the Fractional Kelly Criterion for position sizing.

The process is as follows:

1. **Sector-Level Filtering:** Stocks are grouped by their respective sectors. If the average calibrated probability for a sector falls below the parameter `SECTOR_CONFIDENCE_THRESHOLD`, the entire sector is disregarded for that day.
2. **Best-in-Sector Selection:** For each qualifying sector, the single stock with the highest calibrated probability is selected as the investment candidate.
3. **Fractional Kelly Sizing:** The investment amount for each selected stock is determined using the Fractional Kelly Criterion. The optimal fraction of capital, $f$, is given by:
$$f = k \cdot \left(p - \frac{1-p}{b}\right),$$
where
   - $p$ is the calibrated probability of the stock's price increasing,
   - $b$ is the historical average payout ratio (win size divided by loss size), calculated once from the initial training data,
   - $k$ is the `KELLY_FRACTION`, a conservative scaling factor used to reduce risk.

This approach ensures diversification across sectors and employs a mathematically grounded method for both risk and capital management.

Listing 3.12: Portfolio Selection and Sizing Logic

```python
def select_and_size_portfolio(daily_predictions_df,
                              payout_map, ...):

    investment_decisions = []
    for sector, group in
        daily_predictions_df.groupby('sector'):
        # 1. Sector-Level Filtering
        if group['calibrated_prediction'].mean() <
            sector_threshold:
             continue


        # 2. Best-in-Sector Selection
        best_stock =
            group.loc[group['calibrated_prediction'].idxmax()]
        p = best_stock['calibrated_prediction']
        b = payout_map.get(best_stock['ticker'], 0)

        if b <= 0:
            continue

        # 3. Fractional Kelly Sizing
        kelly_percentage = p - ((1 - p) / b)
        if kelly_percentage > 0:
            investment_fraction = kelly_percentage *
               kelly_fraction
            investment_amount = total_capital *
               investment_fraction
            investment_decisions.append(...)

    return pd.DataFrame(investment_decisions)
```

### 3.6.4 Simulation Loop and Performance Tracking

The main simulation function orchestrates the entire process over a long historical period. It begins by training models on an initial `INITIAL_TRAINING_DAYS` of data. Then, it iterates through the remaining dates one by one. On each day, it:

1. Identifies models that require training or retraining (based on the parameter `RETRAIN_INTERVAL`).
2. Generates predictions for all available assets using the latest historical data.
3. Constructs the daily portfolio and determines investment sizes.
4. Calculates the profit or loss (PnL) from the previous day's investments based on the actual market outcome.
5. Updates the total capital.
6. Logs all daily activity, including capital levels, PnL, and specific trades made.

Finally, upon completion of the simulation, key performance metrics such as the Total Return on Investment (ROI) and the annualized Sharpe Ratio are calculated from the log to provide a comprehensive summary of the strategy's performance. ROI is a fundamental performance measure representing the total percentage gain on the initial capital ([Brealey et al., 2011]). The Sharpe Ratio, developed by William F. Sharpe, is a critical measure of risk-adjusted return, quantifying the excess return earned per unit of volatility ([Sharpe, 1966]).

## 3.7 Implementation Details

The entire system is implemented in Python 3, leveraging its extensive ecosystem of libraries for data science and machine learning. The experimental workflow is managed within standalone Jupyter Notebooks, which allows for a clear, step-by-step documentation of the data processing, modeling, and evaluation stages. For a potential real-world deployment, these notebooks could be automated using a workflow management tool like Papermill.

### 3.7.1 Libraries and Modules by Functionality

- **Data Handling**
  - **pandas (pd)** : Manage stock price series and tweet-derived tabular data.

- **numpy (np)** : Perform numerical operations and efficient array computations.
- **json** : Parse tweet datasets and save model configurations.
- **os** : Manage file paths and environment variables (e.g., suppress TensorFlow logs).
- **site** : Minimal usage for environment path management.

- **Feature Engineering**
  - **pandas_ta (ta)** : Generate technical indicators (RSI, MACD, etc.) from stock price data.
  - **re** : Clean and preprocess tweets using regular expressions.
- **Preprocessing and Calibration (sklearn)**
  - **preprocessing** : - `StandardScaler, MinMaxScaler, RobustScaler` : Normalize features before training.
    - `IsotonicRegression` : Calibrate predicted probabilities.
  - **utils** : - `compute_class_weight` : Handle class imbalance in financial classification tasks.
- **Model Training and Architecture**
  - **tensorflow (tf)**
    - **keras.models** : `Sequential, load_model` — Build and reload deep sequential models.
    - **keras.layers** : `LSTM, Dense, Dropout` — Core RNN and fully-connected layers.
    - **keras.callbacks** : `EarlyStopping, ReduceLROnPlateau` — Stabilize training and prevent overfitting.
    - **keras.regularizers** : `l2` — Apply weight penalties for generalization.
  - **torch** : Backend engine for HuggingFace Transformers; enables GPU-accelerated training.
- **Model Evaluation (sklearn)**
  - **metrics** : - `precision_score, recall_score, f1_score, matthews_corrcoef` : Classification evaluation.
    - `mean_squared_error, mean_absolute_error, r2_score` : Regression evaluation.
    - `confusion_matrix` : Assess classification performance visually.
  - **model_selection** : - `train_test_split` : Split data into training and testing sets.
    - `TimeSeriesSplit` : Ensure chronological integrity in time series validation.
- **NLP and Transformers (HuggingFace)**
  - `AutoTokenizer, AutoModelForSequenceClassification` : General-purpose pretrained models for tweet classification.
  - `RobertaForSequenceClassification, RobertaTokenizer` : Fine-tuned RoBERTa for financial sentiment analysis.
  - `pipeline` : Rapid prototyping of classification workflows.
- **Visualization**
  - **matplotlib.pyplot (plt)** : Create plots for exploratory analysis and results.
  - **seaborn (sns)** : Generate statistical plots such as heatmaps for feature correlations.
  - **wordcloud (WordCloud)** : Visualize word frequency distributions in tweets.

- **Utilities**
  - **joblib** : Save/load trained preprocessing objects like scalers.
  - **tqdm** : Track progress in training loops and data preprocessing.
  - **warnings** : `filterwarnings('ignore')` — Keep experiment logs clean by suppressing irrelevant warnings.

# Chapter 4: Experiments & Results

This chapter presents the findings of the study, structured to systematically answer the core research questions and objectives established in Chapter 2. The experiments are designed to evaluate the system's performance at three key levels. First, we compare different recurrent neural network architectures to identify the most effective base model. Second, we investigate the central hypothesis of this dissertation by analyzing the impact of integrating NLP-derived features with traditional technical indicators on predictive performance. Finally, we assess the practical viability of the complete pipeline by simulating its trading performance to measure profitability and risk-adjusted returns. The following sections detail the evaluation metrics, experimental setup, and the results obtained at each stage.

## 4.1 Evaluation metrics

To ensure a robust evaluation, a combination of classification and regression metrics was employed. The primary goal of the trading system is to correctly predict the direction of the next day's price movement. Therefore, metrics that assess directional correctness are essential.

Accuracy serves as a baseline measure of overall correctness. However, in financial markets where the classes (up or down movements) can be imbalanced, accuracy alone can be misleading. To provide a more nuanced view, we use Precision, Recall, and the F1-Score. For our application, precision is particularly critical; it measures the proportion of positive predictions (i.e., predicted "up" movements) that were actually correct. A high-precision model minimizes false positives, which in a trading context correspond to losing trades. While Recall (the ability to identify all actual "up" movements) is important, a missed opportunity (a false negative) is generally less costly than a realized loss from a poorly placed trade, especially given the large pool of 88 stocks from which to select opportunities.

The Matthews Correlation Coefficient (MCC) is used as the primary metric for model comparison. As noted by Xu and Cohen [2018], MCC is a more reliable statistical measure as it produces a high score only if the prediction obtained good results in all four confusion matrix categories (true positives, false negatives, true negatives, and false positives), making it robust against data skew.

For models trained on the regression task (predicting the magnitude of price change), standard metrics such as Mean Squared Error (MSE) are calculated. However, to assess their practical utility, their Directional Accuracy is also computed by converting the predicted continuous returns into binary up/down signals. This allows for a direct comparison of the practical trading value of both classification and regression approaches.

## 4.2 Model Comparison

To establish the best predictive foundation for our system, an initial set of experiments was conducted using only stock-based technical indicators as features. These experiments evaluated four recurrent neural network architectures (LSTM, BiLSTM, GRU, and BiGRU) across two distinct prediction tasks:

- Classification: Predicting the binary direction of the next day's price movement (i.e., up or down).
- Predicting the magnitude of the next day's price movement, which is then converted to a directional signal for comparison.

Table 4.1: Evaluation Results: Classification models

| Setting | Precision | Recall | F1 | MCC | Accuracy |
|---|---|---|---|---|---|
| **LSTM** | **0.5202** | **0.6081** | **0.5224** | **0.0861** | **0.5368** |
| BiLSTM | 0.5283 | 0.6205 | 0.5219 | 0.0592 | 0.5173 |
| GRU | 0.4947 | 0.6020 | 0.5069 | 0.0440 | 0.5124 |
| BiGRU | 0.5280 | 0.6013 | 0.5189 | 0.0545 | 0.5173 |

As shown in Table 4.1, for the classification task, the standard LSTM model achieved the best performance across all key metrics. It secured the highest MCC (0.0861) and Accuracy (0.5368). While the bidirectional models were competitive, they did not demonstrate a superior predictive advantage over the simpler LSTM architecture in this fundamental test. It is important to contextualize these results: an MCC of 0.0861 indicates a weak positive correlation between the predictions and actual outcomes. While this value may seem low in absolute terms, it is not uncommon in the highly stochastic and noisy environment of day-trading prediction systems, underscoring the inherent difficulty of the task. This metric provides a more realistic performance assessment than accuracy alone, especially in financial datasets where class balance can be near 50%. The superior performance of the simpler LSTM architecture suggests that for this dataset and feature set, the additional context from future time steps captured by bidirectional layers did not translate into improved predictive power. Given its balance of performance and efficiency, the LSTM was selected for all subsequent classification-based experiments.

Table 4.2: Evaluation Results: Regression models

| Setting | Precision | Recall | F1 | MCC | MSE | Accuracy |
|---|---|---|---|---|---|---|
| LSTM | 0.4703 | 0.6630 | 0.5180 | 0.0245 | 0.000239 | 0.5006 |
| BiLSTM | 0.4937 | 0.6480 | 0.5198 | 0.0259 | 0.000240 | 0.5055 |
| **GRU** | **0.5216** | **0.6178** | **0.5149** | **0.0521** | **0.000238** | **0.5142** |
| BiGRU | 0.4703 | 0.6630 | 0.5180 | 0.0245 | 0.000239 | 0.5006 |

In the regression task, evaluated based on their directional accuracy, the GRU model emerged as the top performer (Table 4.2), achieving the highest MCC of 0.0521 and the lowest MSE. However, its overall directional performance was significantly lower than that of the best classification model (LSTM). This disparity is likely because predicting the precise magnitude of a stock's movement is a substantially more complex task than predicting its direction. The model's objective function in regression is focused on minimizing price-distance error, which may not align perfectly with maximizing directional correctness. The added complexity of predicting magnitude can obscure the primary directional signal, leading to poorer performance in a practical trading context. Consequently, the classification approach was deemed superior and was used for the final trading simulation.

This was also an advantage in system complexity, since the original design for the system expected a prediction akin to a probability for the next steps involving the Kelly criterion. Since the regression models output a magnitude instead of a degree of confidence, the process to use the kelly criterion with a regression model becomes much more intensive in both memory and processing time. The process involves using an array of independent models predicting the next price and processing the set of predictions in a different formulation of the kelly criterion. This approach would have been mathematically valid, but is out of scope for this project.

## 4.3 Impact of feature selection in model performance

This section investigates the central hypothesis of this dissertation: whether integrating NLP-derived features from Twitter enhances stock movement prediction. The best-performing architecture from the previous section (LSTM) was used to evaluate various combinations of features.

Table 4.3: Evaluation Results: LSTM model with different sets of features

| Setting | Precision | Recall | F1 | MCC | Accuracy |
|---|---|---|---|---|---|
| **Stock features only** | **0.5202** | **0.6081** | **0.5224** | **0.0861** | **0.5368** |
| Stance | 0.4943 | 0.5585 | 0.4771 | 0.0509 | 0.5123 |
| Sentiment | 0.4805 | 0.5831 | 0.4880 | 0.0457 | 0.5137 |
| Emotion | 0.5195 | 0.5837 | 0.5110 | 0.0473 | 0.5148 |
| Stance+Sentiment | 0.5026 | 0.5885 | 0.5054 | 0.0479 | 0.5152 |
| Sentiment+Emotion | 0.4845 | 0.5851 | 0.4958 | 0.0359 | 0.5180 |
| **Stance+Emotion** | **0.5162** | **0.6005** | **0.5220** | **0.0513** | **0.5223** |
| **Stance+Sentiment+Emotion** | **0.5280** | **0.6013** | **0.5189** | **0.0545** | **0.5173** |

The results presented in Table 3 reveal a counterintuitive outcome. The model trained on Stock features only outperformed all configurations that included Twitter-derived features substantially, achieving the highest MCC (0.0861) and Accuracy (0.5368). While the model incorporating all NLP features (Stance + Sentiment + Emotion) produced competitive F1 and Accuracy scores, its MCC was substantially lower (0.0545), indicating a less balanced and reliable predictive performance.

This suggests that, in their current aggregated form, the NLP features introduced more noise than signal. We speculate this may be due to two primary factors. First, there may be conflicting signals between the backward-looking technical indicators and the forward-looking, but often noisy, social media sentiment. A stock might be technically ”overbought”, signaling a downturn, while public sentiment remains positive, creating a contradictory input for the model and ultimately lowering the performance. Second, the raw aggregation of all tweets fails to account for their ”newsworthiness” or credibility. A handful of influential, high-impact tweets may be drowned out by a large volume of low-information content.

During the development stages of the project, the possibility of leveraging an agentic large language model (LLM) pipeline for tweet newsworthiness was explored but ultimately had to be abandoned due to time and computing power constraints. However, an important finding during this exploration stage was that, in many cases, the LLM would point out that the tweet by itself did not provide any information relevant to investment decisions. This points to shortcomings in our dataset and potential avenues of improvement for the system.

This finding does not invalidate the potential of social media data but highlights the need for more sophisticated feature engineering and strengthening of the dataset. Future work should explore methods to filter or weight tweets based on user influence, engagement metrics, or topic modeling to distill high-quality signals from the noise, potentially unlocking the predictive value of tweets and other social media sources.

## 4.4 Model comparison to baseline and previous work

To contextualize our findings, we compare the performance of our best model against an ARIMA baseline and the models from the literature that used the same dataset, most notably the HedgeFundAnalyst (HFA) variation of the stocknet model from [Xu and Cohen, 2018].

Table 4.4: Best models compared to previous papers

| **Model** | **Avg. Accuracy** | **MCC** |
|---|---|---|
| ARIMA, [Brown, 2004] | 0.514 | -0.020 |
| HAN, [Hu et al., 2018] | 0.576 | 0.051 |
| HFA, [Xu and Cohen, 2018] | 0.582 | 0.080 |
| LSTM (Stock features only) | 0.536 | 0.086 |
| LSTM (Stock + Twitter features) | 0.522 | 0.051 |

This difference can be attributed to fundamental architectural distinctions. The HFA model is a deep generative model that uses recurrent, continuous latent variables and neural variational inference to explicitly model the high stochasticity and chaotic nature of the market. In contrast, our approach is discriminative, learning a direct mapping from features to outcomes. The HFA model's generative nature is purpose-built to handle market randomness, which likely contributes to its higher accuracy. Nevertheless, the fact that our discriminative LSTM model, trained on a comprehensive set of technical indicators, can achieve a comparable MCC is a significant result. It validates the effectiveness of using a robust suite of technical features within a well-established deep learning framework, suggesting it is a sound and effective approach.

## 4.5 Simulation performance

The ultimate test of the system is its ability to generate profit in a realistic trading simulation. This final experiment evaluates the entire end-to-end pipeline, integrating the best predictive model that included tweet features (LSTM with stock and all twitter features) with the Isotonic Regression calibration and the Fractional Kelly Criterion for capital allocation over a 100-day trading period.

| Final Capital | ROI (%) | Sharpe Ratio |
|---|---|---|
| **$101,976.26** | **1.98** | **0.96** |
| $101,611.46 | 1.61 | 0.38 |
| $101,481.75 | 1.48 | 0.22 |
| $101,279.62 | 1.28 | 0.11 |
| $101,063.83 | 1.06 | -0.24 |

Table 4.5: Simulation Results for 100 Trading days (Initial Capital = $100,000.00)

The simulation results in Table 5 demonstrate the practical success of the proposed system. Starting with an initial capital of $100,000.00, the strategy concluded the 100-day period with $101,976.26, yielding a Return on Investment (ROI) of 1.98%. While a modest return, the most critical outcome is that the system was consistently profitable, validating the viability of the integrated pipeline.

Furthermore, the strategy achieved a consistent positive Sharpe Ratio. The Sharpe Ratio measures risk-adjusted return per unit of risk (volatility) taken. On its current state, the system performs better than a risk free asset with annualized returns of 3% even accounting for the volatility and risk taken. This suggests the portfolio construction and risk management components, particularly the sector-based filtering and the conservative Fractional Kelly sizing, were highly effective at creating a stable growth trajectory and mitigating risk. The simulation thus confirms that the end-to-end system, from prediction to execution, forms a robust and profitable trading framework.

With further refinements and improvements, the end-to-end pipeline could be deployed with a performance that consistently surpasses risk free options and could be used to turn a significant profit long term. A final discussion on the suggested refinements and improvements on the next and final chapter.

# Chapter 5: Conclusions

## 5.1 Achievements

This project successfully developed a complete end-to-end day trading system and systematically evaluated its performance, providing us with several key insights in response to the research questions posed in Chapter 2.

The first and most central research question concerned the impact of integrating NLP-derived features with technical indicators. The experiments in Chapter 4 revealed a surprising outcome: the LSTM model trained exclusively on historical stock data and technical indicators achieved a superior predictive performance (MCC of 0.0861) compared to all models that included NLP features. The addition of tweet-derived stance, sentiment, and emotion features, in their current aggregated form, appear to introduce more noise than signal, thereby degrading the model's accuracy.

This outcome presents a notable contrast to the findings of Xu and Cohen (2018), who demonstrated that their Hedge Fund Analyst (HFA) model benefited from incorporating tweet data. The divergence in our results does not appear to stem from the complexity of the predictive model, but rather from the strength of the baseline it was compared against. The LSTM model in this project was trained on a comprehensive suite of engineered technical indicators, which established a very strong baseline on its own. In fact, this technical-only model achieved an MCC of 0.0861, already surpassing the performance of the more complex HFA model (MCC of 0.080) on the same dataset. It is plausible that for our feature-rich baseline, the addition of raw, aggregated tweet data introduced more noise than valuable signal. In contrast, the baseline used by Xu and Cohen may have had more capacity to improve from the signals present in the tweet data. This suggests that the utility of NLP-derived signals is highly sensitive to the feature engineering pipeline used to process them, as simply aggregating sentiment may be insufficient to improve upon a strong technical foundation.

This finding directly answers the second research question regarding which NLP feature provides the most utility. While the model combining all NLP features was the best among the social media configurations, none of the NLP features, individually or combined, offered a clear advantage over a purely technical indicator approach. This suggests that without more sophisticated filtering, the raw sentiment of the crowd is not a reliable predictor in this context.

To answer the third research question, the end-to-end trading system, which leveraged calibrated LSTM predictions and a Fractional Kelly Criterion, successfully achieved positive returns. The simulation over 100 trading days yielded a return on investment (ROI) of 1.98%, demonstrating the viability of the overall framework. The system's ability to generate profit, even with a predictive model that did not benefit from NLP features, validates the robustness of the probability calibration, risk management, and portfolio construction components.

Finally, regarding the fourth research question on predictive consistency, the appendix tables show that the model's performance was not uniform across different stocks and sectors. Certain sectors consistently appeared in the top-performing lists (i.e. Consumer Goods) across various feature sets, while others (i.e. Basic Materials, Technology), consistently ranked among the worst. This indicates that the model's effectiveness is asset-dependent, and a one-size-fits-all approach may not be optimal.

A significant secondary achievement was that the baseline LSTM model, trained only on technical indicators, achieved an MCC that was competitive with the HFA model from Xu and Cohen (2018) on the same dataset. This validates the effectiveness of using a well-established discriminative model with a robust suite of technical features.

## 5.2 Limitations and Future work

Despite the successful implementation of a profitable system, this project faced several limitations that offer paths for future work.

The primary limitation lies in the tweet dataset and its application. The finding that NLP features degraded performance points to a lack of signal quality. Raw aggregation of all tweets for a given stock fails to distinguish between high-impact, insightful commentary and low-information noise. The dataset itself may be insufficient in volume or quality to capture meaningful shifts in market sentiment.

Hardware limitations also posed a significant challenge. The final simulation pipeline required loading, storing, and running predictions for 88 individual models in quick sequence for each trading day. This process was computationally intensive and time-consuming, constraining the scope and speed of experimentation and highlighting scalability challenges for a real-world, low-latency deployment.

For future work, it would be highly valuable to explore alternatives to tweets. Platforms like Reddit, particularly financial subreddits such as r/wallstreetbets or r/investing, offer a promising alternative. Reddit data is often easier to scrape via modern APIs, and its structure of threads and comments is inherently topic-classified, potentially providing a cleaner and more context-rich source of public sentiment.

Furthermore, a key area for improvement is the filtering and weighting of social media data. An LLM-based agent designed as a "newsworthiness" classifier was briefly explored during development but was abandoned due to time and processing power constraints. Reviving this concept represents a critical next step. Such an agent could be trained to identify and prioritize tweets or posts from credible sources or those containing genuinely market-moving information, effectively filtering out the noise that hampered the current model. This would allow the system to focus on high-quality signals, potentially unlocking the predictive power of social media data that this project was unable to take advantage of.

# Bibliography

Miriam Ayer, H. D. Brunk, G. M. Ewing, W. T. Reid, and Edward Silverman. An empirical distribution function for sampling with incomplete information. *The Annals of Mathematical Statistics*, 26(4):641–647, 1955. ISSN 00034851. URL http://www.jstor.org/stable/2236377.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155, March 2003. ISSN 1532-4435.

R.A. Brealey, S.C. Myers, and F. Allen. *Principles of Corporate Finance*. McGraw-Hill/Irwin series in finance, insurance, and real estate. McGraw-Hill/Irwin, 2011. ISBN 9780071314176. URL https://books.google.co.uk/books?id=T9y4SgAACAAJ.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2): 79–85, 1990. URL https://aclanthology.org/J90-2002/.

Robert Goodell Brown. *Smoothing, forecasting and prediction of discrete time series*. Courier Corporation, 2004.

Noam Chomsky. *Syntactic Structures*. De Gruyter Mouton, Berlin, Boston, 1957. ISBN 9783112316009. doi: doi:10.1515/9783112316009. URL https://doi.org/10.1515/9783112316009.

Wikimedia Commons. File:isotonic regression.svg — wikimedia commons, the free media repository, 2020. URL https://commons.wikimedia.org/w/index.php?title=File:Isotonic_regression.svg&oldid=472818415. [Online; accessed 31-August-2025].

Wikimedia Commons. File:kelly criterion 3d.png — wikimedia commons, the free media repository, 2025. URL https://commons.wikimedia.org/w/index.php?title=File:Kelly_Criterion_3D.png&oldid=1058134207. [Online; accessed 31-August-2025].

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. URL https://arxiv.org/abs/1810.04805.

A. Fielding. Statistical inference under order restrictions. the theory and application of isotonic regression. *Journal of the Royal Statistical Society. Series A (General)*, 137(1):92–93, 1974. ISSN 00359238. URL http://www.jstor.org/stable/2345150.

D.V. Godoy. *DEEP LEARNING WITH PYTORCH STEP-BY-STEP: A Beginner's Guide*. INDEPENDENTLY PUBLISHED, 2022. ISBN 9798485032760. URL https://books.google.co.uk/books?id=9j-0zwEACAAJ.

Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. 04 1991.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 11 1997. doi: 10.1162/neco.1997.9.8.1735.

Ziniu Hu, Weiqing Liu, Jiang Bian, Xuanzhe Liu, and Tie-Yan Liu. Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction, 2018. URL `https://arxiv.org/abs/1712.02136`.

J. L. Kelly. A new interpretation of information rate. *The Bell System Technical Journal*, 35(4):917–926, 1956. doi: 10.1002/j.1538-7305.1956.tb03809.x.

John F. Kolen and Stefan C. Kremer. *Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies*, pages 237–243. 2001. doi: 10.1109/9780470544037.ch14.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks, 2018. URL `https://arxiv.org/abs/1703.07015`.

Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194):20200209, February 2021. ISSN 1471-2962. doi: 10.1098/rsta.2020.0209. URL `http://dx.doi.org/10.1098/rsta.2020.0209`.

Christopher D. Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA, 1999. ISBN 0262133601.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL `https://arxiv.org/abs/1301.3781`.

J.J. Murphy. *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications*. New York Institute of Finance Series. Penguin Publishing Group, 1999. ISBN 9780735200661. URL `https://books.google.co.uk/books?id=5zhXEqdr_IcC`.

Louis M. Rotando and Edward O. Thorp. The kelly criterion and the stock market. *Am. Math. Monthly*, 99(10):922–931, December 1992. ISSN 0002-9890. doi: 10.2307/2324484. URL `https://doi.org/10.2307/2324484`.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986. URL `https://api.semanticscholar.org/CorpusID:205001834`.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. CARER: Contextualized affect representations for emotion recognition. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1404. URL `https://aclanthology.org/D18-1404/`.

William F. Sharpe. Mutual fund performance. *The Journal of Business*, 39(1): 119–138, 1966. ISSN 00219398, 15375374. URL http://www.jstor.org/stable/2351741.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014. URL https://arxiv.org/abs/1409.3215.

Edward O. Thorp. Chaper 9 - the kelly criterion in blackjack sports betting, and the stock market*. In S.A. Zenios and W.T. Ziemba, editors, *Handbook of Asset and Liability Management*, pages 385–428. North-Holland, San Diego, 2008. ISBN 978-0-444-53248-0. doi: https://doi.org/10.1016/B978-044453248-0.50015-0. URL https://www.sciencedirect.com/science/article/pii/B9780444532480500150.

A. M. Turing. I.—computing machinery and intelligence. *Mind*, LIX(236):433–460, 10 1950. ISSN 0026-4423. doi: 10.1093/mind/LIX.236.433. URL https://doi.org/10.1093/mind/LIX.236.433.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. URL https://arxiv.org/abs/1706.03762.

Jan von Plato. Chapter 75 - a.n. kolmogorov, grundbegriffe der wahrscheinlichkeitsrechnung (1933). In I. Grattan-Guinness, Roger Cooke, Leo Corry, Pierre Crépel, and Niccolo Guicciardini, editors, *Landmark Writings in Western Mathematics 1640-1940*, pages 960–969. Elsevier Science, Amsterdam, 1933. ISBN 978-0-444-50871-3. doi: https://doi.org/10.1016/B978-044450871-3/50156-X. URL https://www.sciencedirect.com/science/article/pii/B978044450871350156X.

Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1): 36–45, January 1966. ISSN 0001-0782. doi: 10.1145/365153.365168. URL https://doi.org/10.1145/365153.365168.

Yumo Xu and Shay B. Cohen. Stock movement prediction from tweets and historical prices. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1970–1979, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1183. URL https://aclanthology.org/P18-1183/.

Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, page 694–699, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 158113567X. doi: 10.1145/775047.775151. URL https://doi.org/10.1145/775047.775151.

# Appendix A: Source Code

The full implementation and supporting scripts for this dissertation are available on GitHub. They include data preprocessing, model training, and evaluation code, as well as the end-to-end investment simulation pipeline.

`https://github.com/DashieLashie/summer-project`

# Appendix B: Additional Tables

Table B.1: Stance: Performance by Sector

| Sector | Accuracy | MCC |
|---|---|---|
| Consumer Goods | $0.548 \pm 0.085$ | 0.103 |
| Financial | $0.545 \pm 0.054$ | 0.089 |
| Healthcare | $0.540 \pm 0.053$ | 0.057 |
| Utilities | $0.530 \pm 0.206$ | 0.075 |
| Industrial Goods | $0.512 \pm 0.067$ | 0.076 |
| Services | $0.497 \pm 0.063$ | 0.054 |
| Technology | $0.481 \pm 0.060$ | 0.006 |
| Basic Materials | $0.448 \pm 0.093$ | -0.055 |

Table B.2: Stance: Top 10 Performers (by Directional Accuracy)

| Ticker | Sector | Accuracy | MCC |
|---|---|---|---|
| D | Utilities | 0.676 | 0.376 |
| PEP | Consumer Goods | 0.659 | 0.319 |
| ABBV | Healthcare | 0.617 | 0.211 |
| JPM | Financial | 0.612 | 0.235 |
| CELG | Healthcare | 0.610 | 0.249 |
| BHP | Basic Materials | 0.600 | 0.176 |
| DIS | Services | 0.600 | 0.010 |
| UPS | Services | 0.600 | 0.192 |
| WFC | Financial | 0.600 | 0.193 |
| KO | Consumer Goods | 0.596 | 0.192 |

Table B.3: Stance+Sentiment: Performance by Sector

| Sector | Accuracy | MCC |
|---|---|---|
| Consumer Goods | $0.581 \pm 0.072$ | 0.176 |
| Financial | $0.532 \pm 0.057$ | 0.033 |
| Industrial Goods | $0.516 \pm 0.079$ | 0.046 |
| Utilities | $0.512 \pm 0.072$ | 0.030 |
| Basic Materials | $0.508 \pm 0.094$ | 0.086 |
| Healthcare | $0.507 \pm 0.072$ | 0.014 |
| Services | $0.503 \pm 0.058$ | 0.022 |
| Technology | $0.482 \pm 0.061$ | 0.020 |

Table B.4: Stance+Sentiment: Top 10 Performers (by Directional Accuracy)

| **Ticker** | **Sector** | **Accuracy** | **MCC** |
|---|---|---|---|
| ABBV | Healthcare | 0.681 | 0.352 |
| PEP | Consumer Goods | 0.659 | 0.321 |
| MO | Consumer Goods | 0.644 | 0.286 |
| BA | Industrial Goods | 0.615 | 0.246 |
| MA | Financial | 0.614 | 0.160 |
| T | Technology | 0.605 | 0.239 |
| SLB | Basic Materials | 0.600 | 0.150 |
| UPS | Services | 0.600 | 0.175 |
| PM | Consumer Goods | 0.591 | 0.087 |
| CVX | Basic Materials | 0.585 | 0.161 |

Table B.5: Sentiment: Performance by Sector

| **Sector** | **Accuracy** | **MCC** |
|---|---|---|
| Healthcare | $0.535 \pm 0.069$ | 0.062 |
| Utilities | $0.531 \pm 0.026$ | 0.080 |
| Financial | $0.523 \pm 0.079$ | 0.053 |
| Technology | $0.522 \pm 0.056$ | 0.066 |
| Consumer Goods | $0.511 \pm 0.106$ | 0.053 |
| Industrial Goods | $0.503 \pm 0.073$ | 0.012 |
| Services | $0.502 \pm 0.053$ | 0.015 |
| Basic Materials | $0.483 \pm 0.062$ | 0.065 |

Table B.6: Sentiment: Top 10 Performers (by Directional Accuracy)

| **Ticker** | **Sector** | **Accuracy** | **MCC** |
|---|---|---|---|
| V | Financial | 0.667 | 0.336 |
| ABBV | Healthcare | 0.638 | 0.339 |
| KO | Consumer Goods | 0.632 | 0.243 |
| CELG | Healthcare | 0.627 | 0.255 |
| PEP | Consumer Goods | 0.614 | 0.311 |
| MDT | Healthcare | 0.605 | 0.244 |
| DIS | Services | 0.600 | 0.053 |
| BA | Industrial Goods | 0.596 | 0.179 |
| FB | Technology | 0.584 | 0.182 |
| UPS | Services | 0.578 | 0.120 |

Table B.7: Emotion: Performance by Sector

| **Sector** | **Accuracy** | **MCC** |
|---|---|---|
| Consumer Goods | $0.588 \pm 0.031$ | 0.123 |
| Financial | $0.542 \pm 0.061$ | 0.091 |
| Industrial Goods | $0.537 \pm 0.085$ | 0.091 |
| Utilities | $0.536 \pm 0.178$ | 0.109 |
| Healthcare | $0.514 \pm 0.032$ | 0.008 |
| Technology | $0.493 \pm 0.081$ | 0.019 |
| Basic Materials | $0.486 \pm 0.086$ | 0.066 |
| Services | $0.484 \pm 0.065$ | -0.019 |

Table B.8: Emotion: Top 10 Performers (by Directional Accuracy)

| **Ticker** | **Sector** | **Accuracy** | **MCC** |
|---|---|---|---|
| D | Utilities | 0.662 | 0.381 |
| BA | Industrial Goods | 0.654 | 0.300 |
| PG | Consumer Goods | 0.625 | 0.000 |
| MO | Consumer Goods | 0.622 | 0.237 |
| GOOG | Technology | 0.618 | 0.260 |
| GE | Industrial Goods | 0.606 | 0.118 |
| T | Technology | 0.605 | 0.213 |
| DIS | Services | 0.600 | -0.024 |
| WFC | Financial | 0.600 | 0.193 |
| V | Financial | 0.596 | 0.189 |

Table B.9: Sentiment+Emotion: Performance by Sector

| **Sector** | **Accuracy** | **MCC** |
|---|---|---|
| Consumer Goods | $0.566 \pm 0.099$ | 0.131 |
| Financial | $0.536 \pm 0.033$ | 0.032 |
| Basic Materials | $0.523 \pm 0.105$ | 0.041 |
| Industrial Goods | $0.512 \pm 0.096$ | 0.009 |
| Technology | $0.510 \pm 0.079$ | 0.050 |
| Services | $0.505 \pm 0.067$ | 0.037 |
| Healthcare | $0.504 \pm 0.066$ | -0.005 |
| Utilities | $0.500 \pm 0.090$ | 0.003 |

Table B.10: Sentiment+Emotion: Top 10 Performers (by Directional Accuracy)

| **Ticker** | **Sector** | **Accuracy** | **MCC** |
|---|---|---|---|
| PEP | Consumer Goods | 0.705 | 0.410 |
| SLB | Basic Materials | 0.675 | 0.232 |
| BA | Industrial Goods | 0.654 | 0.317 |
| CELG | Healthcare | 0.644 | 0.369 |
| MSFT | Technology | 0.608 | 0.199 |
| GOOG | Technology | 0.605 | 0.213 |
| MO | Consumer Goods | 0.600 | 0.190 |
| PM | Consumer Goods | 0.591 | 0.058 |
| AAPL | Consumer Goods | 0.590 | 0.233 |
| MCD | Services | 0.586 | 0.210 |

Table B.11: Stance+Emotion: Performance by Sector

| **Sector** | **Accuracy** | **MCC** |
|---|---|---|
| Consumer Goods | $0.594 \pm 0.024$ | 0.179 |
| Financial | $0.553 \pm 0.054$ | 0.082 |
| Utilities | $0.544 \pm 0.008$ | 0.079 |
| Services | $0.517 \pm 0.069$ | 0.043 |
| Healthcare | $0.511 \pm 0.059$ | 0.006 |
| Basic Materials | $0.511 \pm 0.178$ | 0.057 |
| Technology | $0.500 \pm 0.087$ | 0.011 |
| Industrial Goods | $0.490 \pm 0.061$ | 0.030 |

Table B.12: Stance+Emotion: Top 10 Performers (by Directional Accuracy)

| **Ticker** | **Sector** | **Accuracy** | **MCC** |
|---|---|---|---|
| SLB | Basic Materials | 0.775 | 0.498 |
| MA | Financial | 0.659 | 0.285 |
| GOOG | Technology | 0.645 | 0.290 |
| ABBV | Healthcare | 0.638 | 0.297 |
| HD | Services | 0.633 | 0.192 |
| PG | Consumer Goods | 0.625 | 0.166 |
| MO | Consumer Goods | 0.622 | 0.250 |
| CVX | Basic Materials | 0.604 | 0.200 |
| PEP | Consumer Goods | 0.591 | 0.183 |
| KO | Consumer Goods | 0.579 | 0.258 |

Table B.13: Stance+Sentiment+Emotion: Performance by Sector

| **Sector** | **Accuracy** | **MCC** |
|---|---|---|
| Consumer Goods | $0.575 \pm 0.049$ | 0.121 |
| Industrial Goods | $0.538 \pm 0.097$ | 0.118 |
| Utilities | $0.535 \pm 0.140$ | 0.082 |
| Services | $0.529 \pm 0.069$ | 0.088 |
| Financial | $0.523 \pm 0.054$ | 0.104 |
| Technology | $0.499 \pm 0.086$ | 0.058 |
| Healthcare | $0.477 \pm 0.074$ | -0.059 |
| Basic Materials | $0.475 \pm 0.109$ | -0.053 |

Table B.14: Stance+Sentiment+Emotion: Top 10 Performers (by Directional Accuracy)

| **Ticker** | **Sector** | **Accuracy** | **MCC** |
|---|---|---|---|
| BA | Industrial Goods | 0.712 | 0.424 |
| GOOG | Technology | 0.684 | 0.361 |
| PEP | Consumer Goods | 0.659 | 0.319 |
| D | Utilities | 0.634 | 0.279 |
| DIS | Services | 0.633 | 0.238 |
| UPS | Services | 0.622 | 0.253 |
| CVX | Basic Materials | 0.604 | 0.200 |
| V | Financial | 0.596 | 0.202 |
| PM | Consumer Goods | 0.591 | 0.058 |
| CAT | Industrial Goods | 0.588 | 0.265 |