Chapter 3: Research Methodology

3.1 Introduction

The purpose of this chapter is to outline the research methodology employed in this dissertation and will detail the specific methods used for sentiment analysis, including the various models and techniques for data processing and analysis.

The present study adopts a mixed-methods approach, combining both quantitative and qualitative elements in its methodology. The quantitative aspect of the research is evident in the application of various sentiment analysis models and techniques, as well as the use of statistical tests to evaluate the relationship.

On the other hand, the qualitative aspect of the study is crucial for understanding and interpreting the sentiment analysis results, as well as for providing context to the relationships identified in the quantitative analysis. Hence, this approach offers a comprehensive and rigorous examination of the research questions, effectively integrating quantitative and qualitative methods to provide valuable insights into the relationship between football-related tweet sentiment and league standings and match results. The following sections will elaborate on the specific models and techniques used in the study, as well as the data collection and analysis procedures.

3.2 Prototype Pipeline

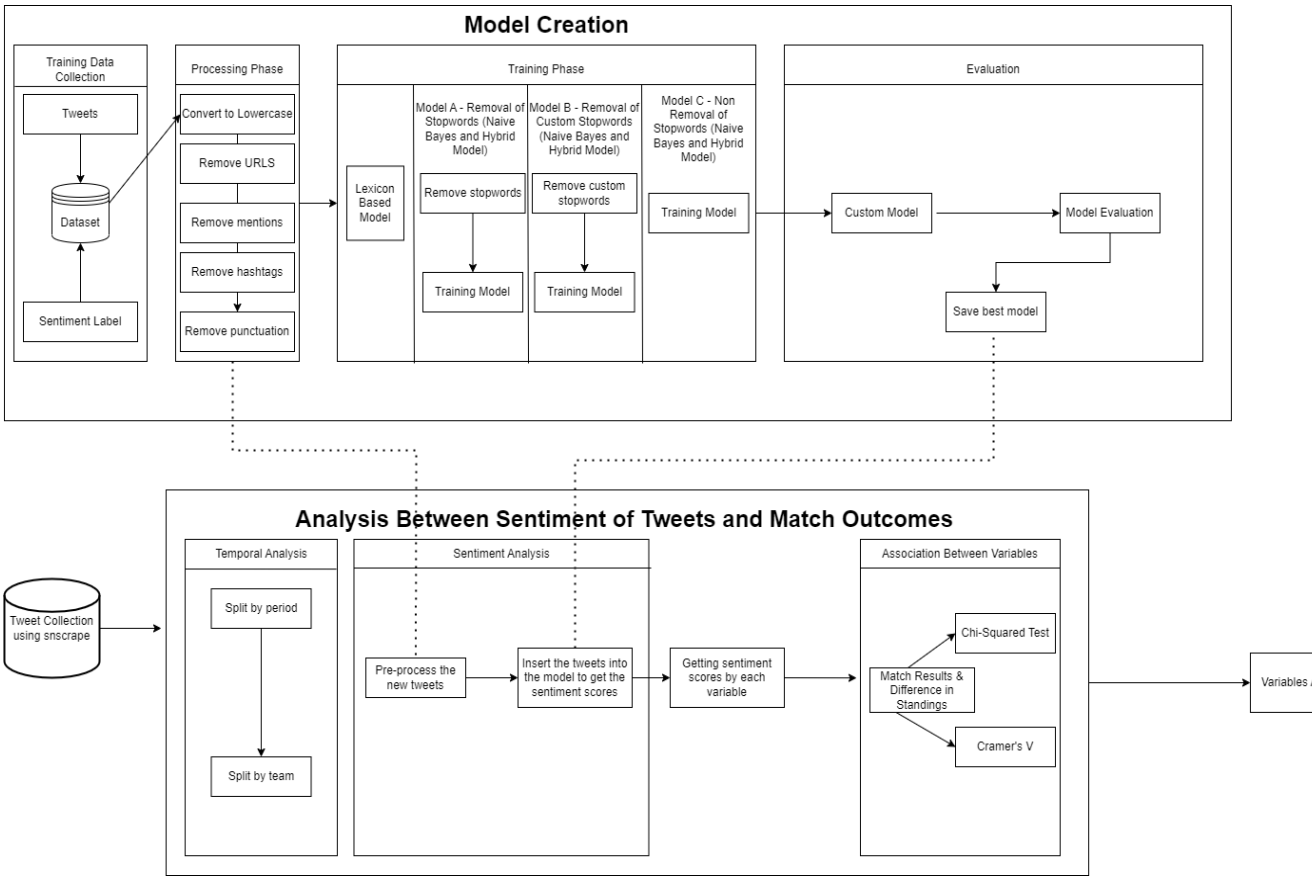

Figure 3.1: *Research Pipeline*

3.3 Frameworks and Libraries

In developing the sentiment analysis system, Python was utilized as the programming language of choice due to its versatility, simplicity, power, and dynamic nature, as well as its proficiency in processing natural language data. Python's extensive collection of open-source libraries and large user community further contributed to its appeal and functionality for various applications. Additionally, to facilitate ease of package management, the virtual environment was created and activated using the Anaconda platform, allowing for seamless installation, updates,

and removals of necessary packages. The Anaconda version 22.9.0 was employed for this research.

The main packages used in this study are as follows:

Package	Description	Sentiment Analysis Applications
pandas	Data manipulation library for structured data	Data preprocessing, cleaning, and feature extraction
nltk	Natural language processing library	Text preprocessing, tokenization, stemming, and sentiment analysis
sklearn	Machine learning library	Building and evaluating sentiment analysis models
scipy	Collection of mathematical and scientific functions for numerical computation	Supporting mathematical operations in sentiment analysis algorithms
torch	Deep learning library with GPU acceleration	Building and training deep learning models for sentiment analysis
numpy	Numerical computing library for array and matrix operations	Data preprocessing and backend support for other libraries
optuna	Hyperparameter optimization framework	Fine-tuning models for improved sentiment analysis performance

Table 3.1: *Summary of Python packages for sentiment analysis*

3.4 Training Dataset

3.4.1 Overview

The data set utilized for training the football sentiment models was sourced from a publicly available GitHub repository [1]. This repository comprises three discrete files that pertain to football-related tweets, player tweets and World Cup tweets, along with the associated sentiments expressed. Since our research, solely focuses on the sentiment of football team, the data sets of both teams and players were merged.

[1] `https://github.com/charlesmalafosse/open-dataset-for-sentiment-analysis`

3.4.2 Data Manipulation

The data set under examination featured various fields, including Tweet Creation Date, Tweet ID, Tweet Language and Sentiment Score. Both the date of tweet creation and sentiment rating attributes were subsequently omitted, as they were deemed inapplicable for the purposes of training the model. Upon the compilation of the data sets, a total of 762,643 redundant tweets were discerned and subsequently deleted. The tweet identifier column was ultimately discarded, as it no longer held any relevance to the analytical investigation being conducted.

In order to ensure the effectiveness of our sentiment analysis model, a language verification was executed to corroborate that all the tweets within the dataset were composed in the English Language. Upon inspecting the sentiment analysis categories of the dataset, the mixed category was removed due to the relatively small number of tweets assigned to it. Since the tweets were not equally distributed for each category, the categories were down-sampled to a uniform count of 354,501 tweets per classification. This action was deemed necessary to maintain the integrity of our analysis and avoid potential inaccuracies in the model training and evaluation process.

The pre-processing pipeline of the training tweets included removing redundant words, abbreviations, and unwanted patterns such as URLs, usernames, punctuation's and hashtags.

Dataset Sizes		
Pre-processing Tasks	**File Size (MB)**	**Processing Time (s)**
Before pre-processing	734.2	NA
After removing URLs	699.5	42.3
Removing of mentions ”@”	585.6	57.7
Filtering # from tweets	544.0	50.2
Removing punctuation’s	523.3	62.7

Table 3.2: *Data Refinement Process*

This table below provides a clear comparison of tweets before and after the pre-processing tasks and showcasing the cleaned and simplified tweet text that will be used for training the sentiment analysis model.

Before pre-processing Tasks	**After pre-processing Tasks**
I have to agree with #Lovren he has become one of the best defenders in the world but can he keep it up I hope he can #LFC DejanLovren	i have to agree with he has become one of the best defenders in the world but can he keep it up i hope he can
I hate supporting NUFC while Mike Ashley owns the club. It has ruined football and my love of the sport year after year. #NUFC	i hate supporting while mike ashley owns the club it has ruined football and my love of the sport year after year
Why haven’t we signed anyone in almost 24 hours. This is a shambles. #cpfc	why havent we signed anyone in almost 24 hours this is a shambles

Table 3.3: *Table to show the tweets before and after processing*

3.4.3 Stop Words Filtering

Stopwords are common words in a language that are often considered insignificant and removed from text data during natural language processing tasks. They are typically articles, prepositions and conjunctions which carry little meaning on their own.

However, in the context of analyzing football-related tweets, eliminating stop words may inadvertently strip away important contextual information. For instance, the tweet ”My team didn’t play well”, removing stop words leaves us with ”team, play, well” which may lead to misinterpretation or loss of sentiment.

To address this issue, three different approaches are proposed for the Naive Bayes and Hybrid models in processing football-related tweets.

1. This involves training the models using the standard NLTK stop words list, which is recommended by researchers for normal sentiment analysis tasks.

2. This approach involves creating a customized stop words list by retaining important words from the NLTK list that are relevant to the football domain:

Stop word	Original Tweet	Processed Tweet	Reason for Importance in the context
Off	”The players look off today”	”players look today”	Maintains poor performance context
Over	”The game is over”	”game”	Keeps game over context
Under	”Our team is under performing”	”team performing”	Losing context of superlative degree
Few	”Few supporters showed to the game”	”supporters showed game”	Losing quantity of supporters context

More	”Our team needs more training”	”team training”	Losing superlative degree context
No	”No goals were scored in the match”	”goals scored”	Losing lack of goals context
Not	”Our team did not play well today”	”team play well today”	Losing the negation of the sentiment
Don’t	”Don’t underestimate our team”	”underestimate team”	Losing advice or warning context
Should	”We should have played better”	”played better”	Losing the context and sentiment
Should’ve	”We should’ve won the match”	”won match”	Losing negative result context
Aren’t	”Our players aren’t performing well”	”players performing well”	Losing the context of negation
Couldn’t	”Our team couldn’t score a goal”	”team score goal”	Losing the context of inability
Didn’t	”We didn’t win the match”	”win match”	Losing the context of negation
Doesn’t	”Our team doesn’t give up easily”	”team give easily”	Losing the context of negation

Hadn't	"We hadn't practiced enough before the game"	"practiced enough game"	Losing the context of negation and past event
Haven't	"We haven't lost a game this season"	"lost game season"	Losing the context of negation
Mustn't	"We mustn't lose focus during the match"	"lose focus match"	Losing advice or warning context
Shouldn't	"We shouldn't have underestimated the other team"	"underestimated team"	Losing recommendation context
Wasn't	"The referee wasn't fair in his decisions"	"referee fair decisions"	Losing the context of negation
Weren't	"Our players weren't in good form today"	"players good form today"	Losing the context of negation
Won't	"Those decisions won't help the team"	"decisions help team"	Losing the context of negation
Wouldn't	"Our coach wouldn't make such a decision without reason"	"coach make decision reason"	Losing negation and assumption context

3. Entails not using any stop words, allowing the models to process the

tweets in their entirety.

The NLTK stopwords list accounts for words with an apostrophe, such as ”wouldn’t”, but does not include variations without the apostrophe, like ”wouldnt”. Given that many tweets may contain informal language or unconventional spellings, a customized function has been implemented to accommodate these variations and ensure that stopwords without apostrophes are also considered during the text processing.

The primary aim of conducting a comparative analysis of the performance of these methodologies is to ascertain the most efficacious approach with regard to the utilization of stopwords in the context of football-related tweets.

3.5 Model Selection

The primary objective of this chapter is to provide a systematic approach to selecting the most effective model for sentiment analysis in the context of football-related tweets. As the field of sentiment analysis encompasses a diverse range of methods and techniques, it is imperative to identify a model that offers superior performance and reliability when applied to the unique characteristics and nuances of football-related content.

3.5.1 VADER Implementation

In the present study, the VADER (Valence Aware Dictionary and sEntiment Reasoner) lexicon-based model was utilized as an alternative approach to classify the sentiment of football-related tweets. VADER constitutes a lexicon and rule-

based sentiment analysis instrument, explicitly designed for discerning sentiment expressed in social media text. This method considers the intensity of emotions and furnishes a compound score indicative of the overarching sentiment present within a given textual sample.

In order to perform sentiment analysis utilizing VADER, the SentimentIntensityAnalyzer [2] class from the NLTK library was employed. A custom function, named 'predict-sentiment,' was defined to accept textual input and return the corresponding sentiment polarity, which is determined by the compound score generated by VADER. Adhering to the approach outlined by [48], a threshold of ±0.05 was established for the compound score. Consequently, a score greater than or equal to 0.05 was classified as positive, while a score less than or equal to -0.05 was deemed negative; scores falling in between these values were considered neutral.

3.5.2 Naive Bayes Model

The chosen machine learning algorithm as described in the previous chapter for sentiment analysis is the Multinomial Naïve Bayes classifier. The employment of the Multinomial Naive Bayes classifier, as opposed to the Bernoulli Naive Bayes classifier, is justified by the nature of the text data. This classifier is capable of capitalizing on the wealth of information furnished by word frequencies, potentially having a superior classification performance in comparison to a Bernoulli Naive Bayes classifier, which solely takes into account the presence or absence of words.

[2] `https://www.nltk.org/api/nltk.sentiment.html`

The study aims to classify the sentiment of football-related tweets using this probabilistic approach. The classifier uses two distinct techniques: Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF).

Bag of Words Approach

The Bag-of-Words (BoW) model representation was used using CountVectorizer [3], which tokenizes the text data and constructs a vocabulary containing all unique words present in the corpus. The text data is then transformed into a matrix where each row represents a document, and each column corresponds to a word in the vocabulary. This approach doesn't take into consideration the order of words; it merely focuses on their presence and frequency.

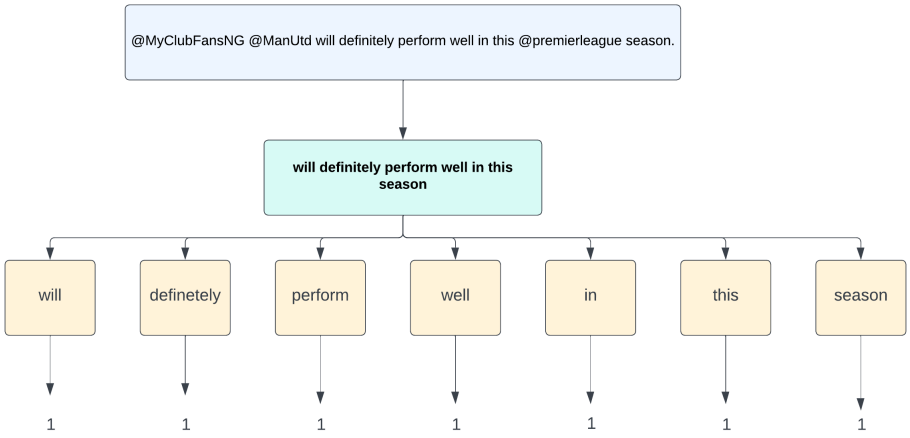

Figure 3.2: *Example of a Training Tweet in Bag of Words Representation*

The following parameters were used to test the best parameters for the BoW model:

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html`

Hyperparameter	Description
ngram-range	Determines the size of word combinations (uni-grams, bi-grams, tri-grams) to be considered.
max-features	Limits the number of features to be extracted from the text. The search space included 1000, 5000, 10000, and None, where None indicates no limit on the number of features.
alpha	Represents the smoothing parameter for the Multinomial Naive Bayes classifier. Different values (0.01, 0.1, 1.0, 10.0, 100.0) reflect different assumptions about the training data's representativeness and the influence of unseen words on the classification.

Table 3.5: *Hyper-parameters considered for the BoW Naive Bayes model.*

Term Frequency-Inverse Document Frequency (TF-IDF)

The subsequent prototype that underwent testing utilized the TF-IDF methodology, an alternative representation to Bag of Words (BoW) that considers the significance of words in a given dataset. This methodology is derived from two distinct measures: term frequency (TF) and inverse document frequency (IDF). The former measures the frequency of occurrence of a word within a given document, while the latter measures the significance of a word across all documents in the dataset. As a result, the resulting TF-IDF value assigns greater weight to words that possess a higher level of importance but are less frequent across all documents.

Metric	**Description**
Term Frequency (TF)	The raw frequency of a word in a document is normalized by dividing it by the total number of words in that document. This adjustment accounts for varying document lengths and provides a proportional measure of each word's contribution to the document.
Inverse Document Frequency (IDF)	This metric aims to quantify the significance of a word in the entire corpus. By calculating the logarithm of the total number of documents divided by the number of documents containing the word, common words that appear in many documents are assigned lower weights, while unique and context-specific words are assigned higher weights.

Table 3.6: *Description of Term Frequency and Inverse Document Frequency*

In addition to the parameters that were used to the Bag of Words implementation, the ensuing parameters were also utilized to attain the optimal parameters possible:

Hyperparameter	**Description**
use-idf	A boolean flag that determines whether to use the Inverse Document Frequency (IDF) weighting..
norm	Specifies the normalization method for the TF-IDF scores. L1 normalization (Manhattan or Taxicab) calculates the sum of the absolute values of vector components. L2 normalization (Euclidean) calculates the square root of the sum of the squared vector components.
fit-prior	A boolean flag that determines whether to learn class prior probabilities from the training data. If set to True, the classifier will estimate the prior probabilities based on the training data. If set to False, it will use uniform prior probabilities, assuming that each class is equally likely.

Table 3.7: *Hyper-parameters considered for the TF-IDF Naive Bayes model.*

For each sentiment class, the model calculates the probability that the input

text belongs to that class. This is done by multiplying the conditional probabilities of each token (TF-IDF value) given the sentiment class, and then multiplying by the prior probability of the class (based on the proportion of each class in the training dataset).

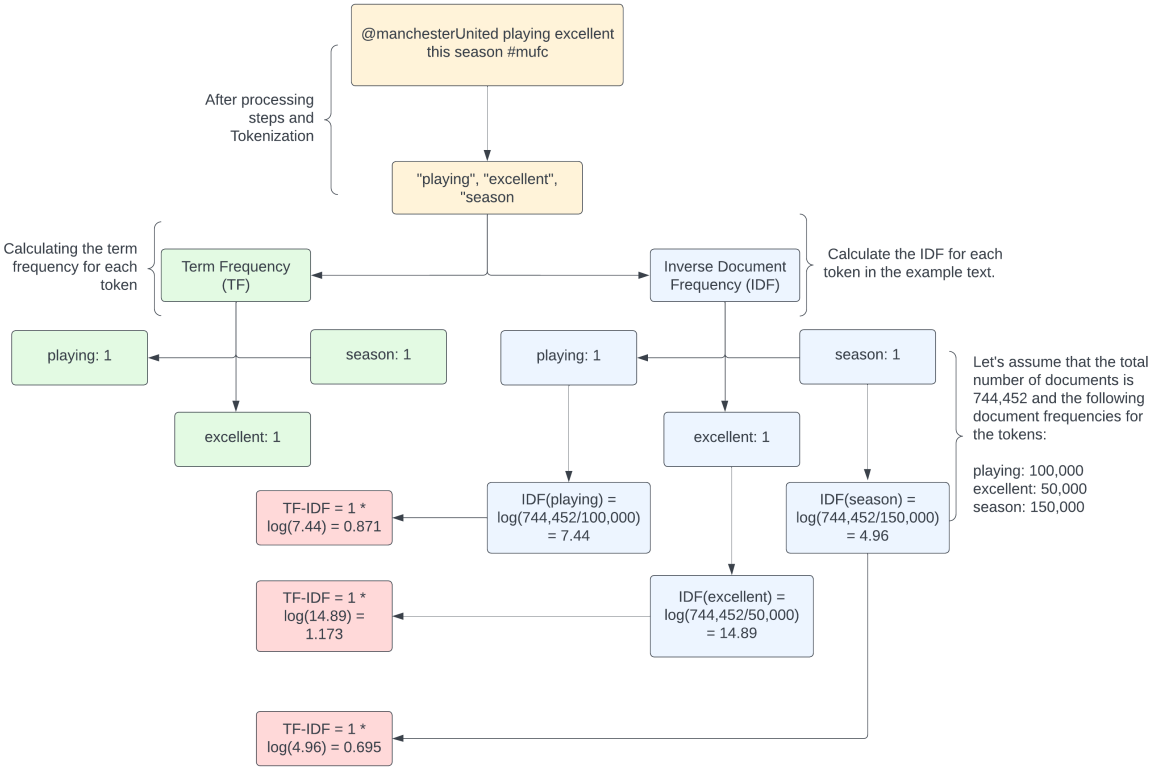

Figure 3.3: *Example of the TF-IDF process*

Training the Model and Hyper-parameter Tuning

The dataset was split into a training, validation and test dataset, with a 70:15:15 allocation, respectively. This allowed the model to be trained on a large portion of the data and subsequently evaluated on a separate, unseen dataset to assess its performance and generalizability.

In order to optimize the model's performance and identify the best hyper-

parameters, a search cross-validation using the GridSearchCV [4] function from the Scikit-learn library was used. RandomizedSearchCV is an alternative to GridSearchCV, another popular method for hyper-parameter tuning.

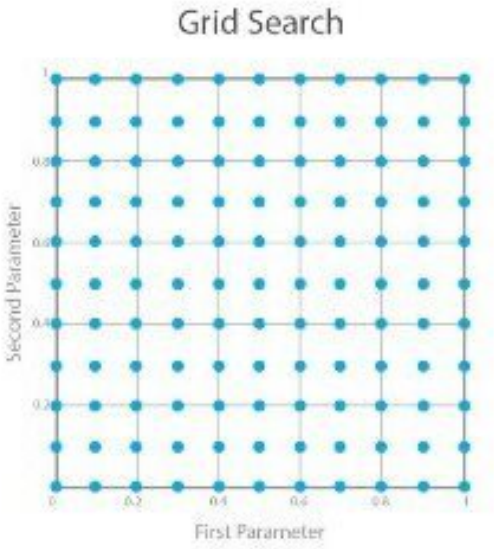

Figure 3.4: *An illustration of a grid search space. A range of the possible parameters and the algorithm makes a complete search over them.*

In contrast to GridSearchCV, which searches all possible combinations of hyper-parameters, RandomizedSearchCV randomly selects a pre-defined number of combinations from the specified hyper-parameter space [49]. This approach is more computationally efficient, especially when dealing with a large number of hyper-parameters or when the search space is vast. Nevertheless, it is imperative for the model to be trained utilizing the most optimal hyper-parameters available. Consequently, GridSearchCV emerges as the superior solution, given that RandomizedSearchCV encompasses a limited set of hyper-parameters in its pursuit of attaining the highest accuracy.

In an effort to improve upon the methodology employed in a previous Naive

[4] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`

Bayes study by Iqbal et al. [33], the GridSearchCV instance employed a 10-fold cross-validation approach to ensure a more thorough evaluation of the model's performance across different hyper-parameter configurations. The optimal parameters, as determined by the validation accuracy, were evaluated using the test dataset to determine if there were any discrepancies in accuracy when stopwords were used, including customized stopwords, in comparison to the model without stopwords. The evaluation metrics included accuracy and a classification report, which provided detailed information on precision, recall, and F1-score for each class.

3.5.3 Hybrid CNN+LSTM Model

This part outlines the methodology employed in our study, which combines a hybrid CNN-LSTM model with GloVe word embeddings to perform sentiment analysis. The following sections detail the steps taken to achieve the best possible accuracy, the hyper-parameters tested, the validation and test accuracy, connections to previous studies, and other relevant information.

Data Preparation

The initial step in model development was data preparation and pre-processing, involving tokenization and padding of input text. Tokenization, a critical step in natural language processing, breaks raw text into smaller units or tokens. In our implementation, tokens represent individual words in the input text, converting unstructured data into a structured format for machine learning models.

Keras Tokenizer class was employed for tokenization, initialized with a 'num-

words' parameter set to a maximum of 30,000. This limits the vocabulary size and reduces computational requirements, considering only the top 30,000 frequent words in the dataset. Less frequent words, often seen as noise, may not significantly impact the model's performance. Further- more, to ensure uniform length of input sequences, padding was applied using the pad-sequence function from Keras, setting the maximum length to 100.

Model Architecture

The methodology employed in this study leverages a sophisticated hybrid CNN-LSTM model with GloVe word embeddings to enhance sentiment analysis performance. GloVe (Global Vectors for Word Representation) is an unsupervised learning algorithm that generates pre-trained word vectors by analyzing the co-occurrence statistics of words in large text corpora. These embeddings capture the semantic meaning of words in a high-dimensional space, allowing for efficient representation of their relationships and contextual similarities.

GloVe embeddings are utilized in this study to provide a rich, context-aware representation of words in football-related tweets. By initializing the hybrid CNN-LSTM model with these pre-trained vectors, the model can benefit from the semantic knowledge already embedded in the vectors, potentially improving the model's ability to understand and analyze the sentiment in the text. The rationale behind using a hybrid CNN-LSTM model for sentiment analysis lies in the strengths of both convolutional and recurrent layers in processing sequential data like text.

Convolutional layers are adept at detecting local patterns or features within

the input text, such as n-grams or phrases, that can be indicative of sentiment. These local features are often translation-invariant, meaning they can be recognized regardless of their position in the text. On the other hand, LSTM layers are designed to capture long-range dependencies and contextual information by maintaining a hidden state that can store information from earlier tokens. By combining both CNN and LSTM layers, our model can effectively identify local patterns as well as their context, resulting in improved sentiment analysis performance.

Hence, the architecture of the model is outlined below:

Layer	**Description**
1. Pre-trained Glove Embedding Layer	Pre-trained word embedding for mapping input tokens to fixed-size vectors
2. 1D convolutional layer	Identifies local patterns or features within the input text.
3. ReLU Activation Function	Introduces non-linearity into the model using ReLU function.
4. 1D max-pooling layer	Reduces the spatial dimensions of feature maps, selecting the most important features and reducing over-fitting.
5. Bidirectional LSTM layer with dropout	Captures context from both past and future tokens in the input text.
6. Fully connected layer	Combines the features extracted by the previous layers to make a final decision
7. Softmax activation function	Converts the logits into probabilities for each sentiment class, allowing for the prediction of the highest probability sentiment.

Table 3.8: *Hybrid CNN+LSTM Architecture*

In the final stage of the model, the fully connected layer combines the features extracted by the previous layers to make a final decision about the senti-

ment. This decision is based on the patterns and contextual information captured by the convolutional and LSTM layers, taking into account the positive, negative, or neutral aspects of the text.

The softmax activation function in the last layer then converts the logits from the fully connected layer into probabilities for each sentiment class (e.g., positive, negative, and neutral). The model predicts the sentiment class with the highest probability as the final sentiment of the given text. This way, the model categorizes the input text into one of the three sentiment classes based on the features and context identified throughout the layers.

Parameter Tuning and Model Training

In order to optimize the model performance, an extensive search for the best hyper-parameter to optimize the models' performance was done using Optuna [5], an optimization framework. The hyperparameters under consideration included the number of filters, filter size, pool size, LSTM output size, and dropout rate. A total of 96 combinations were executed to identify the most suitable set of hyperparameters. The model was trained and evaluated on the test dataset for each set of hyperparameters, with accuracy scores and hyperparameters recorded for each trial.

The following table presents the hyper-parameters and their respective ranges considered during the optimization process:

The model was evaluated on both validation and test data sets with a respective distribution of 75:15:15. The validation data set was used during hyper-

[5] `https://optuna.org/`

Hyperparameter	Description	Range
No. of filters	Number of convolutional filters	32 to 256 with a step of 32
Filter size	Size of convolutional filters	3 to 7 with a step of 2
Pool size	Size of max-pooling filters	2 to 4 with a step of 2
LSTM output	Number of LSTM output nodes	64 to 512 with a step of 64
Dropout rate	Dropout rate for regularization	0.1 to 0.5 with a step of 0.1

Table 3.9: *Hyperparameters and their respective ranges considered during tuning*

parameter optimization, in which the best hyper-parameters was used to train the model. The model was trained for five epochs, and the loss was calculated using the cross-entropy loss function and the adam optimizer was used to update the model parameters. Ultimately, the performance of the final model was evaluated using the test dataset. Furthermore, an analysis was conducted to determine if there were any variations in the model's efficacy when utilizing stopwords and customized stopwords. The evaluation metrics used were accuracy, confusion matrix, and classification report.

3.6 Data Collection

This chapter aims to outline the methodology of acquiring tweets by employing prominent team-specific hashtags, with a focus on discrete intervals within the football season. This approach will allow to facilitates the understanding of sentiment and discourse evolution throughout the progression of the season.

In order to retrieve the data that will be utilized for the correlation aspect of this study, a set of hashtags representing each team in the Premier League throughout the 2021/20222 season were retrieved. The adoption of these hashtags aimed to minimize the inclusion of irrelevant tweets in our analysis, as suggested

by Schumaker et al. [12] and Kampakis and Adamides [13]. Given that our study is divided into different periods within the 2021-2022 season, the Twitter API proved unsuitable due to its 7-day retrieval limit. Consequently, we employed "snscrape" [6] as an alternative to gather a more comprehensive range of tweets throughout the season.

The hashtags utilized in this research align with those used in studies by Kampakis and Adamides [13] and D. Pacheco et al. [50].

Table 3.10: *Premier League Teams and Hashtags*

Team	Hashtag
Manchester City	#mcfc
Liverpool	#lfc
Chelsea	#cfc
Tottenham	#thfc
Arsenal	#afc
Manchester United	#mufc
West Ham	#whufc
Leicester City	#lcfc
Brighton	#bhafc
Wolves	#wolves
Newcastle	#nufc
Crystal Palace	#cpfc
Brentford	#brentfordfc
Aston Villa	#avfc
Southampton	#southampton
Everton	#efc
Leeds United	#lufc
Burnley	#burnley
Watford	#watfordfc
Norwich City	#ncfc

As a precaution against sarcasm, a manual check of the tweets was conducted to identify any sarcasm-related hashtags, and such tweets were removed accordingly, in accordance with the guidelines outlined in C. Liebrecht et al. [51].

[6] `https://github.com/JustAnotherArchivist/snscrape`

Table 3.11: *Data collection dates for each time period during the 2021/2022 Premier League season*

Dates	**Number of Games**
1st December 2021 - 1st January 2022	240
1st April 2022 - 1st May 2022	240

3.6.1 Sentiment Analysis

Upon the completion of the data collection, a sentiment analysis was executed on the tweets to ascertain the expressed sentiment towards each team, divided on a weekly basis and within two distinct months. The most accurate model from the dataset was utilized in order for the sentiment of the tweets to be predicted in which these were subsequently preserved as a newly-created CSV file. In order to maintain consistency and comparability between the training data set and the collected data, the same pre-processing methods were applied.

The tweets demonstrating neutral sentiment were excluded from the dataset, as they frequently pertained to matters unrelated to the team's performance. The newly generated file encompassed the original tweet text in conjunction with the predicted sentiment for each respective tweet. This procedure facilitated the expeditious analysis and visualization of sentiment trends throughout the specified periods, thereby enabling a correlation with the team's performance to be established.

3.7 Association Between Variables

This final step within the methodology aims to examine the relationship between tweet sentiment and football outcomes, specifically match results and league standings changes. To achieve this, two categorical variables were analyzed using statistical methods, including the Chi-Squared Test and Cramér's V.

Within the framework of our research, a deliberate choice was made to utilize the Chi-Squared Test and Cramér's V instead of correlation analysis methods. The reasoning for this decision is based on the categorical nature of the data under investigation. It is well-established that correlation analysis, such as Pearson or Spearman correlation, is more applicable for continuous or ordinal data. Given that our dataset comprises counts of positive and negative tweets, as well as categorical outcomes such as match results or league standings changes, the Chi-Squared Test and Cramér's V are deemed to be more appropriate techniques for examining the associations between these variables.

The Chi-Squared Test is a non-parametric statistical method used to determine if there is a significant relationship between two categorical variables. A contingency table displays the frequency distribution of categorical variables in a matrix format, showing the observed frequencies of different combinations of categories of two or more variables. The Chi-Squared Test employs a mathematical formula to calculate the deviation between the observed and expected data, resulting in a Chi-Squared test statistic that quantifies the discrepancy between these frequencies.

Apart from the Chi-Squared test statistic, the Degrees of Freedom (DF) play a

critical role in determining the distribution of the test statistic and the significance of the test. The Degrees of Freedom are calculated using the formula:

$$df = (r-1)(c-1) \tag{3.1}$$

Where:

- r: is the number of rows in the contingency table.
- c: is the columns in the contingency table.

The significance of the test is determined by comparing the p-value to a predetermined level of significance, commonly set at 0.05. While the Chi-Squared Test allows us to determine the existence of an association between the variables, it does not provide information about the strength or direction of the relationship. To assess the strength of the association, Cramers V will be used, which is a measure derived from the Chi-Squared statistic.

Cramér's V is a statistic measure that ranges from 0 to 1, used to determine the strength of the relationship between two variables. Cramér's V can be interpreted to determine the strength and direction of the relationship between the variables. It is derived from the Chi-Squared statistic and calculated using the formula:

$$V = \sqrt{\frac{\chi^2}{n \min(r-1, c-1)}} \tag{3.2}$$

Where:

- V: is the value of Cramér's V

- χ^2: is the result of the Chi-Squared Test
- n: is the total number of observations
- r: is the number of rows in the contingency table
- c: is the number of columns in the contingency table

.

To conduct the analysis, two contingency tables were created using the pandas library in Python. Expected frequencies were calculated assuming independence between variables. The chi2-contingency [7] function from the scipy.stats library was used to find degrees of freedom and expected frequencies. The Chi-Squared Test statistic (X^2) and its corresponding p-value were calculated, with a manual calculation performed to validate the accuracy of the code.

	Match Won	Match Drawn	Match Lost
Positive Tweets	45,000	30,000	25,000
Negative Tweets	15,000	20,000	40,000

Table 3.12: *A sample contingency table of tweet sentiment and match outcomes*

	Increase	Decrease	No Change
Positive Tweets	139,000	21,000	45,000
Negative Tweets	24,000	73,000	64,000

Table 3.13: *A sample contingency table of tweet sentiment and changes in standings*

After determining the significance of the association using the Chi-Squared Test, Cramér's V was computed to measure the strength of the association. This

[7] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.chi2_contingency.html`

allowed us to discern the degree to which tweet sentiment is correlated with match outcomes or alterations in league standings.

3.8 Conclusion

In this chapter, various approaches to sentiment analysis employing distinct preprocessing techniques and configuration have been explored. By utilizing hyperparameter tuning, an extensive array of combinations was also discussed to ascertain the most effective method for achieving optimal performance in sentiment analysis. Finally, the statistical models employed for evaluating the relationships between football-related tweets and the corresponding sentiment scores, while considering diverse criteria and time frames has also been discussed.

In the forthcoming chapter, we shall provide a comprehensive presentation and analysis of the results derived from the evaluation of all tested models. This will facilitate the identification of the best-performing model for sentiment analysis in the context of football-related tweets. Additionally, we will scrutinize the hypotheses outlined in Section 2, determining the extent to which they hold true in light of the empirical evidence obtained through our analyses.