

Generating Sequences of Play In Rugby Union Using Recurrent Neural Networks



Presented by:

Thomas Edley
EDLTHO001

Graham Davies
DVSGRA012

Supervisor:

Mr Neil Watson

Submitted to the Department of Statistical Sciences at the University of Cape Town in partial fulfilment of the academic requirements for a Bachelor of Science Honours in Statistical Sciences

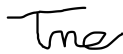
November 15, 2021

Keywords: Recurrent Neural Networks, Long Short-Term Memory, Rugby Union, Sequence Generation

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

Signature:
T. Edley



Signature:
G. Davies



Date: November 15, 2021

Acknowledgements

Foremost, we would like to acknowledge and give thanks to our supervisor, Neil Watson, for his continuous support throughout the writing of this study. His patience, advice and passion for the subject guided us throughout the entire process, allowing us to complete and present a final thesis that we are incredibly pleased with.

Secondly, we would like to extend our gratitude to the University of Cape Town's Statistical Department and to all the lecturers who have taught us the valuable information needed to complete this study and have accommodated our busy schedule.

Abstract

Robust statistical analysis of sports data is becoming increasingly important as the availability of data amenable to such analysis grows. Most sporting analysis is based on aggregate performance measures over a fixed time interval, most of which are univariate and isolated. The sequential nature of sports is often overlooked [1]. Studies have used the sequential nature of sport to predict the outcome of games in various sports, but very few have sought to generate realistic domain-appropriate sequences. Due to the sequential nature of rugby union, there is an opportunity to investigate whether statistical models can be used to generate realistic sequences of play. The primary aim of this project is to determine whether one can use recurrent neural networks (RNN), particularly long short-term memory (LSTM) networks, to generate realistic sequences of play. Further, the sequences generated by winning and losing teams are investigated to determine if significant differences exist.

Contents

1	Introduction	1
1.1	Background	1
1.2	Objectives	2
1.3	Scope and Limitations	3
1.4	Report Outline	3
2	Literature Review	5
2.1	Sequence Generation	5
2.2	Sport Related Studies	6
2.3	Statistical Analysis of Rugby Union	7
3	Methods	9
3.1	Recurrent Neural Networks	9
3.2	Long Short-term Memory Networks	10
3.2.1	LSTM Steps	11
4	Data	15
4.1	Data Collection	15
4.2	Data Preparation	15
4.2.1	Winning and Losing Teams	20
5	Model Fitting	22
5.1	Model Specifications	22
5.1.1	Validation	22

5.1.2	Dense Layer (Fully-connected layer)	22
5.1.3	Activation Function	22
5.1.4	Loss Function	23
5.1.5	Optimizer	23
5.1.6	Epochs, Stopping Criterion and Patience	24
5.1.7	Temperature	25
5.1.8	Batch Size and Units	25
5.1.9	One-hot Encoding	25
5.2	Measures of Performance	26
5.3	Model Design	26
5.4	Hyperparameter Optimization	28
5.4.1	Single-layer LSTM	28
5.4.2	Two-layer Stacked LSTM	29
6	Results	30
6.1	Model Evaluation	30
6.1.1	Loss Value	30
6.1.2	Temperature	31
6.1.3	Distribution of Actions	31
6.1.4	Distribution of Next Actions	34
6.1.5	Common Sequences	40
6.1.6	Best Performing Model	43
6.2	Winning versus Losing Teams	44
6.2.1	Performance comparison	44
6.2.2	Game Simulation	48

7	Discussion	51
7.1	Results Summary	51
7.2	Future Considerations	52
7.3	Future Applications	52
7.4	Limitations	53
7.4.1	Cloud Computing	53
7.4.2	Time Constraints	53
8	Conclusion	54
A	Examples of Sequences Generated	60
A.1	Single-layer LSTM Sequences	60
A.2	2-layer stacked LSTM Sequences	61
B	Link to GitHub Repository	62

List of Figures

1.1	Rugby union field dimensions.	2
3.1	Structure of an RNN chunk.	9
3.2	Repeating module of a standard RNN.	10
3.3	Repeating module of an LSTM.	11
3.4	First step of an LSTM.	12
3.5	Second step of an LSTM.	12
3.6	Third step of an LSTM.	13
3.7	Fourth step of an LSTM.	14
4.1	Distribution of action variables in the data set.	16
4.2	Example of a sequence of actions in a possession.	17
4.3	Example of multiple sequences.	18
4.4	Boxplot of the sequence lengths before outlier removal.	19
4.5	Boxplot of the sequence lengths after outlier removal.	19
4.6	Flow diagram illustrating how attacking sequences are classified.	20
4.7	Distribution of actions between winning and losing teams	21
5.1	Implementation of early stopping criterion.	24
5.2	Model architecture.	27
6.1	Histogram comparing the distribution of actions from Model 1.3 to the original data set.	32
6.2	Histogram comparing the distribution of actions from Model 1.9 to the original data set.	33

6.3	Histogram comparing the distribution of actions from Model 2.9 to the original data set.	33
6.4	Histogram comparing the distribution of actions from Model 2.15 to the original data set.	34
6.5	Histogram comparing the distribution of actions from four chosen models.	35
6.6	Distribution of next actions after attacking quality action from the original data set.	36
6.7	Distribution of next actions after attacking quality action for the four chosen LSTM models.	37
6.8	Distribution of next actions after possession action from the original data set.	38
6.9	Distribution of next actions after possession action for the four chosen LSTM models.	38
6.10	Distribution of next actions after penalty conceded action from the original data set.	39
6.11	Distribution of next actions after penalty conceded action for the four chosen LSTM models.	40
6.12	Distribution: Winning teams vs. Losing teams.	44
6.13	Box Plot of Length of Sequences by Winning and Losing Teams.	45
6.14	Actions before attacking qualities: Winning teams vs. Losing teams. . . .	46
6.15	Actions after attacking qualities: Winning teams vs. Losing teams. . . .	47
6.16	Actions before a kick: Winning teams vs. Losing teams.	47
6.17	Actions before a kick: Winning teams vs. Losing teams.	48

List of Tables

4.1	Action Numerical ID and their associated action name.	16
4.2	Sequence lengths summary statistics.	18
5.1	Original sequence.	25
5.2	One-hot encoded sequence.	26
5.3	Model specifications	27
5.4	Single-layer LSTM hyperparameter results.	28
5.5	Two-layer Stacked LSTM hyperparameter results.	29
6.1	Models with lowest loss values.	30
6.2	Attacking quality next actions failed to be generated, or incorrectly generated, by the four chosen LSTM models.	36
6.3	Possession next actions failed to be generated, or incorrectly generated, by the four chosen LSTM models.	39
6.4	Penalty conceded next actions failed to be generated, or incorrectly generated, by the four chosen LSTM models.	40
6.5	Most frequently occurring sequences in original data set.	41
6.6	Most frequently occurring generated sequences using Model 1.3.	41
6.7	Most frequently occurring generated sequences using Model 1.9.	41
6.8	Most frequently occurring generated sequences using Model 2.9.	41
6.9	Most frequently occurring generated sequences using Model 2.15.	42
6.10	Most frequently occurring long sequences in original data set.	42
6.11	Most frequently occurring long sequences using Model 1.3.	42
6.12	Most frequently occurring long sequences using Model 1.9.	42
6.13	Most frequently occurring long sequences using Model 2.9.	43

6.14	Most frequently occurring long sequences using Model 2.15.	43
6.15	Length of Sequences Summary Statistics.	45
6.16	Goal kick success rates from data set.	49
6.17	Results of winning team model vs. losing team model.	49

Chapter 1

Introduction

1.1 Background

The sport of rugby union involves two teams of 15 players, each competing for the possession of the ball and trying to move closer to the opposition in-goal area, situated at each end of the field. Figure 1.1 displays a standard rugby union field. A game of rugby lasts 80 minutes, divided into equal halves of 40 minutes each. The goal of each team is to achieve a higher points tally than their opponents at the end of the game. Points are earned by placing the ball in these in-goal areas, known as scoring a ‘try’, or by taking ‘penalty kicks’, post-try ‘conversions’, and ‘drop goals’, where a player will attempt to kick the ball through the posts situated in the middle of the oppositions in-goal area.

Teams with possession move the ball through sequences of actions known as phases. These phases are sequences of play that involve actions performed by the players on each team in an attempt to move closer towards the opposition in-goal area. Each phase can consist of multiple actions. Phases typically occur between ‘breakdowns’. A breakdown occurs when a player is tackled to the ground by a player on the opposition team. After being brought to the ground, multiple players from each team compete for the ball in a situation that is known as a ‘ruck’. The movement of the ball from the ruck indicates the start of a new phase. A change in possession from one team to the other also indicates the beginning of a new phase; however, a possession can consist of multiple phases. The actions in a game of rugby can be assessed sequentially, this being the basis for this study.

Technology advancements in recent times have resulted in a surge in the availability of data. With large amounts of data available, the use of data analytics in sport has become increasingly popular. The sports industry uses data analytics to increase revenue, improve player and team performance and prevent injuries, among other enhancements. Sports analysts are currently in high demand, with many teams now developing departments with the sole purpose of analyzing statistics. With the technology available, sports analysts can use data to create insightful visualizations to communicate to the key decision-makers in a team [2]. The development of sports analytics has also been beneficial to betting companies. Statistical techniques can be used to calculate a team’s chance of winning based on the history of a team’s players, coach, type of play and past matches [3].

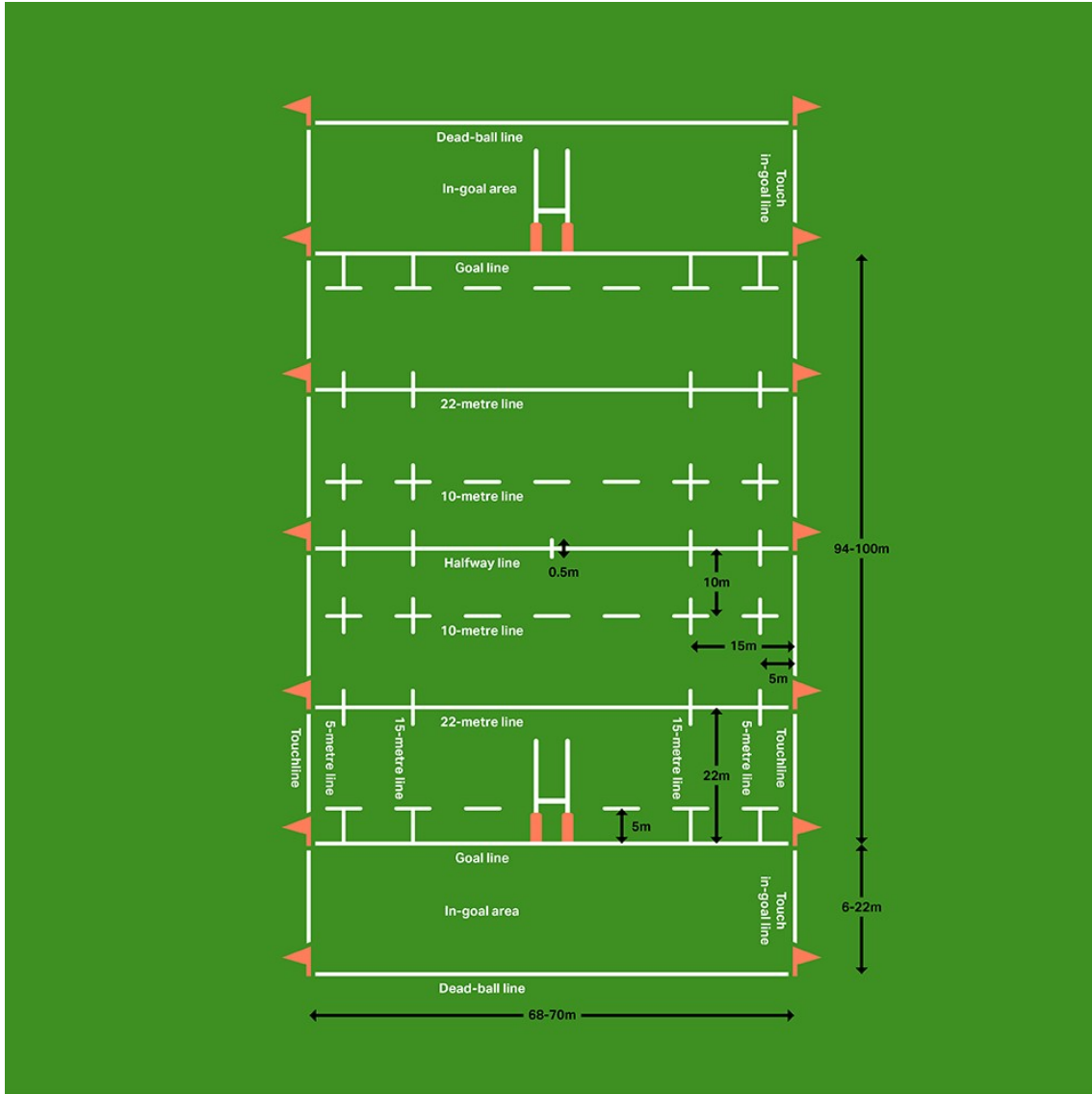


Figure 1.1: Rugby union field dimensions [4]. Teams attempt to move the ball forward in an effort to score points at the in-goal areas located at the two ends of the field.

1.2 Objectives

This study aimed to achieve two objectives:

Primary Objective: Determine whether RNNs, more specifically LSTMs, can be used to generate realistic sequences of play for rugby union.

Secondary Objective: Use an LSTM model to test for differences in the sequences of play from winning and losing teams. This objective will be based on the differences in attacking sequences between winning and losing teams.

1.3 Scope and Limitations

Running an LSTM model is computationally difficult and expensive. To reduce the complexities of running these models, the models were fitted on a Rstudio Amazon Web Server ‘g4ad.xlarge’ instance with 4 virtual vCPUs, a Nvidia T4 GPU, 16 GB of RAM, and a network performance of up to 10 GB [5]. The Amazon Web Server instance charges an hourly rate. The budget for this study was limited to ZAR1500. The time constraint of running LSTM models meant that only a limited number of neural network architectures could be evaluated. This study only assessed LSTM networks in achieving the objectives.

The data provided for this study was limited to rugby competitions that took place between the years 2013 and 2015. Recent rugby data has the potential of being more relevant for future applications. However, this did not affect the objective of the study.

The primary objective of this study was to generate realistic sequences of actions in a game of rugby union and not to predict sequences. For this reason, there was no need to split the data into training, testing and validation sets. The only statistical measures that could be used to assess the results were the loss functions of LSTM models and histograms assessing the distribution of actions in generated sequences. The primary method of evaluating the accuracy of the sequences generated was using domain knowledge of rugby union to assess whether generated sequences were realistic or not.

1.4 Report Outline

In Chapter 2, the study gives an overview of previously published literature in the domain of this study. Section 2.1 reviews the use of RNNs and LSTMs for sequence generation, and Section 2.2 assesses literature regarding the use of these models to generate sequences in the domain of sport. Section 2.3 reviews previous studies focusing on the statistical analysis of rugby union.

Chapter 3 describes the statistical models used in this study. Section 3.1 and Section 3.2 review the theory and applications of RNNs and LSTMs, respectively.

Chapter 4 describes the data and data preparation that is used in the study. Section 4.1 describes the data collection process. Section 4.2 provides an in-depth explanation of how the data is prepared for the models.

Chapter 5 gives an overview of how the data has been modelled using LSTM networks to achieve the desired outcome.

Chapter 6 describes the results of this study. Section 6.1 evaluates various criteria in order to assess the performance of the LSTM models in generating realistic sequences of play. Section 6.2 gives the results of the secondary objective, which was to evaluate the differences in sequences generated between winning and losing teams.

Chapter 7 will provide thoughts for future research into the use of neural networks in rugby. This chapter will also speak to some of the issues that the study came across and what can be improved. Areas where we feel that this study will be most applicable will also be discussed.

Chapter 8 will provide a conclusion to the study, briefly summarizing the results and whether the objectives of the study were achieved or not.

Chapter 2

Literature Review

2.1 Sequence Generation

RNNs are a rich class of dynamic models that have become useful in generating sequences in various domains, including speech recognition [6], language modelling [7], translation [8], image captioning [9] and more. Graves [10] wrote the seminal paper on RNNs, in which he uses an LSTM model to develop predictions for handwriting. Positional ‘XY’ data of handwriting taken from a smart whiteboard was used to train an LSTM network without pre-processing the collected data. This model could predict the next letter or word and generate sequences based on different initial starting points. Another area where LSTM sequence generation has been used is human trajectory prediction, which refers to predicting human movements based on past positional ‘XY’ data. Alahi et al. [11] trained an LSTM model that was able to learn general human movement and predict their future trajectories. They found that the LSTM model outperforms state-of-the-art prediction methods by more than 42%.

The most prominent use of LSTMs is in the domain of text generation. LSTMs are suitable for analyzing sequences of text data and predicting the next word or letter. Islam et al. [12] proposed an artificial ‘Bangla’ (the Bangladesh national language) text generator using an LSTM. Their neural network was trained using Bangla newspapers spanning 917 days, with a daily newspaper containing on average 4500 sentences. They found that the LSTM model was able to accurately predict sequences of words based on this data. Mangal et al. [13] present how different sequence to sequence deep learning models perform when generating new conversations between characters and new scenarios based on scripts from TV series. They compared the results of three memory models: LSTMs, Gated Recurrent Units (GRU), and Bidirectional RNNs. Each model was designed to learn the sequence of recurring characters from input sequences taken from TV scripts. They concluded that the LSTM generates text the most efficiently out of these models and takes the least time to run.

From Karpathy’s blog [14], the effectiveness of RNNs for sequence generation is demonstrated. Karpathy states that the limitation of standard feed-forward Vanilla Neural Networks (and Convolutional Neural Networks) is that their API (Application Programming Interface) is too constrained. This constraint refers to how these models only accept a fixed-sized vector as an input and produce a fixed-sized vector as an output. Addition-

ally, Convolutional Neural Networks are limited by a fixed amount of computational steps, such as a fixed amount of layers. This study further demonstrates the capabilities of RNNs for text generation, using RNN models to generate text based on ‘Paul Graham’ essays, Shakespeare writings, Wikipedia pages, and baby names.

The automatic generation of sequences has been highly explored in the domain of automatic music composition in recent times. Garcia-Valencia et al. [15] analyze the effect of using three different RNN memory mechanisms to generate musical note sequences. These mechanisms were Neural Architecture Search (NAS), Update Gate Recurrent Neural Networks (UGRNN), and LSTMs. Their results concluded that when using LSTMs, the instances where there are consecutive repetitions of notes are minimal, which indicates that the LSTM cell shows an excellent learning capacity to keep the scale of musical notes when generating sequences.

2.2 Sport Related Studies

RNNs and LSTMs are becoming increasingly useful for statistical analysis in sports because of their effectiveness in sequence generation problems. Sequences of actions have primarily been used to predict the outcome of matches [16], short passages of play [17] and physical movements of players or the ball [18].

A study by Verpalen [19] used an LSTM and GRU to predict player movement in soccer using ‘XY’ positional data. The study focused on predicting player movement via object tracking from visual data such as images and videos. The results show that LSTMs are well capable of predicting the next change in a player’s position. However, on longer input sequences, the predictive performance decreases. They concluded that both LSTMs and GRUs are suitable deep learning techniques for predicting movement in a game of soccer, with the LSTM being the most effective model and resulting in lower metric errors on almost all of their experimental results. One of the key outcomes of this study is that they were able to predict the next movement of individual players one at a time and predict the successive positions of all the players on the field simultaneously.

Romijnders and Shah [18] used LSTMs for sequence generation to predict whether a three-point shot in a professional basketball game will be successful. They trained models to learn the trajectory of a basketball without any knowledge of the physics behind it. The model results were compared to those of a baseline static machine learning model with a complete set of features, such as angle and velocity of the shot, in addition to positional ‘XY’ data. They found that LSTM models based simply on sequential positional data

outperform a static feature-rich machine learning model in predicting whether a three-point shot will be successful or not.

Goddijn et al. [20] compared three techniques to predict the outcomes of football matches. They used simple logistic regression, a 3-layer standard feed-forward neural network, and an LSTM. The feed-forward neural network was found to perform best with an accuracy metric of 51%, even though LSTMs could take into account historical data. The LSTM model was found to have an accuracy metric of 47.34% after adjustments for overfitting. They suggest that future considerations for the study should include looking at different architecture types and LSTM variations.

2.3 Statistical Analysis of Rugby Union

Most research on neural networks in sports focus on soccer, American football and basketball. There has been little coverage of statistical analysis for rugby union. The sequential nature of rugby presents an excellent opportunity to apply these methods to the sequences of actions in a game of rugby.

A majority of the statistical studies done on rugby union have focused on outcomes based on key performance indicators (KPIs) [21, 22]. Watson et al. [23] conducted a validation study to assess which historically significant team KPIs in rugby union continue to be valid differentiators between winning and losing teams. Of the 69 KPIs evaluated, they found that only 12 remained significant based on their results. They also pointed out that the current univariate form of KPIs does not capture the complexities of rugby. They state that more complex statistical techniques should be used to assess the performance of rugby union.

The use of neural networks in the domain of rugby is undeveloped. Watson et al. [1], who use the same data as this project, used convolutional and recurrent neural networks to predict specific outcomes of sequences of play, based on the order of the phases and the position in the field where the action occurs. The performance of these models were compared to that of a baseline Random Forrest model. They used these results to investigate how their model could provide tactical decision-making support to rugby coaches. This study showed that when the sequence of actions and position of these actions on the field are used as input for a convolutional and recurrent neural network, prediction is more accurate than the baseline model. They demonstrated that the sequential nature of rugby union is an essential factor in the assessment of team performance.

Based on the literature review for this study, the study by Watson et al. [1] was the

first to consider the sequential nature of rugby union in predicting intra-game outcomes. This project draws many similarities from their study; however, this study will use the sequential nature of rugby union to generate realistic sequences of play. From previous research, there are currently no studies that have attempted to investigate this outcome, which therefore makes this project an exciting and unique experiment.

Chapter 3

Methods

3.1 Recurrent Neural Networks

Recurrent neural networks are a rich class of dynamic models that can be used to model sequential data. It is the first algorithm that can remember its past inputs, which makes it very useful for machine learning problems involving sequential data. RNNs are similar to standard feed-forward neural networks, except that their output distribution is additionally influenced by internal memory. Sequences are generated from a trained network by iteratively sampling from the output distribution. These samples are then fed into the network as input at the next step, such that the network is taking in new input and input from previous iterations [10]. Figure 3.1 displays a chunk of a neural network, A . A looks at some input x_t and outputs some value h_t . The loop in the structure allows information to be passed from one step of the RNN to the next step, such that the output is influenced by the input at each step as well as the internal memory.

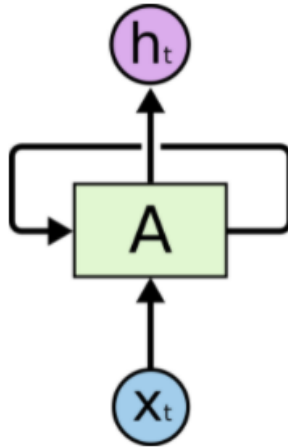


Figure 3.1: Structure of an RNN chunk, A [24]. h_t refers to the hidden layer output and x_t refers to the input.

All RNNs have the form of a chain of repeating modules of a neural network. Figure 3.2 displays the structure of the repeating module for a standard RNN. The module has a straightforward structure consisting of one \tanh neural network layer. The \tanh layer regulates the values going through the network. It does this by ensuring that these values stay between -1 and 1 [25]. At each repeating module (or step) of an RNN, input x is

fed into the model, and output in the form of a hidden layer h is fed out. The internal memory is the information that is passed from one repeating module to the next.

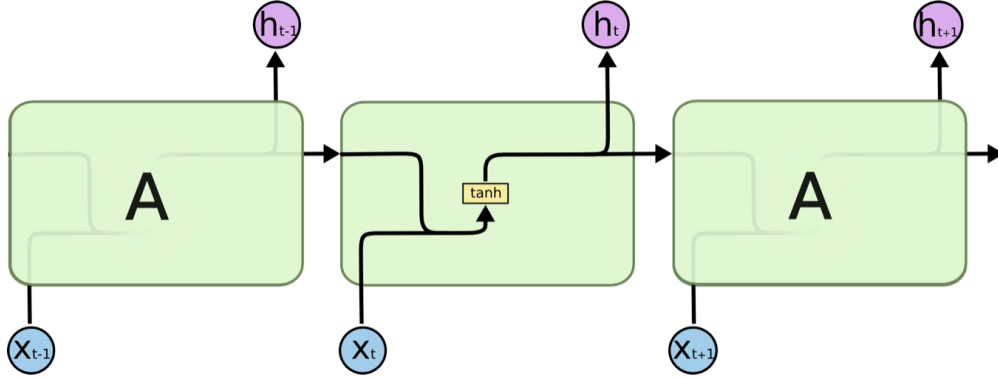


Figure 3.2: Repeating module of a standard RNN. This module contains a single *tanh* neural network layer. [24]

3.2 Long Short-term Memory Networks

Standard RNNs are unable to store information about past inputs for an extended period of time. LSTMs are extensions of RNNs that are capable of learning long-term dependencies. They were introduced by Hochreiter and Schmidhuber [26] and have been widely used since then in many domains. LSTMs are better at storing and accessing information than standard RNNs [10]. Like RNNs, LSTMs have a chain-like structure; however, the repeating module of an LSTM has a different structure compared to a standard RNN. Figure 3.3 displays the repeating module of an LSTM network. From Figure 3.3, one can note that the repeating module of an LSTM network has four interacting neural network layers, as opposed to RNNs, which have one *tanh* layer. These four layers are what allow LSTMs to remember information for an extended time period.

The horizontal line that runs through the top of the LSTM repeating module in Figure 3.3 is known as the cell state, C_t , and is the key element of the LSTM. Information is added or removed from the cell state depending on the nature of the structures known as gates. The cell state can be thought of as the ‘memory’ of the network [25]. Gates allow LSTMs to let information through the cell state optionally. These gates are composed out of a sigmoid neural net layer and a pointwise multiplication operation [24]. They are indicated by the yellow blocks with σ symbols in them in Figure 3.3. The sigmoid layer, σ , outputs numbers between 0 and 1, indicating how much of each component should be let through. A value within a certain threshold of 0 indicates that nothing should be let through. A value within a certain threshold of 1 indicates that everything should be let through. An

LSTM has three of these gates; a forget gate, an input gate and an output gate.

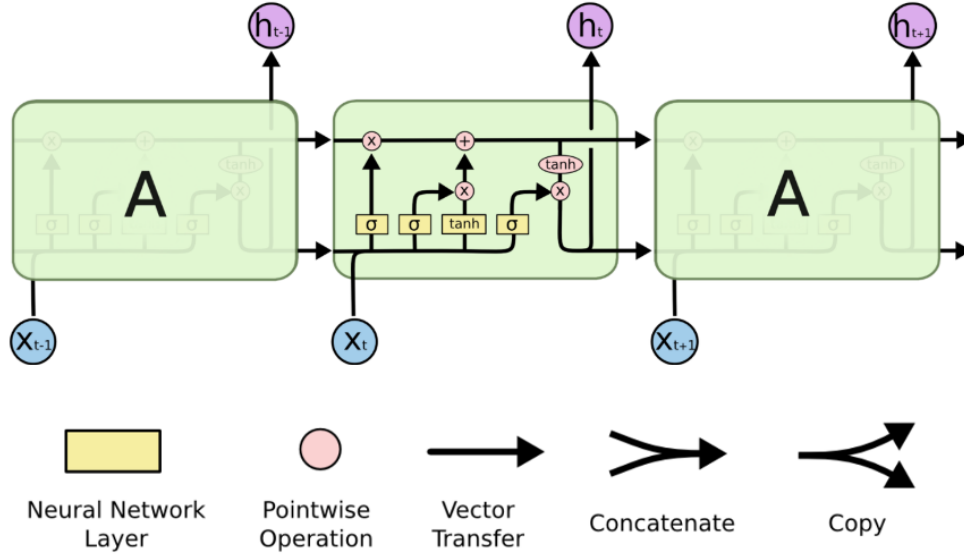


Figure 3.3: Repeating module of an LSTM. This module contains four interacting neural network layers [24].

3.2.1 LSTM Steps

The first step of an LSTM is visualized in Figure 3.4. The sigmoid layer of the forget gate, f_t , decides what information will be deleted or kept. Information from the previous hidden state, h_{t-1} , and information from the current input, x_t , is passed through the sigmoid layer, which converts the input to values between 0 and 1. If a value is within a certain threshold from 0, it will be forgotten, and if it is within a certain threshold distance from 1, it will be kept. The forget gate f_t can be written in the form

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (3.1)$$

where W_f refers to the weight matrix for the forget gate and b_f refers to the bias term. The next step of the process, displayed in Figure 3.5, decides what new information will be stored in the cell state. The input gate layer, i_t , takes in the previous hidden state, h_{t-1} , and current input, x_t , and passes it through a sigmoid layer which decides which values will be updated. A value within a certain threshold of 1 means that value is essential, and a value within a threshold of 0 means it is not essential. Next, the hidden state and current input are passed through a \tanh layer to regulate the information going through the network.

This step creates a vector of new candidate values, \tilde{C}_t , that could be added to the cell state. The following equations define this step of the process:

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \end{aligned} \quad (3.2)$$

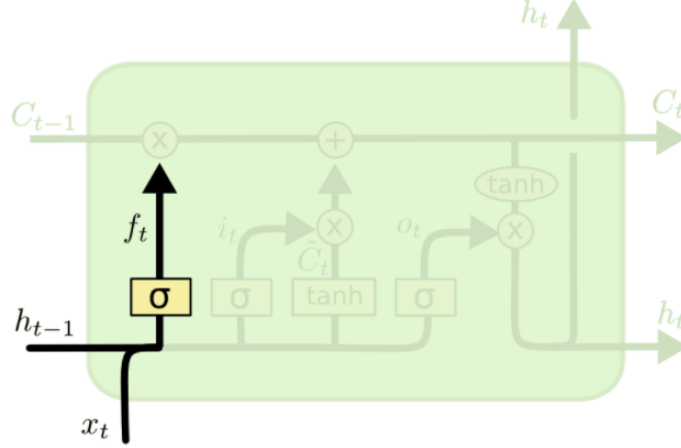


Figure 3.4: First step of an LSTM. The forget gate f_t decides what information is deleted or added to the cell state C_t . [24]

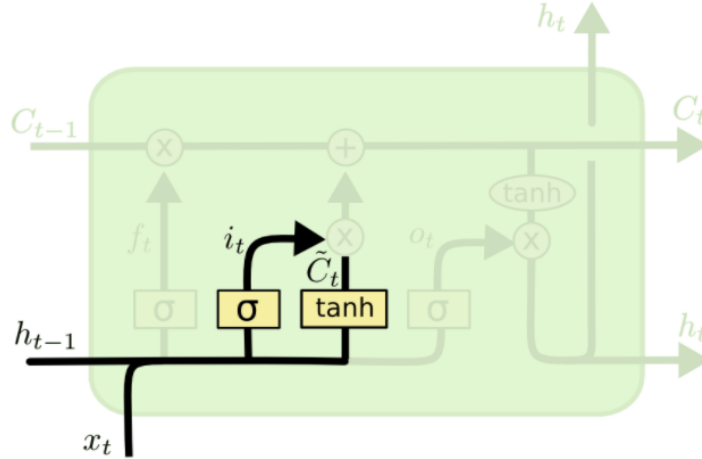


Figure 3.5: Second step of an LSTM. The input gate, i_t , decides what new information will be stored in the cell state, C_t [24].

The next step, displayed in Figure 3.6, updates the old cell state, C_{t-1} , from the previous repeating module into the new cell state, C_t . The old cell state is multiplied by the information from the forget gate, f_t , which omits the information that the network has

chosen to forget. The new candidate values from the input gate, i_t , are multiplied by \tilde{C}_t and added to the old cell state. The following equation defines this step:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (3.3)$$

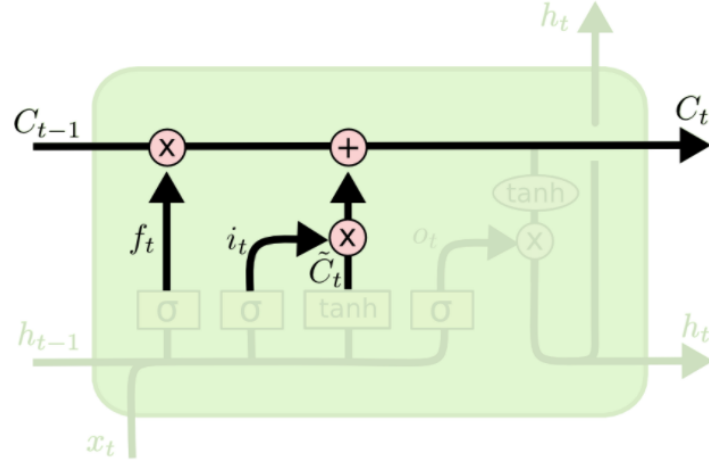


Figure 3.6: Third step of an LSTM. The old cell state, C_{t-1} , is updated to the new cell state, C_t [24].

The final step of the LSTM process is displayed in Figure 3.7. At this stage, the LSTM decides the values of the output and the next hidden state, h_t . The output gate, o_t , decides what the next hidden state should be. The hidden state contains information from previous inputs and is what is used for predictions. The previous hidden state, h_{t-1} , and current input, x_t , are passed through a sigmoid layer. This sigmoid layer decides what parts of the cell state (C_t) will be the output. The newly modified cell state is then passed through a \tanh function, and the output from this is multiplied by the sigmoid output to decide what information the hidden state should consist of. The new cell state, C_t , and the new hidden state, h_t , are then passed through to the next repeating module, or next time step. The following equations define this step:

$$\begin{aligned} o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\ h_t &= o_t * \tanh(C_t) \end{aligned} \quad (3.4)$$

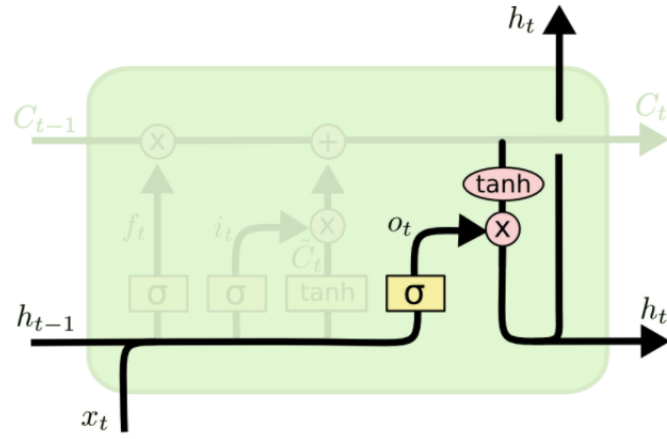


Figure 3.7: Fourth step of an LSTM. The output gate o_t decides what information will be added to the hidden state, h_t [24].

Chapter 4

Data

The data set used in this study consisted of five different rugby competitions, namely, the Heineken Cup, European Rugby Championship, Super Rugby, Six Nations and Rugby Championship. The data set spans 313 games. The years and seasons vary between competitions but range from 2013 to 2015. The data is extensive and contains 485502 observations with 23 variables for each observation. One of these variables is the ‘action’ variable. This variable describes what action is performed by a particular player at the time the observation was recorded and is recorded sequentially. The action variable was the focus of this study. It is these actions that form the sequences that were provided to the model as input. The sequences obtained from the data set were based on possession of the ball, such that one sequence represents a period of play where a particular team had the ball, with the sequence ending with a change in possession. Henceforth, sequences representing a possession will simply be termed sequences.

4.1 Data Collection

The data was provided by the supervisor of this study and originated from ‘Opta’, a sports analytics company based in London, England. Two teams of two Opta analysts each collected the data. The post-match screening was performed by another two analysts, which involved numerous accuracy checks. These analysts require training for three to six months before coding live games, and additionally, Opta monitors each analyst’s accuracy throughout the season by performing regular accuracy checks.

4.2 Data Preparation

This study was only concerned with the generation of sequences consisting of the action variable. Therefore, this variable was extracted from every game in the data set. The action variable consists of numerical values ranging from 1 to 29, where each numerical value indicates a different type of action. There was no action ‘25’ in the data set. A summary of these actions, and descriptions of what each action means, are shown in Table 4.1 below:

ID	Action	ID	Action	ID	Action	ID	Action
1	Carry	8	Turnover	15	Possession	22	Clock
2	Tackle	9	Try	16	Other	23	Ruck
3	Pass	10	Attacking Qualities	17	Period	24	Maul
4	Kick	11	Goal Kick	18	Collection	26	Sequence
5	Scrum	12	Missed Tackle	19	Team Play	27	Lineout Take
6	Lineout Throw	13	Turnover Won	20	Ref Review	28	Offensive Scrum
7	Penalty Conceded	14	Restart	21	Card	29	Defensive Scrum

Table 4.1: Action Numerical ID and their associated action name.

After extracting the actions from the data set, the distribution of actions can be visualized using a histogram that displays the frequency of each action in the original data set. The relative frequency of each action is indicated by the proportion of that action out of all recorded actions. This plot is displayed in Figure 4.1. It was observed that there were no instances of actions ‘13’, ‘16’, ‘19’, and ‘22’ in the data set.

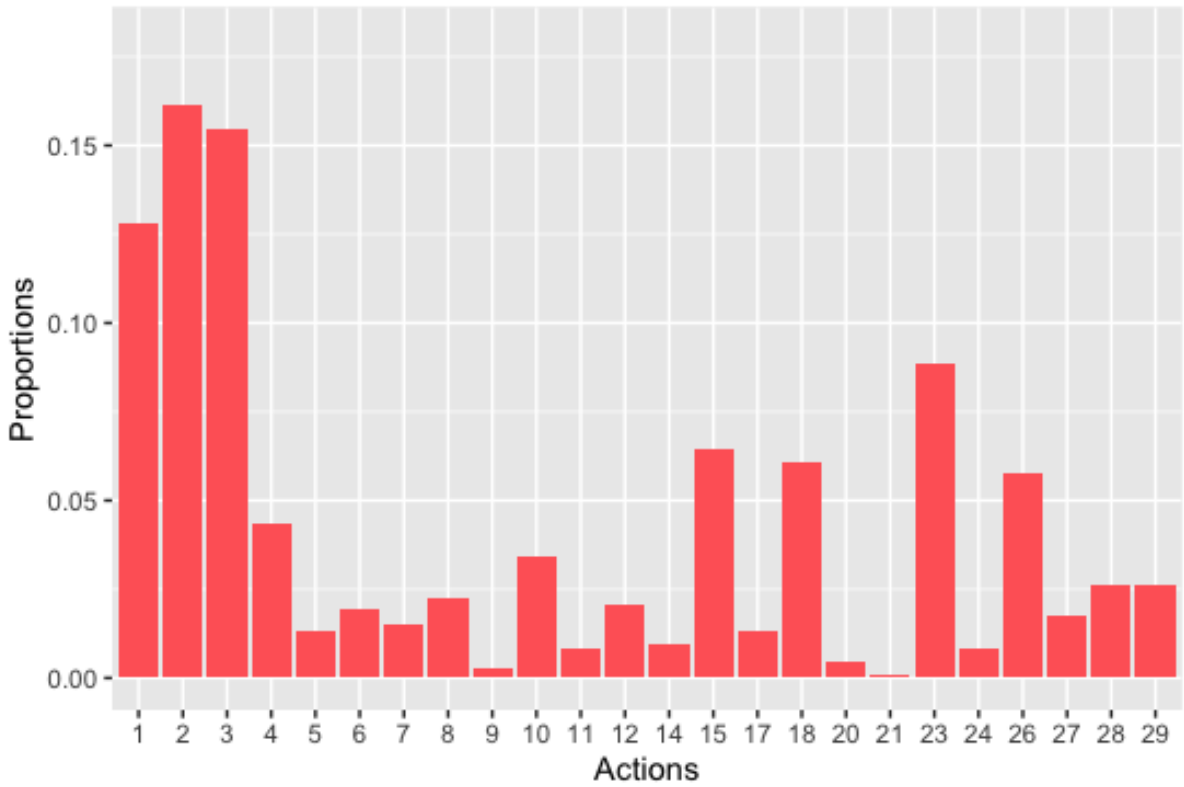


Figure 4.1: Distribution of action variables in the data set.

From Figure 4.1 it is evident that the most prominent actions were actions ‘1’, ‘2’ and ‘3’. These numerical values translate to a ‘carry’, ‘tackle’ and ‘pass’. A carry is when a player has possession of the ball and attempts to move forward. A tackle is an action performed by a player on the team without possession in which that player attempts to bring down the player with the ball. A pass occurs when the player with the ball attempts to throw

the ball to another player on their team. These actions are the most prominent in a game of rugby, so it was expected that their proportions were high. The least frequent variables were actions ‘21’, which translates to a ‘card’, and ‘9’, which translates to a ‘try’. A card refers to situations where a player is sent off for 10 minutes (yellow card) or the entire game (red card) for performing an illegal action. A try involves the player with the ball running into the opposition in-goal area and placing the ball down, resulting in 5 points being allocated to the team who scored the try. Actions ‘28’ and ‘29’, which translate to ‘offensive scrum’ and ‘defensive scrum’, had equal proportions, as with every offensive scrum from the attacking team, there is also a defensive scrum from the defending team. A scrum is a type of set-piece action used to restart play, and occurs when a player drops the ball forward, known as a ‘knock-on’.

The extracted actions from the data set were initially in the form of one whole vector of numerical values that needed to be split into sequences such that the data was of the form of the required input shape for the training of an LSTM model. As previously mentioned, the sequences of interest represent the possessions of the ball by a team. Therefore, the actions were split into sequences on action number ‘15’, which indicates the change in possession from one team to the other. Hence, the ending action of each sequence was action ‘15’.

Figure 4.2 displays an example of a sequence. The top row of green blocks represent the numerical action values from the data set, and the red block indicates a change in possession action, the last action in every sequence. The bottom row describes the translation of what each numerical action means. This sequence describes a situation in a game where a player collects the ball and passes it to their teammate, who carries the ball forward and kicks it to the opposition team, resulting in a change of possession and the end of a sequence. Figure 4.3 displays multiple sequences and how a change in possession splits them.

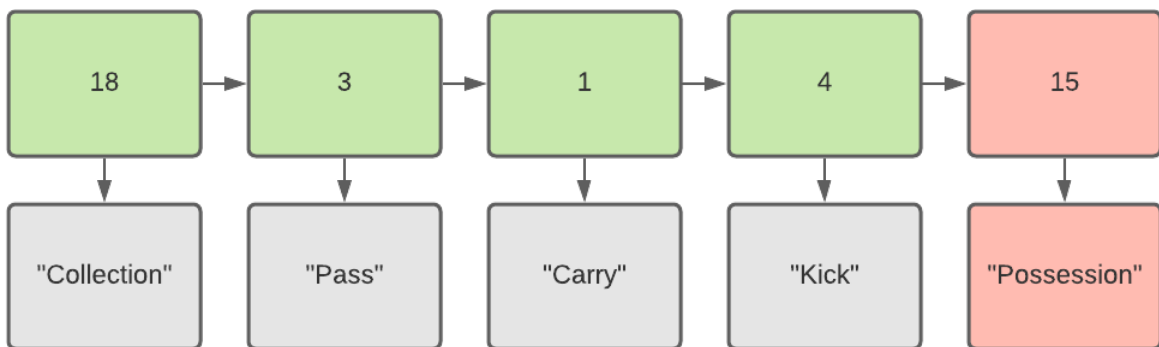


Figure 4.2: Example of a sequence of actions in a possession.

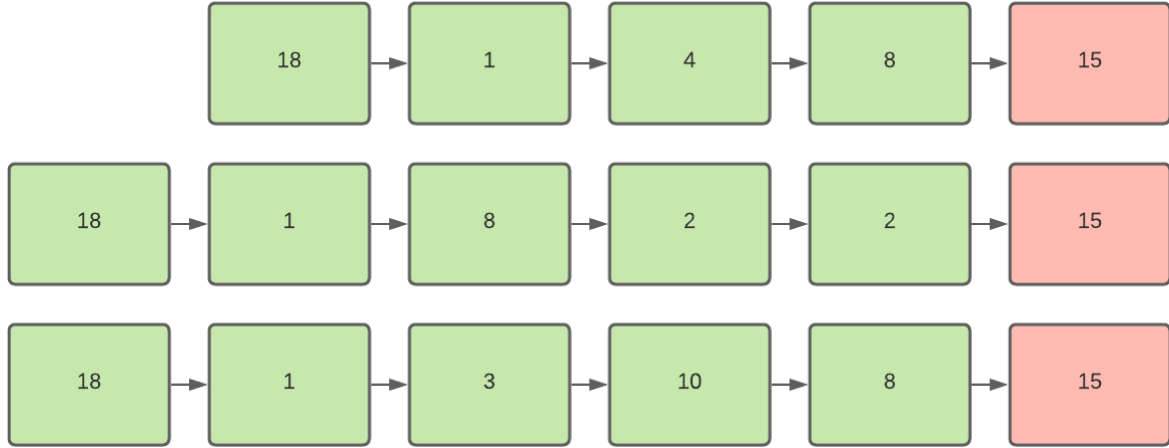


Figure 4.3: Example of multiple sequences.

After splitting the action vector into sequences, there were 26457 sequences of varying lengths. The longest sequence length was 176, and the shortest sequence length was 1. Figure 4.4 visualizes the distribution of sequence lengths using a boxplot. There were a large number of outlying sequence lengths. To improve the accuracy and training times of the model, the sequences of outlying lengths were removed, as well as sequences of length 1. Single action sequences prohibit the assessment of actions that precede or succeed that action. After the removal of these sequences, there were 25005 remaining sequences. Figure 4.5 displays the updated distribution of sequence lengths. The summary statistics for the lengths of these sequences are shown in Table 4.2. The new maximum sequence length was 49, and the average sequence length was 15.58.

Minimum	2
1st Quartile	7
Median	13
Mean	15.58
3rd Quartile	21
Maximum	49

Table 4.2: Sequence lengths summary statistics.

Each sequence then needed to be expanded into subsequences such that each next action could be predicted. This was done by splitting each sequence into cumulative subsequences. For example, a sequence (18, 1, 4) becomes subsequences (18), (18, 1) and (18, 1, 4). To train an LSTM for sequence generation, the input sequences are required to be of the same length. Therefore, sequences were pre-padded with zeros such that all sequences were of the same length. The length specified is the maximum length of the sequences, which was 49. After this, there were 389591 sequences of length 49. These

were the final sequences that were fed into the LSTM model as input.

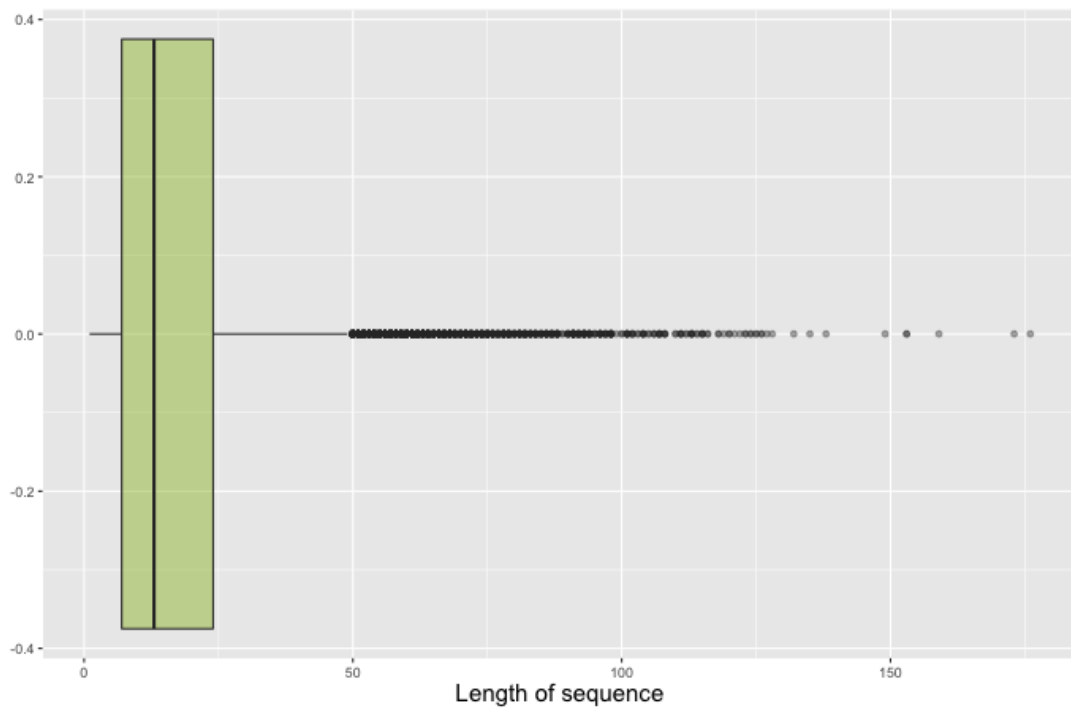


Figure 4.4: Boxplot of the sequence lengths before outlier removal.

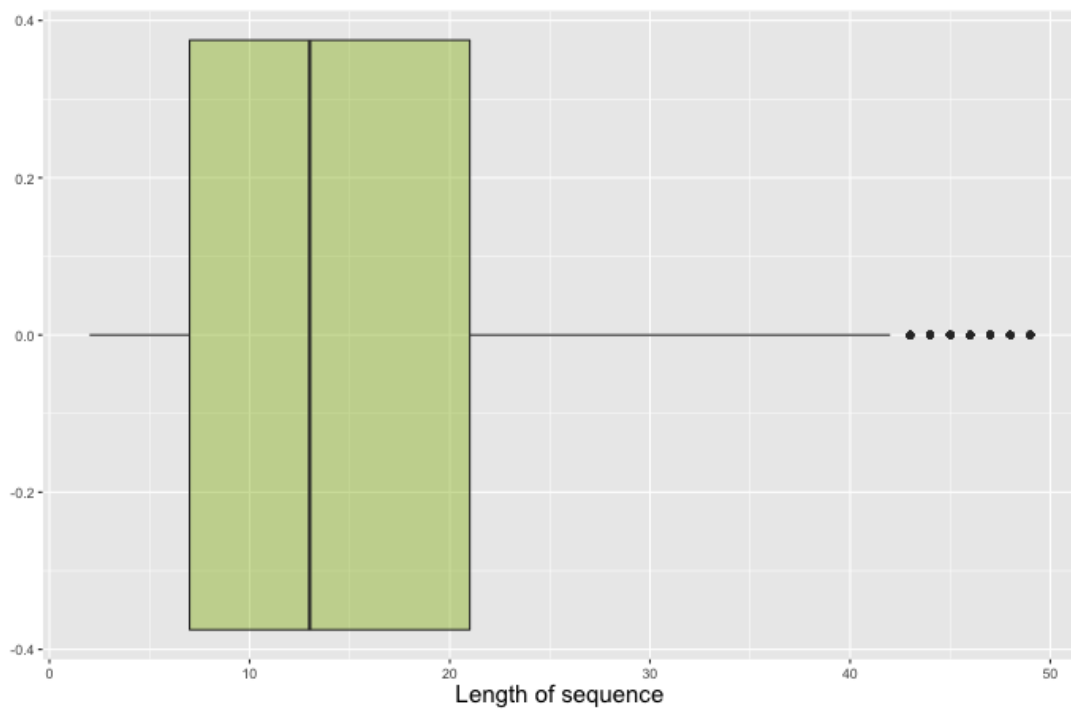


Figure 4.5: Boxplot of the sequence lengths after outlier removal.

4.2.1 Winning and Losing Teams

With the sequences representing possession by particular teams, one can compare sequences from winning and losing teams. For the second objective of the study, the data set was split into winning team and losing team sequences. A winning team was defined as the team whose score was higher than that of the opposition teams score at the end of a game. Previously, a sequence contained actions from both the winning and losing sides. For example, a sequence would contain an attacking action such as a ‘carry’ for one team, as well as a defending action such as a ‘tackle’ for the other team. Therefore, the data needed to be split into sequences that did not contain actions from both teams.

The focus for the secondary objective was attacking sequences of play by each team. Figure 4.6 demonstrates how these sequences were formed. Each team was assigned a ‘Team ID’ variable that indicated which team was conducting a particular action. To establish which teams were winning and losing sides, the scores at the end of each game were assessed, and the team with the highest score was labelled a winning team. Next, each sequence was allocated to the team that had possession during that sequence. From Figure 4.6, Team A had possession of the ball and was therefore attacking during that sequence. Finally, each action that did not belong to the team in possession (defending actions) was removed from each sequence. This resulted in sequences that only contained attacking actions for the winning and losing teams.

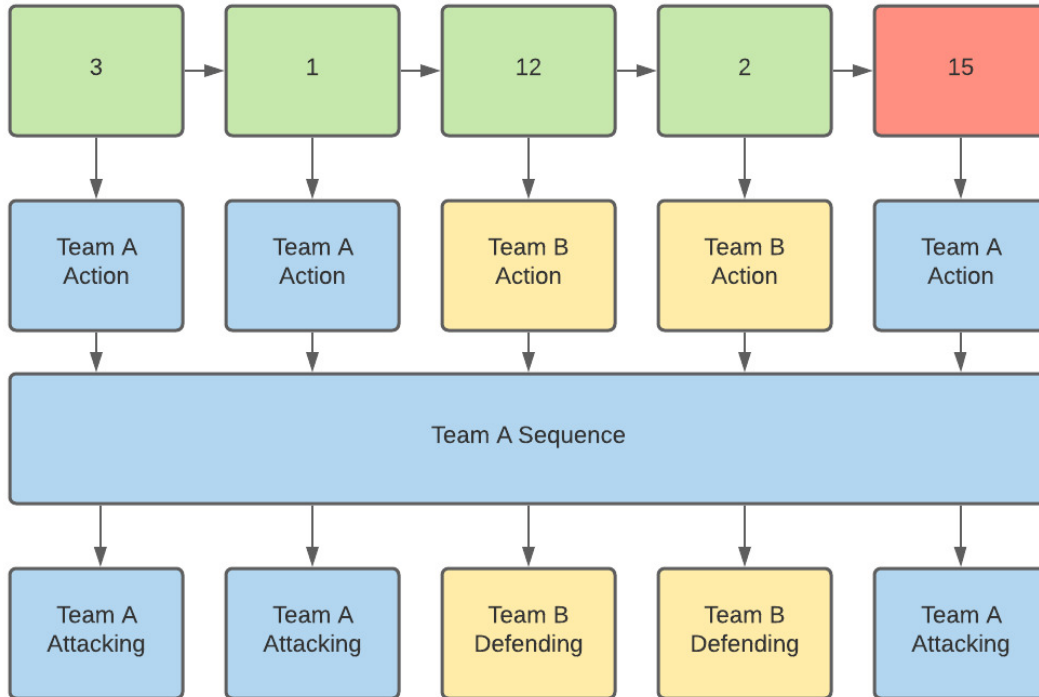


Figure 4.6: Flow diagram illustrating how attacking sequences are classified.

The distribution of the actions in the sequences for winning and losing teams are displayed in Figure 4.7. It was noted that these sequences did not contain any defending actions, which are actions ‘2’, ‘12’ and ‘21’, such that these sequences consisted only of attacking actions for each team.

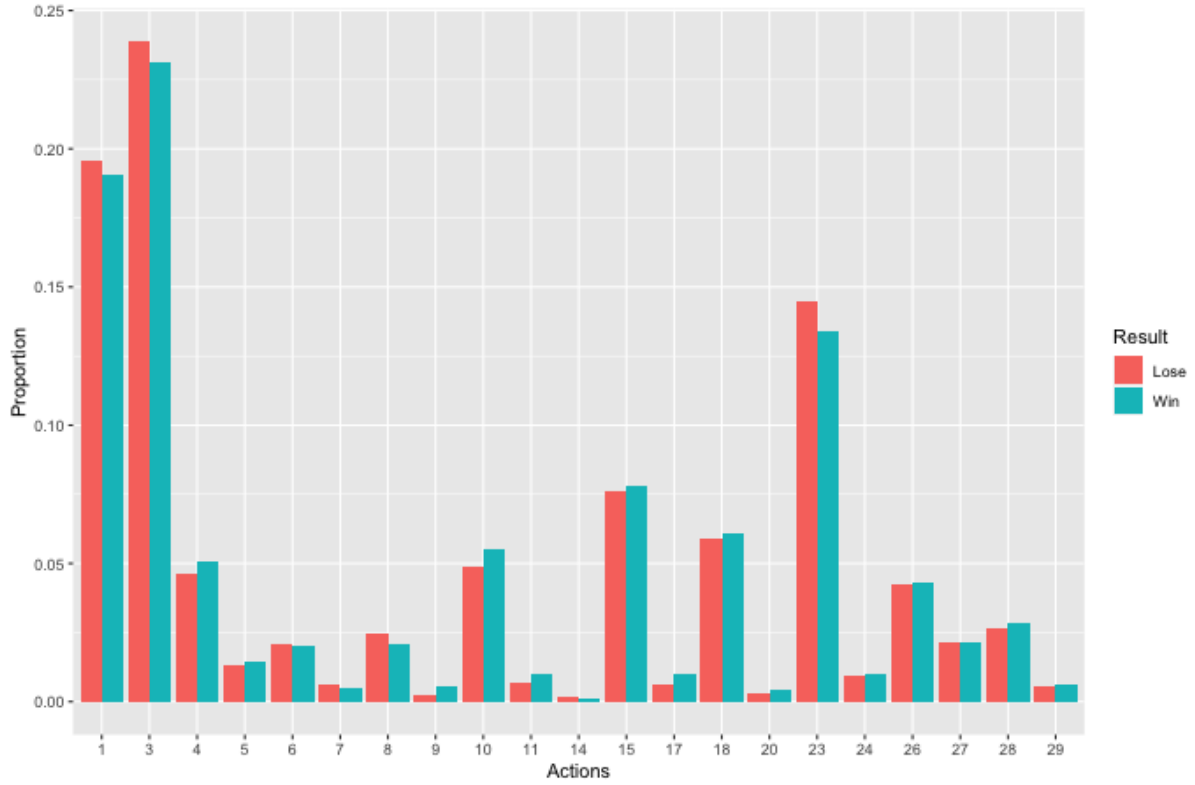


Figure 4.7: Distribution of actions between winning and losing teams

Chapter 5

Model Fitting

This chapter explores two base LSTM models of different depths: A single-layer LSTM and a 2-layer stacked LSTM. Justification for the dense layer, activation function, loss function, optimizer, and the absence of validation are discussed. This chapter will also discuss the measures used to determine which model performs best and the results from various hyperparameter experiments.

5.1 Model Specifications

5.1.1 Validation

Validation allows one to assess the accuracy of a model by assessing the predictions on an unseen subset of the data. The primary objective of this study was a generation task and not a prediction task. Therefore, validation was not needed as this study did not need to compare the prediction accuracy. This allowed for the full data set to be used when training the models.

5.1.2 Dense Layer (Fully-connected layer)

To classify the actions generated by the activation function (Section 5.1.3), the output layer was required to contain the same number of nodes as there were classes. Therefore a dense layer was used in each model to reduce the dimensions of the LSTM layer. The dense layer connects each node from the previous layer (the previous LSTM layer) to a dimension suitable for the activation function. As there were 24 actions and a padding variable '0', the model used 25 nodes.

5.1.3 Activation Function

Activation functions are used in neural networks to transform input into output, which in turn is fed as input to the next layer of the network. In an artificial neural network, an activation function is applied to the sum of the products of inputs and their corresponding weights to get the output of that particular layer, which will be supplied as input for

the next layer. The prediction accuracy of a neural network is defined by the type of activation function applied [27]. A multi-class problem of $C = 25$ different classes requires the Softmax activation function. This function calculates the relative probabilities for each of the classes (C). The formula for the Softmax function is defined as:

$$f(z)_i = \frac{\exp(z_i)}{\sum_j^C \exp(z_j)} \quad (5.1)$$

Where z represents the values from the neurons of the output layer and the exponent introduces non-linearity. To normalize the $\exp(z_i)$, it is divided by the sum of the exponent values and converted into probabilities for each class.

5.1.4 Loss Function

A categorical cross-entropy is a loss function that is used in multi-class classification tasks. This sequence generation problem is a classification task. An output action can only belong to one of many possible categories or actions, and the model must decide which category it belongs to. Hence, this loss function was used for the LSTM models. The categorical cross-entropy function is defined as:

$$CE = - \sum_i^C t_i \log(f(z)_i) \quad (5.2)$$

Where $f(z)_i$ is the scalar vector in the model output, t_i is the target vector, and C is the number of classes, $C = 25$. This loss function is designed to calculate the difference between the target distribution and the observed distribution. The optimizer minimizes the loss function.

5.1.5 Optimizer

All the models used the Adam optimizer. This is an adaptive optimizer which means it does not require a pre-defined learning rate. It instead updates the learning rate throughout the algorithm. The Adam optimizer works by calculating an exponential moving average of the gradient and the squared gradient. It is well suited for problems

that have large data and it compares favourably to other stochastic methods [28].

5.1.6 Epochs, Stopping Criterion and Patience

When training neural networks, one attempts to minimize loss as much as possible. There comes a point where the marginal gain for the network running another epoch (an iteration of training where all the training data is run through the model) is so small it is negligible. Therefore, an early stopping criterion was set to terminate the neural network once the difference in loss from the previous iteration (epoch) was below a threshold, δ . The early stopping criterion was set to a suitably small value, $\delta = 0.001$, such that the algorithm did not terminate early and improvements were negligible. Additionally, a patience of 5 was added to prohibit the algorithm from terminating due to variances in the loss. The models were set to run for a maximum of 200 epochs. The early stopping criterion was implemented before then for all of the models used in this study, meaning the ‘plateau’ shape of the loss curve had been reached and loss had been minimized.

Figure 5.1 illustrates the implementation of the early stopping criterion. The declining gradient indicates that the model is learning. One can note the loss declining rapidly at the the start after which the slope slowly levels out. Once the learning was smaller than the threshold limit of $\delta < 0.001$, the training was terminated and the rest of the epochs need not to have been run.

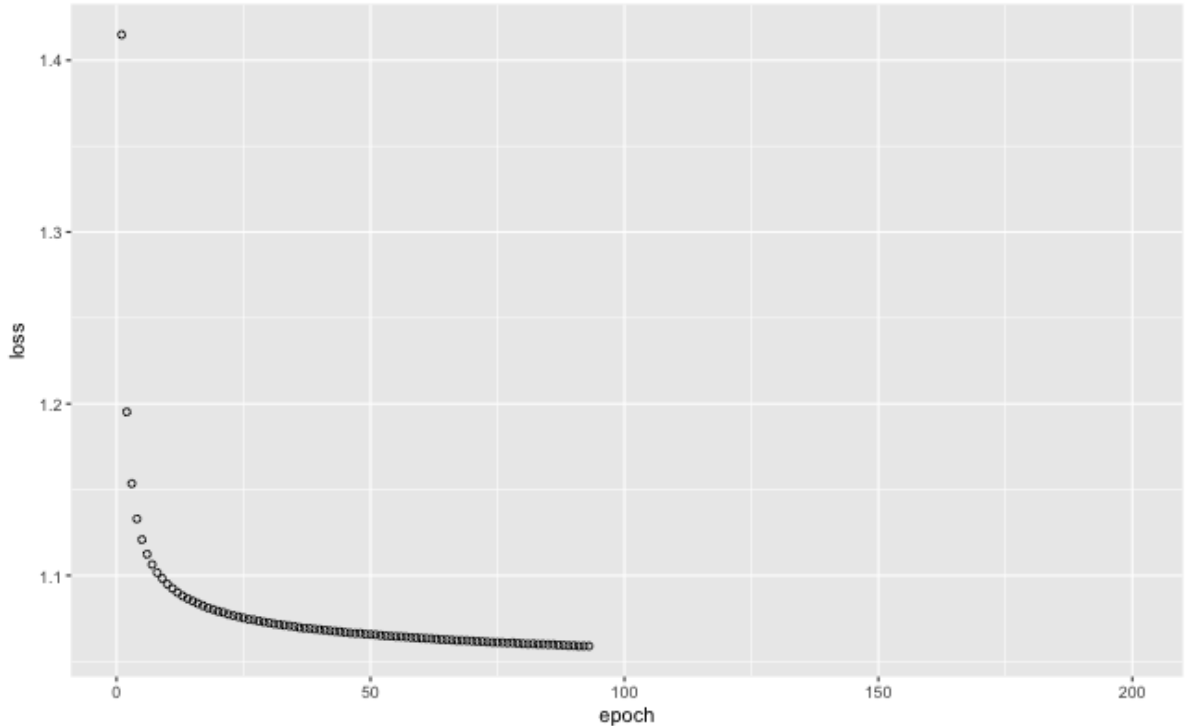


Figure 5.1: Implementation of early stopping criterion.

5.1.7 Temperature

One of the hyperparameters used for the LSTM models was temperature, sometimes referred to as diversity. Temperature is used to control the randomness of predicted sequences. The temperature value ranges between 0 and 1. A higher temperature (closer to 1) will generate more diverse sequences at a potential cost of incorrect predictions. A lower temperature will produce more conservative sequences but will have fewer incorrect predictions. Three values for temperature were used in the hyperparameter experiments: 0.2, 0.6 and 1.

5.1.8 Batch Size and Units

Units are the number of nodes in each hidden layer. This is linked to the dimensions and computations the model processes in each layer. An increase in the number of units creates more complex models but does not necessarily minimize loss. The goal was to find a unit size that minimized loss. Two unit sizes were used for the models: 32 and 128.

Batch size refers to the number of samples processed before the model is updated. Larger batch sizes are often desired to train LSTM models, as this allows for faster computational speeds. However, a batch size that is too large will lead to degradation in the quality of the model, as measured by its ability to generalize. The batch sizes used in the models were 32, 128 and 256.

5.1.9 One-hot Encoding

The most common approach to converting categorical features to a suitable format for input to a machine learning model is one-hot encoding. It is one method of converting data to prepare it for an algorithm to get better predictions. Each categorical value, which are the actions, in this case, is converted into a new categorical column and assigned a binary value of 1 or 0 to those columns, such that each category is represented as a binary vector. In the binary vectors, all the values are zero besides the index, which is marked with a 1. Tables 5.1 and 5.2 demonstrates how one-hot coding works on a sequence of actions. One-hot coding takes in a sequence such as:

Original sequence:	‘3’	‘1’	‘2’
Translation:	‘Pass’	‘Carry’	‘Tackle’

Table 5.1: Original sequence.

and converts each action into a binary vector:

'3'	0	0	1
'1'	1	0	0
'2'	0	1	0

Table 5.2: One-hot encoded sequence.

Assuming a natural ordering between categories may result in poor performance of the model or unexpected predictions. One-hot encoding ensures that the algorithm does not assume that higher numbers are more important. In turn, this improves the predictions and classification accuracy of the model.

5.2 Measures of Performance

Usually, validation accuracy and loss would provide a measure of neural network model performance. Section 5.1.1 described why validation was not used. Therefore the only numerical measure used to assess the performance of the models was the training loss scores.

A large part of the analysis relies on domain knowledge of rugby. The problem is similar to that of text generation. In sequential text generation models, a large part of the analysis is simply reading the output and seeing whether the text generated makes logical sense. The researcher would need to use their domain knowledge of language to assess the accuracy of their results. Similarly, domain knowledge of rugby is used to assess whether the sequences are logical and realistic.

Additionally, histograms can be used to compare the frequency of actions generated in the models. The sequential nature is explored by creating histograms of which actions precede another action. This measure ensures that the actions generated do not only display realistic distribution, but also a realistic sequential nature.

5.3 Model Design

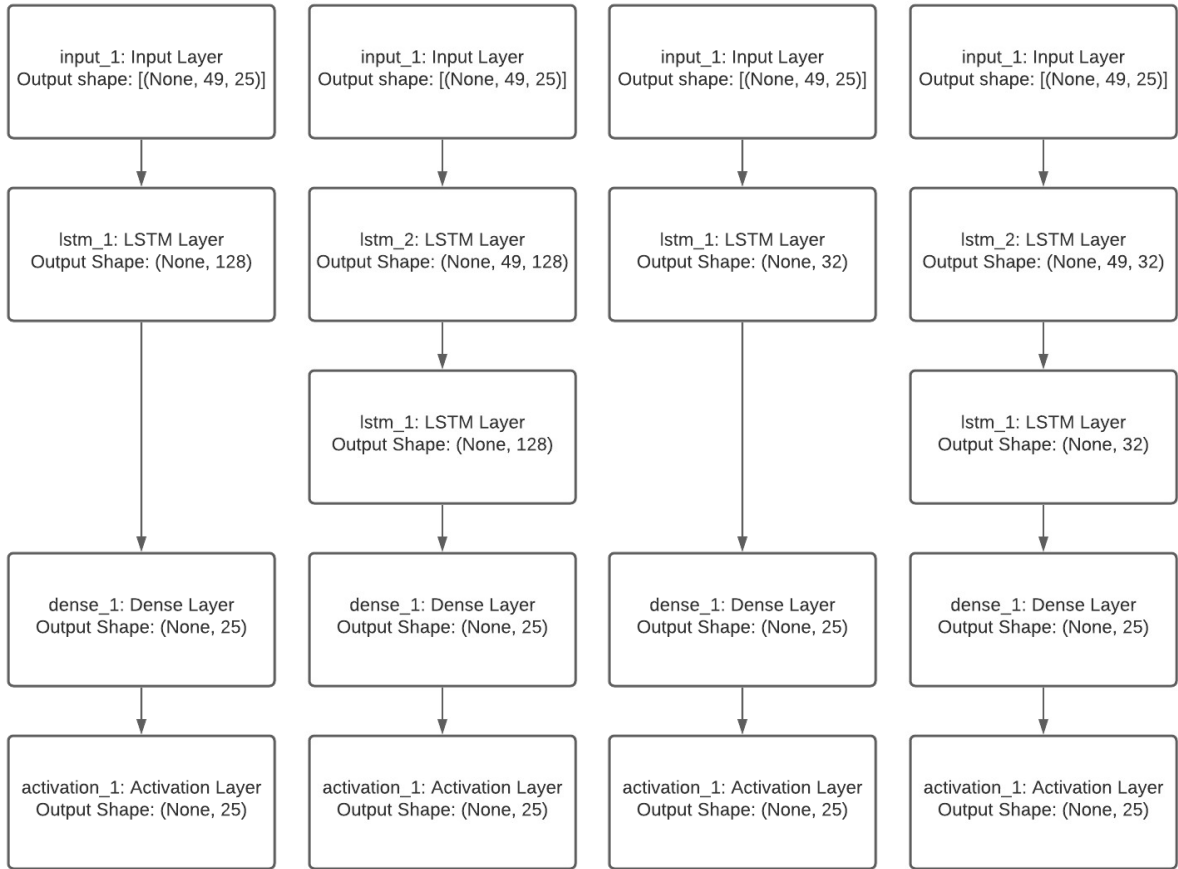
The LSTM models were designed using a one-hot encoding to represent the categorical variables in the sequences. The models were then built using the Keras functional Application Programming Interface (API). A summary of the hyperparameters used in the models is shown in Table 5.3.

Summary of model specifications

Activation function	Softmax
Loss function	Categorical cross-entropy
Optimizer	Adam
Batch sizes	32, 128, 256
LSTM units	32, 128
Temperature	0.2, 0.6, 1

Table 5.3: Model specifications

Figure 5.2 below displays the model architecture. The first column represents the model architecture for a single-layer LSTM with 128 units, the second column a stacked LSTM with 128 units, the third column a single-layer LSTM with 32 units and the fourth column a stacked LSTM with 32 units.



(a) Single-layer, 128 Units. (b) 2-layer Stacked, 128 Units. (c) Single-layer, 32 Units. (d) 2-layer Stacked, 32 Units.

Figure 5.2: Model architecture.

One can note the additional depth in the 2-layer stacked LSTM. The depth creates a more complex model and additional computational time is expected. It may lead to a richer representation of the data and hence better sequence generation. The effectiveness of the additional layer will be discussed in Chapter 6.

5.4 Hyperparameter Optimization

The hyperparameter results from the single-layer LSTM and the 2-layer LSTM experiments are displayed in Section 5.4.1 and Section 5.4.2. Included in the results is an indication of whether the sequences generated were realistic or unrealistic, based on the authors' collective domain knowledge of rugby union.

5.4.1 Single-layer LSTM

Model ID	LSTM Layers	Batch size	Units	Temp.	Loss	Realistic/Unrealistic
1.1	1	256	128	0.2	0.9463	Unrealistic
1.2	1	256	128	0.6	0.9463	Unrealistic
1.3	1	256	128	1	0.9463	Realistic
1.4	1	256	32	0.2	1.052	Unrealistic
1.5	1	256	32	0.6	1.052	Unrealistic
1.6	1	256	32	1	1.052	Realistic
1.7	1	128	128	0.2	0.9561	Unrealistic
1.8	1	128	128	0.6	0.9561	Realistic
1.9	1	128	128	1	0.9561	Realistic
1.10	1	128	32	0.2	1.0544	Unrealistic
1.11	1	128	32	0.6	1.0544	Realistic
1.12	1	128	32	1	1.0544	Realistic
1.13	1	32	128	0.2	0.9704	Unrealistic
1.14	1	32	128	0.6	0.9704	Realistic
1.15	1	32	128	1	0.9704	Realistic
1.16	1	32	32	0.2	1.0552	Unrealistic
1.17	1	32	32	0.6	1.0552	Unrealistic
1.18	1	32	32	1	1.0552	Realistic

Table 5.4: Single-layer LSTM hyperparameter results.

All sequences generated from single-layer models made logical sense in rugby union. However, when lower temperatures were used, the models generated sequences of unrealistic length. Hence they were classified as unrealistic. The 128 unit models achieved a lower loss compared to the 32 unit models. Models with larger batch sizes achieved a lower loss.

5.4.2 Two-layer Stacked LSTM

Model ID	LSTM Layers	Batch size	Units	Temp.	Loss	Realistic/Unrealistic
2.1	2	256	128	0.2	0.8904	Unrealistic
2.2	2	256	128	0.6	0.8904	Realistic
2.3	2	256	128	1	0.8904	Realistic
2.4	2	256	32	0.2	1.041	Unrealistic
2.5	2	256	32	0.6	1.041	Unrealistic
2.6	2	256	32	1	1.041	Realistic
2.7	2	128	128	0.2	0.9166	Unrealistic
2.8	2	128	128	0.6	0.9166	Realistic
2.9	2	128	128	1	0.9166	Realistic
2.10	2	128	32	0.2	1.0395	Unrealistic
2.11	2	128	32	0.6	1.0395	Unrealistic
2.12	2	128	32	1	1.0395	Realistic
2.13	2	32	128	0.2	0.8719	Realistic
2.14	2	32	128	0.6	0.8719	Realistic
2.15	2	32	128	1	0.8719	Realistic
2.16	2	32	32	0.2	1.041	Unrealistic
2.17	2	32	32	0.6	1.041	Unrealistic
2.18	2	32	32	1	1.041	Unrealistic

Table 5.5: Two-layer Stacked LSTM hyperparameter results.

A similar trend was noted in both models, whereby a larger batch size resulted in better results. Similarly, the 128 unit models outperformed the 32 unit models. The two-layer stacked LSTM models achieved a lower loss than the corresponding single-layer models with the same dimensions. However, this did not result in better sequences generated. The models with batch size 32 and 32 units achieved a lower loss in the two-layer stacked LSTM, but the ‘1’ action was predominately generated, which is illogical and therefore an unrealistic sequence. Again, domain knowledge of rugby is needed to interpret the output. This is discussed more in Chapter 6.

Chapter 6

Results

6.1 Model Evaluation

The criteria used in evaluating the performance of the LSTM models were:

1. Loss value: Which models had the lowest loss values?
2. Domain knowledge of rugby union: Did the generated sequences make logical sense based on the temperature value of the LSTM?
3. Distribution of actions: How did the distribution of actions from the generated sequence compare to the distribution of actions from the original data set?
4. Distributions of ‘next actions’: How did the distribution of the actions following after a certain action compare to the same distribution from the original data set?
5. Common Sequences: Were the common sequences generated using the models similar to that of the original data set?

6.1.1 Loss Value

The models with the four lowest loss values from Section 5.4 are displayed in Table 6.1. Based on these loss values, these models were expected to be the best performing models. Although the loss values of all the models were relatively similar, by assessing a few sequences from these four models, it was clear that they were generating sequences that mimic realistic sequences in a game of rugby. At this point of the study, the remaining LSTM models will not be used for further analysis.

Model ID	Model Type	Batch Size	Units	Loss
1.3	Single-layer	256	128	0.9463
1.9	Single-layer	128	128	0.9561
2.9	Two-layer stacked	128	128	0.9166
2.15	Two-layer stacked	32	128	0.8719

Table 6.1: Models with lowest loss values.

6.1.2 Temperature

For each of these four chosen models, sequences were generated using temperatures of 0.2, 0.6 and 1. By looking at sequences generated by the four models using the different temperature values, one can assess which temperature results in more realistic sequences. The assessment of these sequences was done primarily by using domain knowledge of rugby union. It was evident that models using the lower temperature values of 0.2 and 0.4 were generating sequences that were both too long and unrealistic. A temperature value of 1 resulted in shorter and more realistic sequences being generated. The lower temperature allowed the model to be more confident in generating subsequent actions; however, this conservative nature of a low temperature meant it took longer for the model to create sequence ending actions. Hence, a temperature value of 1 was used for further generation of sequences.

6.1.3 Distribution of Actions

For each of the four chosen LSTM models, 10000 sequences were generated. These generated sequences were compared to the original data set to assess the model's performance in generating accurate proportions of actions. The distribution of actions from the generated sequences was obtained for each model and compared to the distribution of actions from the original data set. These distributions can be visualized using histogram plots.

Model 1.3: Single-layer with 256 Batch Size and 128 Units

Figure 6.1 displays the distributions of actions from the original data set compared to the distribution of actions from 10000 generated sequences using a single-layer model with a batch size of 256 and 128 units. The x-axis values of this plot indicate the action number, and the y-axis values indicate the proportion of actions generated. From this plot, the distribution of actions from the generated sequences shows a similar shape to the distribution of actions from the original data set. For the most prominent actions in the original data set (actions '1', '2', '3' and '23'), this model generated a lower proportion of these actions. For the remaining actions, the model generated slightly higher proportions. In the original data set, there are equal proportions of actions '28' and '29', which was emulated by the generated sequences using Model 1.3.

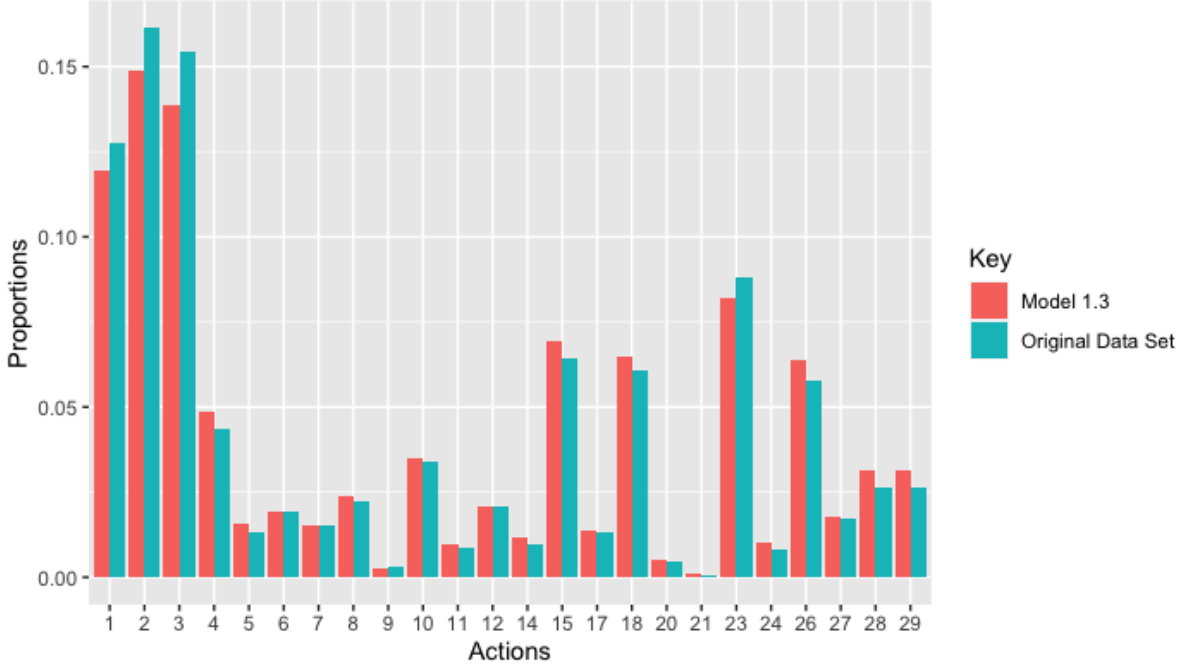


Figure 6.1: Histogram comparing the distribution of actions from Model 1.3 to the original data set.

Model 1.9: Single-layer with 128 Batch Size and 128 Units

Figure 6.2 displays the distributions of actions from the original data set compared to the distribution of actions from 10000 generated sequences using a single-layer model with a batch size of 128 and 128 units. Once again, the model generated less of the most prominent actions ‘1’, ‘2’, ‘3’ and ‘23’, and slightly more of the less occurring actions. The distribution of actions obtained from the generated sequences using Model 1.9 closely follows the distribution shape of actions from the original data set. Model 1.9 generated equal proportions of actions ‘28’ and ‘29’, as seen in the original data set.

Model 2.9: Two-layer Stacked with 128 Batch Size and 128 Units

Figure 6.3 displays the distributions of actions from the original data set compared to the distribution of actions from 10000 generated sequences using a two-layer stacked LSTM model with a batch size of 128 and 128 units. In the original data set, there was a higher proportion of action ‘2’ than action ‘3’; however, it is observed that Model 2.9 generated a higher proportion of action ‘3’ than action ‘2’. This model generated a higher proportion of actions ‘1’ and ‘3’ than the original data set, as opposed to the previous models.

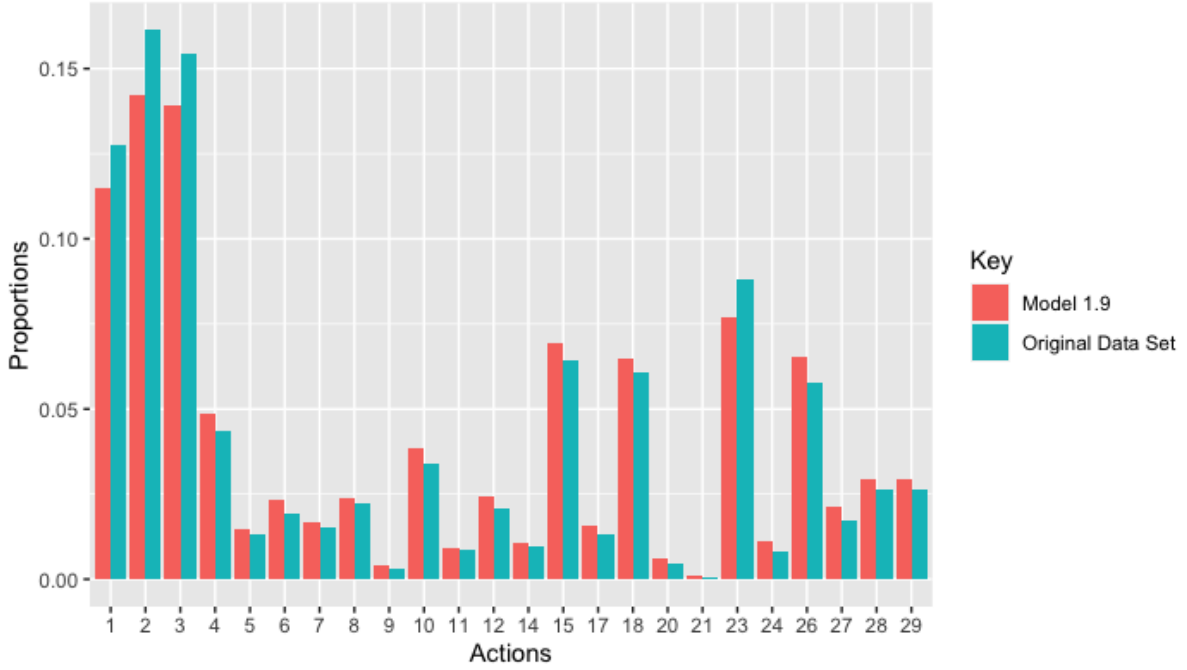


Figure 6.2: Histogram comparing the distribution of actions from Model 1.9 to the original data set.

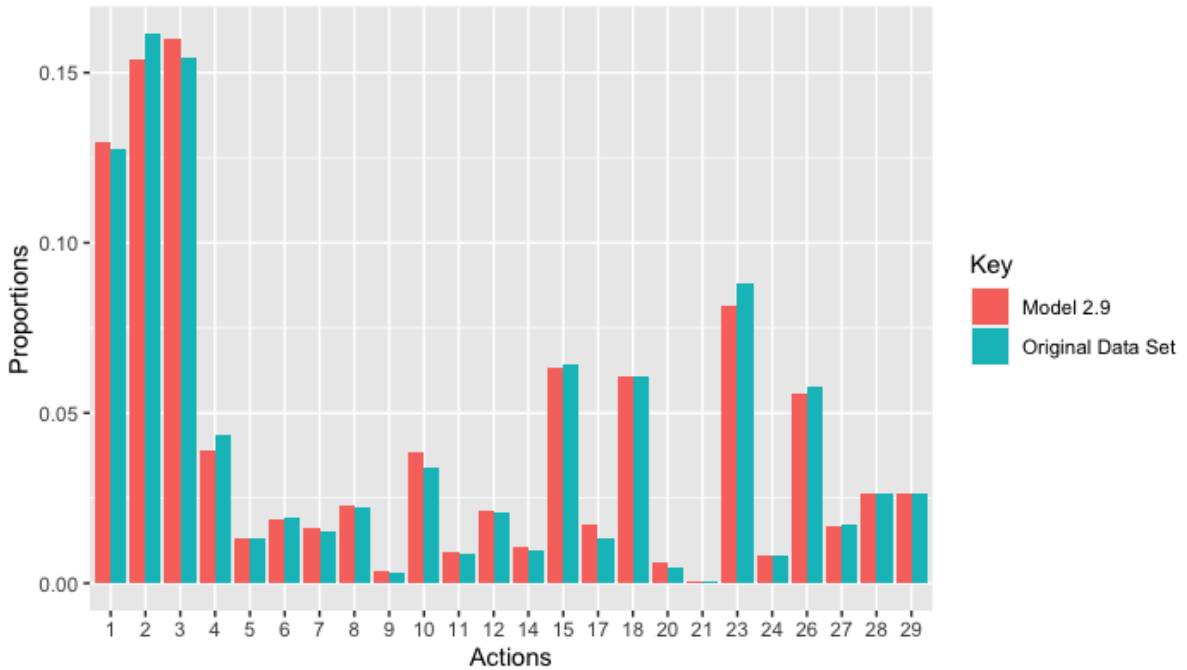


Figure 6.3: Histogram comparing the distribution of actions from Model 2.9 to the original data set.

Model 2.15: Two-layer Stacked with 32 Batch Size and 128 Units

Figure 6.4 displays the distributions of actions from the original data set compared to the distribution of actions from 10000 generated sequences using a two-layer stacked LSTM model with a batch size of 32 and 128 units. Model 2.15 generated slightly higher proportions of actions ‘1’, ‘2’, ‘3’, ‘18’ and ‘23’. It is evident from this plot that

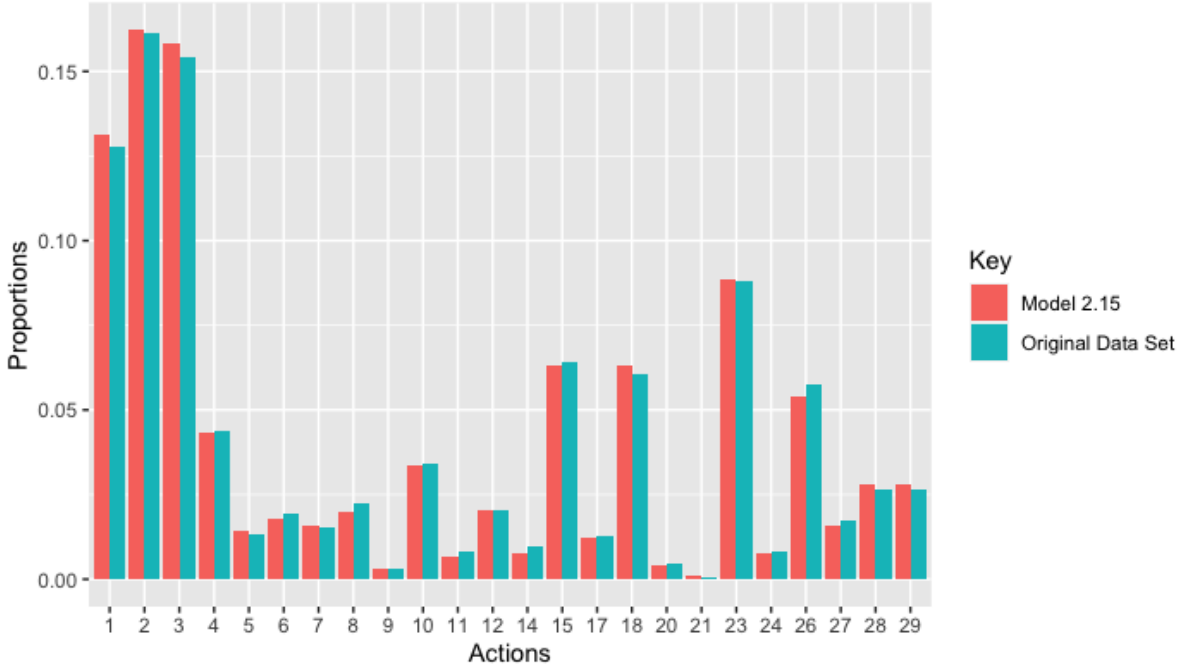


Figure 6.4: Histogram comparing the distribution of actions from Model 2.15 to the original data set.

the distribution of actions from this model was the closest to that of the distribution obtained from the original data set.

Comparison of Model Distributions

Figure 6.5 displays the distributions of actions from all four of the chosen models. From this plot, there were no significant differences observed between the models. The distributions for each model were of a relatively similar shape. For the highest occurring actions (actions ‘1’, ‘2’, ‘3’ and ‘23’), Model 2.15 generated a higher proportion of these actions, except for action ‘3’, which Model 2.9 generated more of.

6.1.4 Distribution of Next Actions

Although the most effective method of assessing the accuracy of the generated sequences is by using domain knowledge of rugby union to interpret how realistic the generated sequences are, one can visually assess the accuracy of the generated sequences by looking at specific actions and noticing what other actions should follow them (referred to as ‘next actions’), and whether this was emulated by the 10000 generated sequences. As there are 24 actions in the data set, only a subset will be used for further evaluation. The actions assessed are:

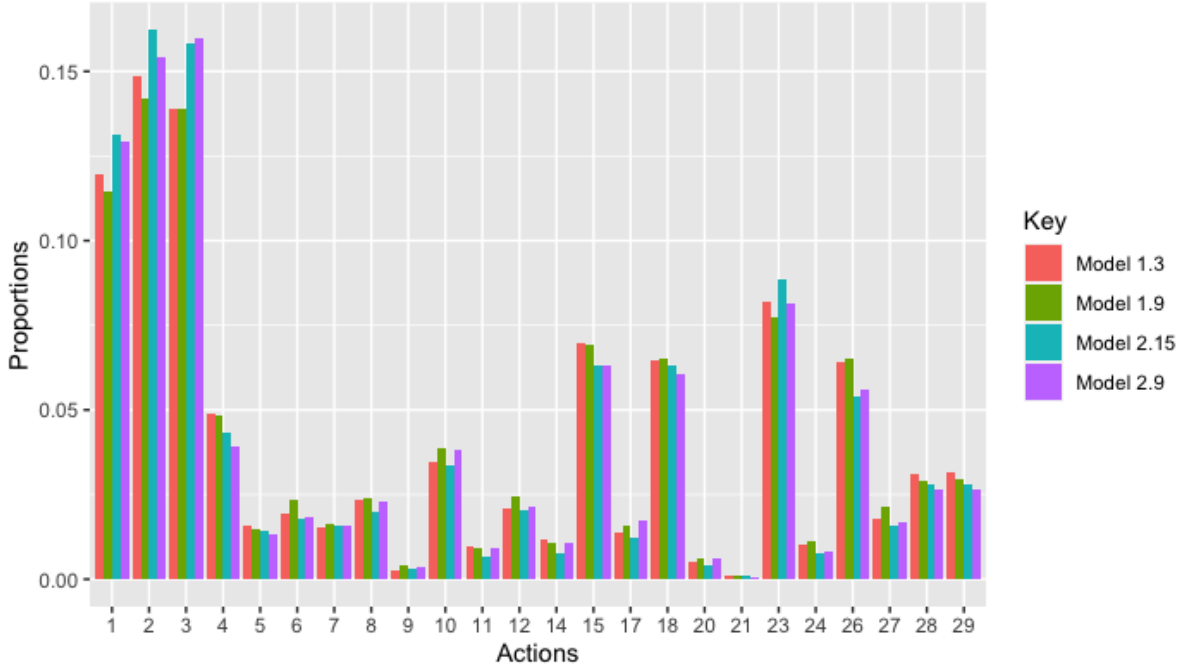


Figure 6.5: Histogram comparing the distribution of actions from four chosen models.

1. Attacking Qualities: Action '10'.
2. Possession: Action '15'.
3. Penalty Conceded: Action '7'.

Attacking Qualities Next Actions

'Attacking qualities' is an action that indicates some form of positive forward movement shown by the team with possession, such as a player breaking through the line of defence. Figure 6.6 displays the distribution of actions that follow after the attacking qualities action in the original data set. Attacking qualities are usually followed by a carry (action '1') or a tackle (action '2'). There is also the possibility of an attacking quality being followed by another attacking quality (action '10'). Attacking quality actions can lead to the scoring of a try (action '9').

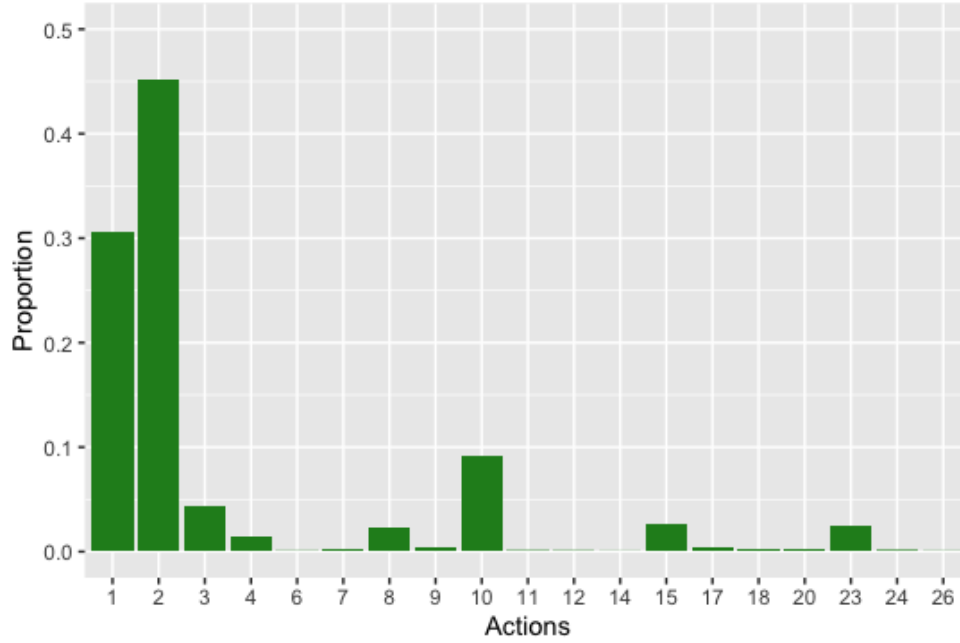


Figure 6.6: Distribution of next actions after attacking quality action from the original data set.

Figure 6.7 displays the distribution of next actions for the attacking quality action based on the generated sequences from the four chosen LSTM models. These distributions are closely related to the distribution in Figure 6.6. The only differences occurred when a model failed to generate certain actions or generated actions that did not follow an attacking sequence in the original data set. Table 6.2 shows which actions were incorrectly generated and which actions failed to be generated as next actions after an attacking quality by the four models. The inability of a model to generate or fail to generate certain actions was related to those actions that were of a very small proportion in the original data set.

Model ID	Actions Failed to Generate	Total	Actions Incorrectly Generated	Total
1.3	6, 14	2	5, 21, 28	3
1.9	14	1	29	1
2.9	26	1		0
2.15	14, 26	2	27	1

Table 6.2: Attacking quality next actions failed to be generated, or incorrectly generated, by the four chosen LSTM models.

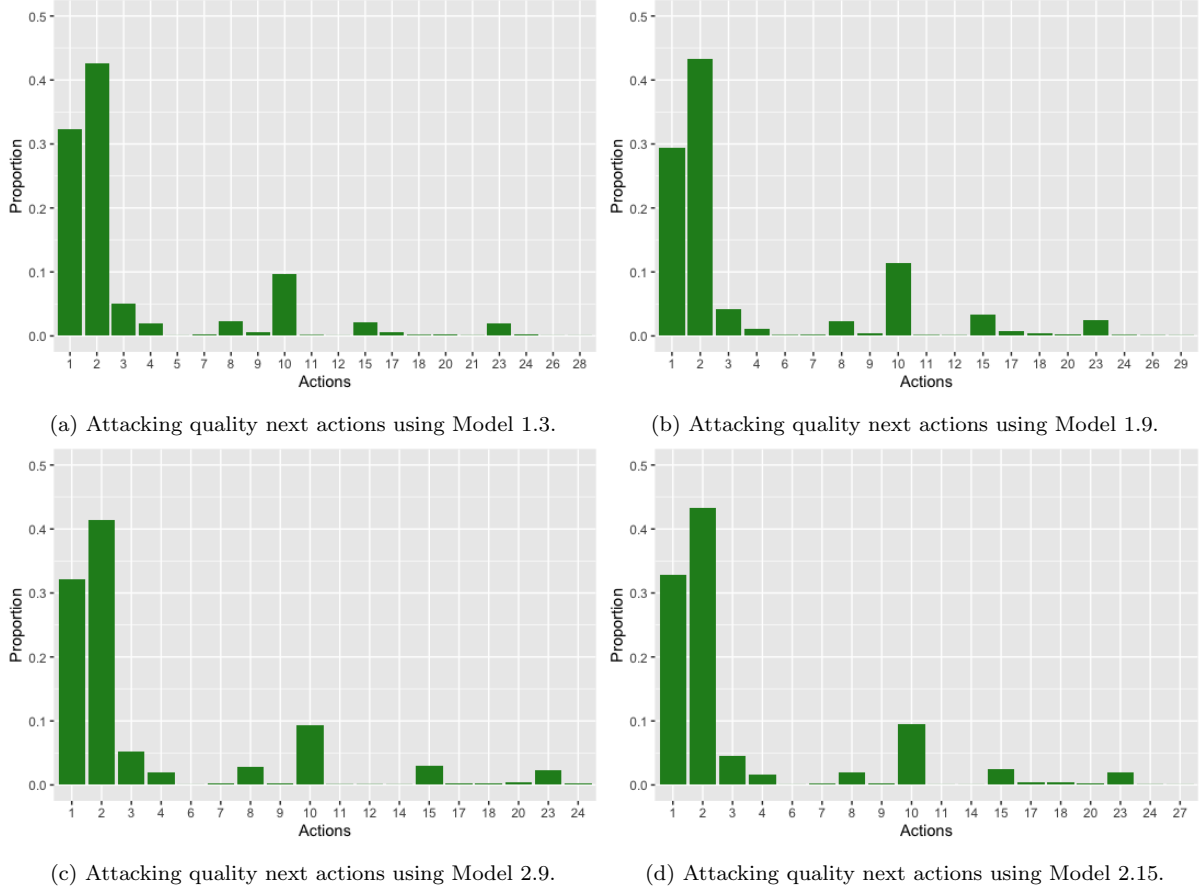


Figure 6.7: Distribution of next actions after attacking quality action for the four chosen LSTM models.

Possession Next Actions

The ‘possession’ action indicates a change in possession from one team to another and is the ending action for every sequence. Therefore, the actions that follow after a change in possession are those actions that begin every generated sequence. Figure 6.8 displays the distribution of next actions for the possession action in the original data set. From this plot, the actions that most often begin a sequence are actions ‘18’ and ‘26’. Action ‘18’ (a ‘collection’) refers to a player collecting a ball either off the ground or from a kick. Action ‘26’ is labelled as a ‘sequence’ action. This action describes the start of a new sequence of play. It continues to increase with every change in possession, as well as every restart of play. This action does not refer to the sequences used in this study.

Figure 6.9 displays the distribution of next actions for the change in possession action based on the generated sequences from the four chosen LSTM models. The four models form an almost identical distribution to that of the original data in Figure 6.8. When it comes to actions with very small proportions, there were some differences. Model 2.9 generated a higher proportion of actions ‘1’ and ‘3’, which was closer to the distribution

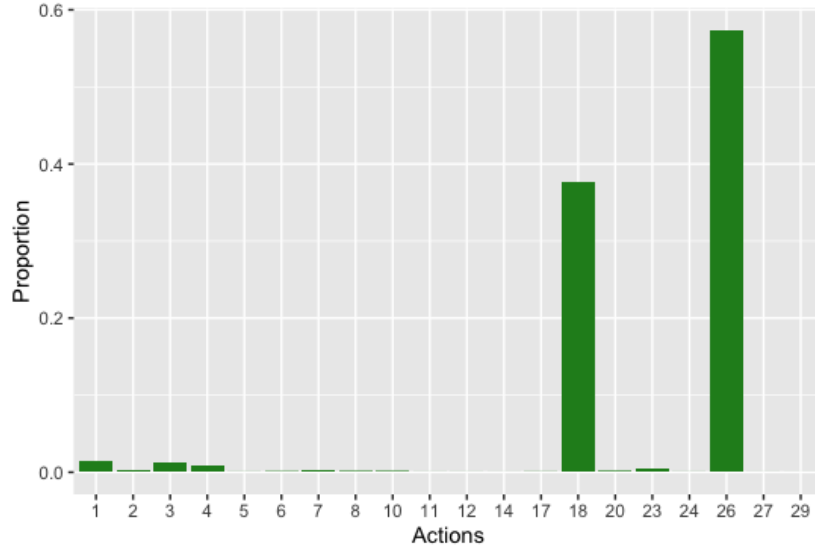


Figure 6.8: Distribution of next actions after possession action from the original data set.

of the original data set. Model 2.15 generated a lower proportion of action ‘26’ compared to the other models and original data set. Table 6.3 shows which actions were incorrectly generated and which actions the four models failed to generate as next actions after a change in possession.

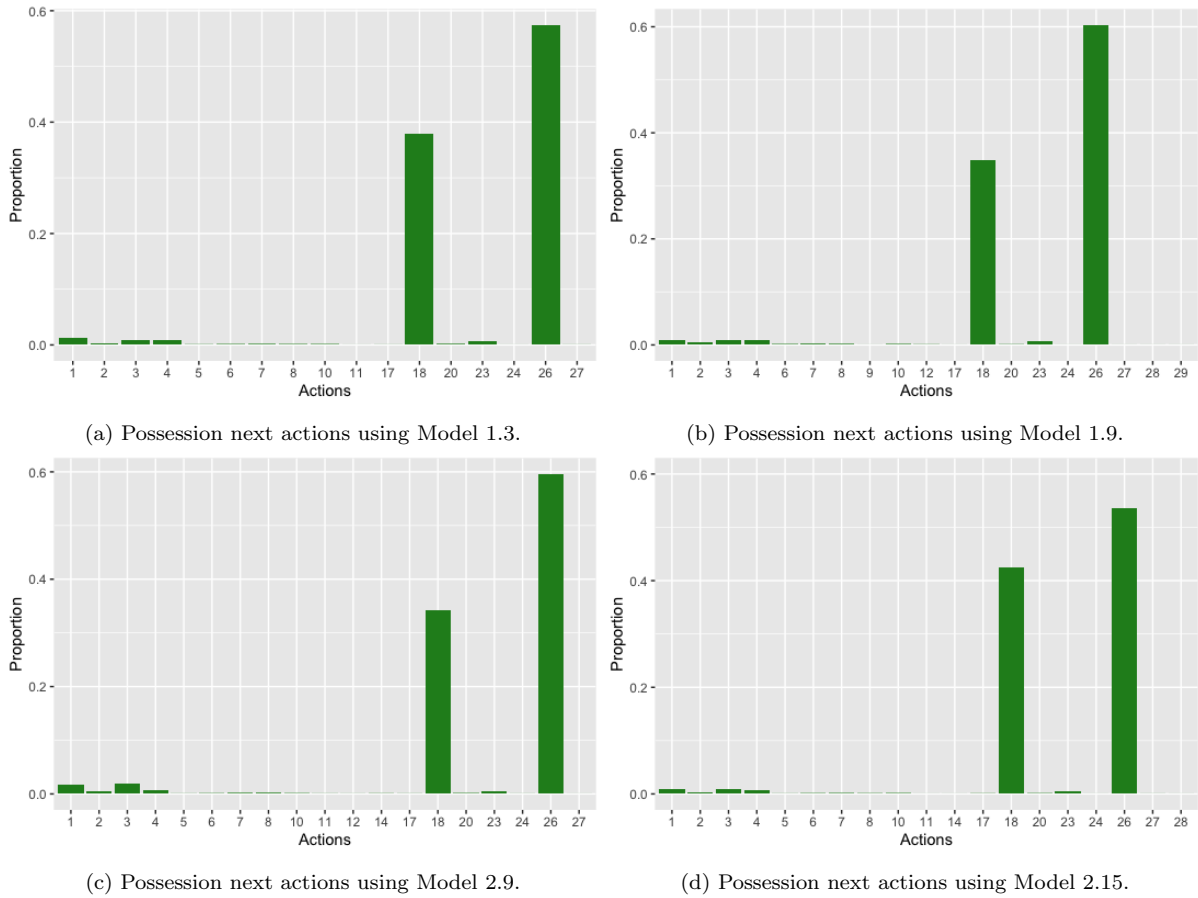


Figure 6.9: Distribution of next actions after possession action for the four chosen LSTM models.

Model ID	Actions Failed to Generate	Total	Actions Incorrectly Generated	Total
1.3	12, 14, 29	3	28	0
1.9	5, 11, 14	3		1
2.9	29	1		0
2.15	12	1		0

Table 6.3: Possession next actions failed to be generated, or incorrectly generated, by the four chosen LSTM models.

Penalty Conceded Next Actions

A penalty conceded is an action where a team performs an illegal act, leading to a penalty being awarded to the opposition team. This team then has the opportunity to choose from several options, such as taking a penalty kick for points or having an attacking scrum. Figure 6.10 displays the distribution of actions that follow after a penalty conceded from the original data set. The action that occurs the most after a penalty is conceded is action ‘15’, a change in possession. The next most prominent next action is action ‘4’, a kick, which is most likely referring to one of the penalty options where a team can kick the ball up field into touch (outside the field boundary) and have an attacking lineout.

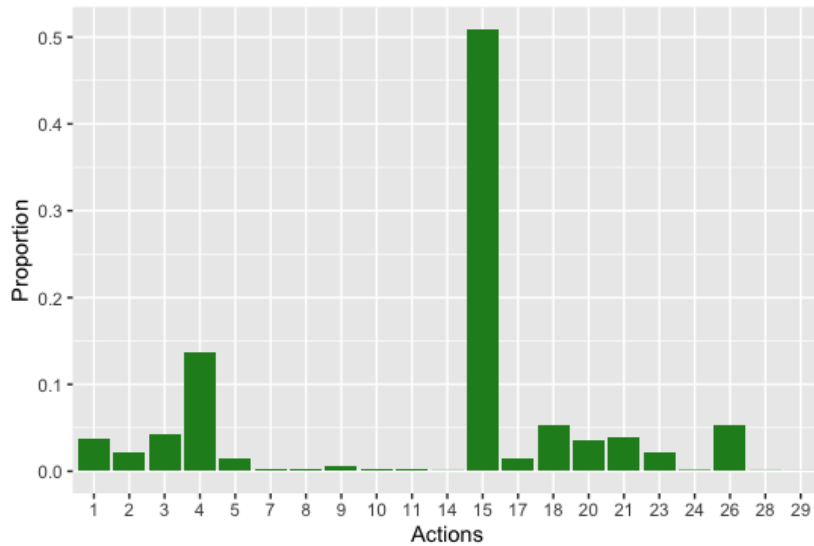


Figure 6.10: Distribution of next actions after penalty conceded action from the original data set.

Figure 6.11 displays the distribution of next actions for the penalty conceded action based on the generated sequences from the four chosen LSTM models. Once again, the distribution from these four models mirrors that of the original data set, apart from some minor differences. One of the differences observed was with Model 1.9 in Figure 6.11b, which generated more of action ‘20’ than actions ‘18’ and ‘21’, which is not seen in the other models and original data set. Model 2.9 in Figure 6.11c also generated action ‘20’

more than action ‘18’. Table 6.4 shows which actions were incorrectly generated and which actions the four models failed to generate as next actions after a penalty conceded.

Model ID	Actions Failed to Generate	Total	Actions Incorrectly Generated	Total
1.3	29	1	6	1
1.9	14	1	12	1
2.9	14	1	6	1
2.15	7, 28	2	6	1

Table 6.4: Penalty conceded next actions failed to be generated, or incorrectly generated, by the four chosen LSTM models.

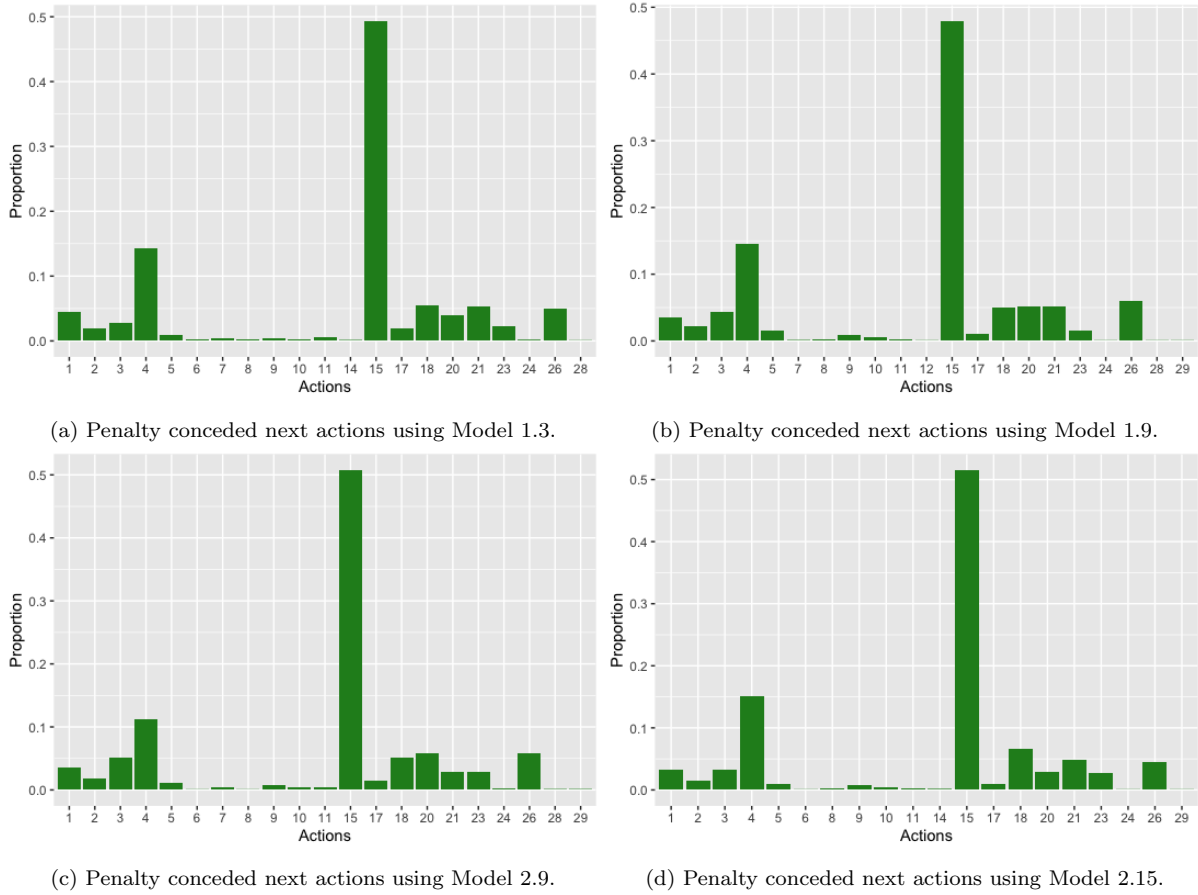


Figure 6.11: Distribution of next actions after penalty conceded action for the four chosen LSTM models.

6.1.5 Common Sequences

The five most frequently occurring sequences in the original data set are displayed in Table 6.5. This table shows the numerical sequence as well as its translation into actions.

It also indicates the proportion of that sequence out of all the sequences from the original data set. These sequences represent short sequences of play that a team performs when receiving a kick in their own half and want to move the ball back into the oppositions half through a kick.

Sequence	Translation	Proportion of Total Sequences
18, 1, 4, 15	Collection, Carry, Kick, Possession	0.036
18, 4, 15	Collection, Kick, Possession	0.026
18, 3, 4, 15	Collection, Pass, Kick, Possession	0.014
18, 3, 1, 4, 15	Collection, Pass, Carry, Kick, Possession	0.012
18, 1, 2, 23, 3, 4, 15	Collection, Carry, Tackle, Ruck, Pass, Kick, Possession	0.007

Table 6.5: Most frequently occurring sequences in original data set.

The five most frequently occurring sequences from the four LSTM models are shown in Tables 6.6 to 6.9. These results show that the models performed well in predicting those sequences that occur most in the original data set. Model 1.3 and Model. 2.15 have the same most frequently occurring sequences as the original data set, with relatively similar proportions.

Sequence	Translation	Proportion of Total Sequences
18, 1, 4, 15	Collection, Carry, Kick, Possession	0.035
18, 4, 15	Collection, Kick, Possession	0.033
18, 3, 4, 15	Collection, Pass, Kick, Possession	0.018
18, 3, 1, 4, 15	Collection, Pass, Carry, Kick, Possession	0.012
18, 1, 2, 23, 3, 4, 15	Collection, Carry, Tackle, Ruck, Pass, Kick, Possession	0.01

Table 6.6: Most frequently occurring generated sequences using Model 1.3.

Sequence	Translation	Proportion of Total Sequences
18, 1, 4, 15	Collection, Carry, Kick, Possession	0.035
18, 4, 15	Collection, Kick, Possession	0.033
18, 3, 1, 4, 15	Collection, Pass, Carry, Kick, Possession	0.01
26 6 27 24 4 15	Sequence, Lineout Throw, Lineout Take, Maul, Kick, Possession	0.01
18 1 2 23 3 4 15	Collection, Carry, Tackle, Ruck, Pass, Kick, Possession	0.01

Table 6.7: Most frequently occurring generated sequences using Model 1.9.

Sequence	Translation	Proportion of Total Sequences
18, 1, 4, 15	Collection, Carry, Kick, Possession	0.029
18, 4, 15	Collection, Kick, Possession	0.019
18, 3, 1, 4, 15	Collection, Pass, Carry, Kick, Possession	0.017
18, 3, 4, 15	Collection, Pass, Kick, Possession	0.006
18 1 2 23 3 4 15	Collection, Carry, Tackle, Ruck, Pass, Kick, Possession	0.006

Table 6.8: Most frequently occurring generated sequences using Model 2.9.

Sequence	Translation	Proportion of Total Sequences
18, 1, 4, 15	Collection, Carry, Kick, Possession	0.036
18, 4, 15	Collection, Kick, Possession	0.032
18, 3, 4, 15	Collection, Pass, Carry, Possession	0.019
18, 3, 1, 4, 15	Collection, Pass, Carry, Kick, Possession	0.013
18 1 2 23 3 4 15	Collection, Carry, Tackle, Ruck, Pass, Kick, Possession	0.009

Table 6.9: Most frequently occurring generated sequences using Model 2.15.

From these results, it is evident that these models perform well in generating realistic short sequences. Table 6.10 below displays the most frequently occurring long sequences from the original data set, where a long sequence is defined as a sequence longer than 20 actions. This table shows the numerical sequence of actions and indicates the frequency of that sequence in the data set. The longest, most frequently occurring sequences describe situations in a game where there are multiple resets of scrums (actions ‘28’ and ‘29’).

Sequence	Count
26, 17, 5, 28, 28, 29, 29, 26, 5, 28, 28, 29, 29, 26, 5, 28, 28, 29, 29, 3, 4, 15	4
26, 5, 28, 28, 29, 29, 26, 5, 28, 28, 29, 29, 26, 5, 28, 28, 29, 29, 26, 7, 4, 15	3
26, 5, 28, 28, 29, 29, 26, 5, 28, 28, 29, 29, 26, 7, 26, 11, 14, 18, 3, 1, 2, 2, 23, 4, 15	3

Table 6.10: Most frequently occurring long sequences in original data set.

Tables 6.11 to 6.14 respectively show the three most frequently occurring long sequences from the generated sequences using the four chosen models. Further analysis of these sequences indicates that these models can either generate identical long sequences as the original data set or generate long sequences that are relatively similar to those from the original data set but are still realistic. From this, it is evident that these four models can accurately generate realistic short sequences as well as longer sequences.

Sequence	Count
26, 5, 29, 29, 28, 28, 26, 5, 29, 29, 28, 28, 26, 5, 29, 29, 28, 28, 26, 7, 4, 15	4
26, 5, 28, 28, 29, 29, 26, 5, 28, 28, 29, 29, 26, 5, 28, 28, 29, 29, 26, 7, 4, 26, 6, 27, 24, 7, 15	2
26, 5, 28, 28, 29, 29, 26, 5, 28, 28, 29, 29, 26, 7, 17, 4, 26, 17, 6, 27, 24, 7, 15	2

Table 6.11: Most frequently occurring long sequences using Model 1.3.

Sequence	Count
26, 17, 5, 28, 28, 29, 29, 26, 5, 28, 28, 29, 29, 26, 5, 28, 28, 29, 29, 3, 4, 15	2
26, 5, 29, 29, 28, 28, 26, 5, 29, 29, 28, 28, 26, 5, 29, 29, 28, 28, 26, 7, 4, 15	2
26, 5, 28, 28, 29, 29, 26, 7, 4, 26, 6, 27, 24, 7, 21, 20, 3, 1, 3, 2, 15	2

Table 6.12: Most frequently occurring long sequences using Model 1.9.

Sequence	Count
26, 17, 5, 28, 28, 29, 29, 26, 5, 28, 28, 29, 29, 26, 5, 28, 28, 29, 29, 3, 4, 15	4
26, 5, 29, 29, 28, 28, 26, 5, 29, 29, 28, 28, 26, 7, 4, 26, 6, 27, 24, 7, 15	3
26, 17, 5, 28, 28, 29, 29, 26, 5, 28, 28, 29, 29, 26, 7, 26, 11, 14, 18, 1, 2, 23, 4, 15	2

Table 6.13: Most frequently occurring long sequences using Model 2.9.

Sequence	Count
26, 5, 28, 28, 29, 29, 26, 5, 28, 28, 29, 29, 2, 12, 2, 10, 1, 3, 2, 12, 10, 2, 1, 10, 4, 15	2
26, 5, 28, 28, 29, 29, 26, 5, 28, 28, 29, 29, 26, 5, 28, 28, 29, 29, 3, 4, 15	2
26, 5, 28, 28, 29, 29, 3, 10, 1, 3, 2, 12, 1, 3, 10, 1, 9, 10, 2, 12, 15	2

Table 6.14: Most frequently occurring long sequences using Model 2.15.

6.1.6 Best Performing Model

Based on the results thus far, one of the four chosen LSTM models was selected to be used for the remainder of the study. It was evident from these results that all four of the models were able to generate sequences that emulate realistic sequences of actions in a game of rugby. Based on the criteria for evaluating which model performed the best, it is difficult to define a model that outperforms the rest. Based on the distribution of actions from the generated actions for the four chosen models, Model 2.15, a two-layer stacked LSTM with 32 batch size and 128 units, showed a distribution of actions that best emulated the distribution of actions from the original data set. This model also had the lowest loss value.

In terms of the ability of a model to generate the correct next actions after a particular action, each model performed well, with Model 2.9 failing to generate and incorrectly generating the least number of actions. The differences in models using this criterion were minimal, such that the authors of this study were unable to differentiate between the performances of the models. Based on the most frequently occurring generated sequences by each model, once again, all four models performed well in generating short and long realistic sequences. Based on the lowest loss value, and the distribution of actions that best emulated that of the original data set, Model 2.15 was deemed to be the best performing model. It was used for the remainder of the study, although the use of the other models would likely not lead to largely differing results.

6.2 Winning versus Losing Teams

6.2.1 Performance comparison

The data from the winning and losing teams' attacking sequences were used to create models to learn how the respective teams play. The process of data preparation for this objective was explained in Section 4.2.1. The analysis begins by comparing the distributions of all the actions between the winning and losing teams. These distributions are displayed in Figure 6.12.

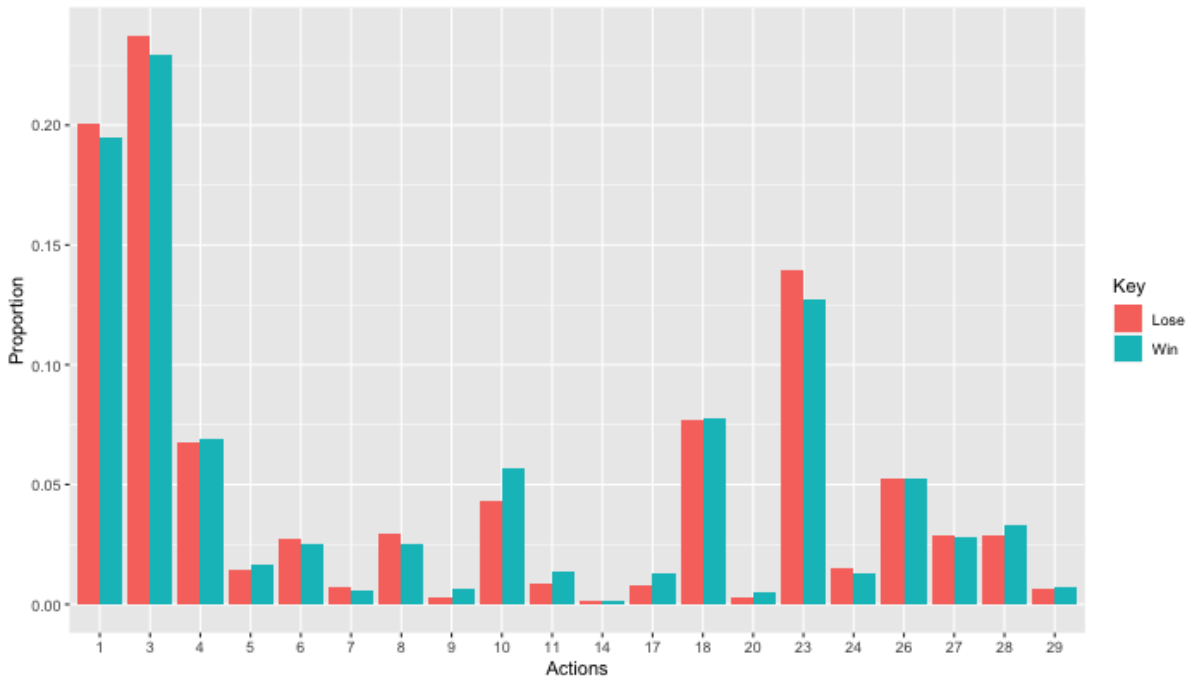


Figure 6.12: Distribution: Winning teams vs. Losing teams.

From Figure 6.12 one can note a similar distribution to the data in Figure 4.7. This demonstrates that the neural network performed well in generating the actions within the sequences. The winning sides contained proportionally more attacking sequences, tries and goal kicks compared to the losing sides, which makes logical sense as these actions contain points and ultimately determine the winner. The losing teams contained proportionally more carries, passes and rucks compared to winning teams. To understand whether the sequences are realistic, the length of sequences and the sequential nature of the actions were investigated. A box plot showing the distributions of sequence lengths for winning and losing teams is displayed in Figure 6.13.

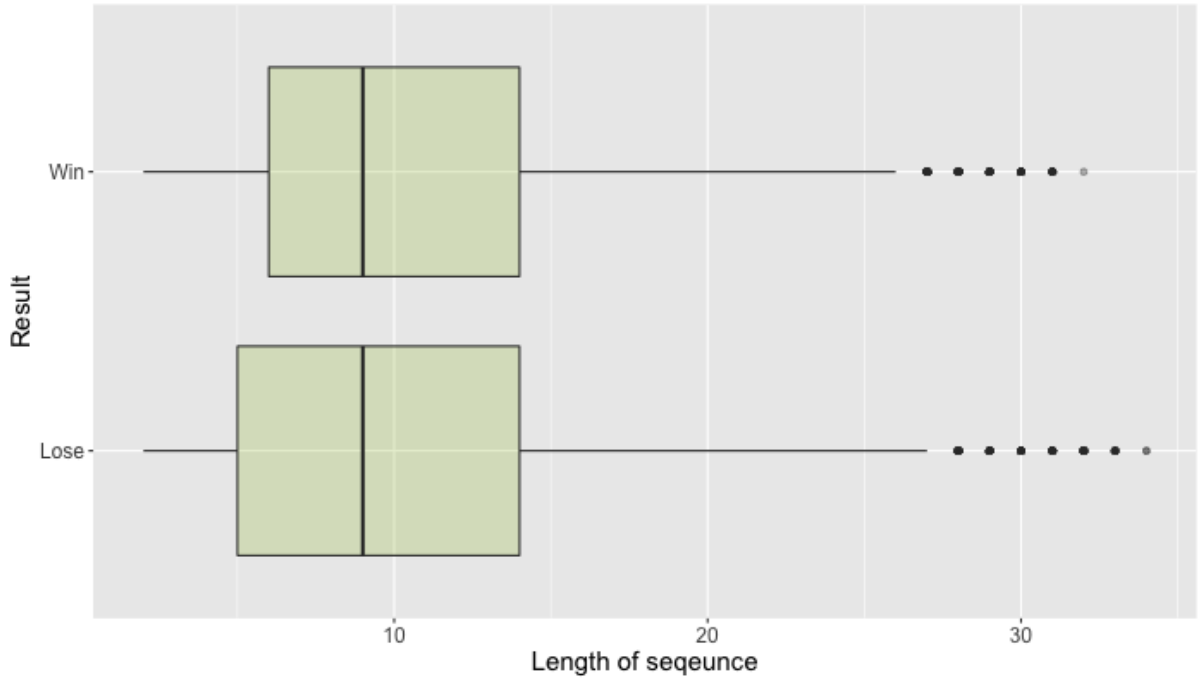


Figure 6.13: Box Plot of Length of Sequences by Winning and Losing Teams.

The summary statistics for the winning and losing team sequence lengths are shown in Table 6.15. The distribution of the lengths of sequences differs between winning and losing teams. The inter-quartile range for the winning team model is smaller than that of the losing team, which suggests that winning teams are, on average, able to hold onto the ball more consistently and do not give away possession after only a few sequences.

	Winning Teams	Losing teams
Min.	2	2
1st Quartile:	6	5
Median:	9	9
Mean:	10.78	10.76
3rd Quartile:	14	14
Max:	33	34

Table 6.15: Length of Sequences Summary Statistics.

Other than the 1st quartile difference, the summary statistics appear to be very similar. The next part of the analysis compared the sequential nature of the actions in sequences. The first action investigated was ‘attacking qualities’. Figure 6.14 displays the actions that precede this action in the generated sequences for the winning and losing teams. This plot highlights some similarities between the teams, as well as some significant differences. It is found that both show similar responses after a lineout (action ‘27’) and after a carry (action ‘1’). The ‘pass’ (action ‘3’) variable illustrates that losing teams

tend to pass more before attacking qualities. Losing teams also have more collections of the ball before attacking qualities.

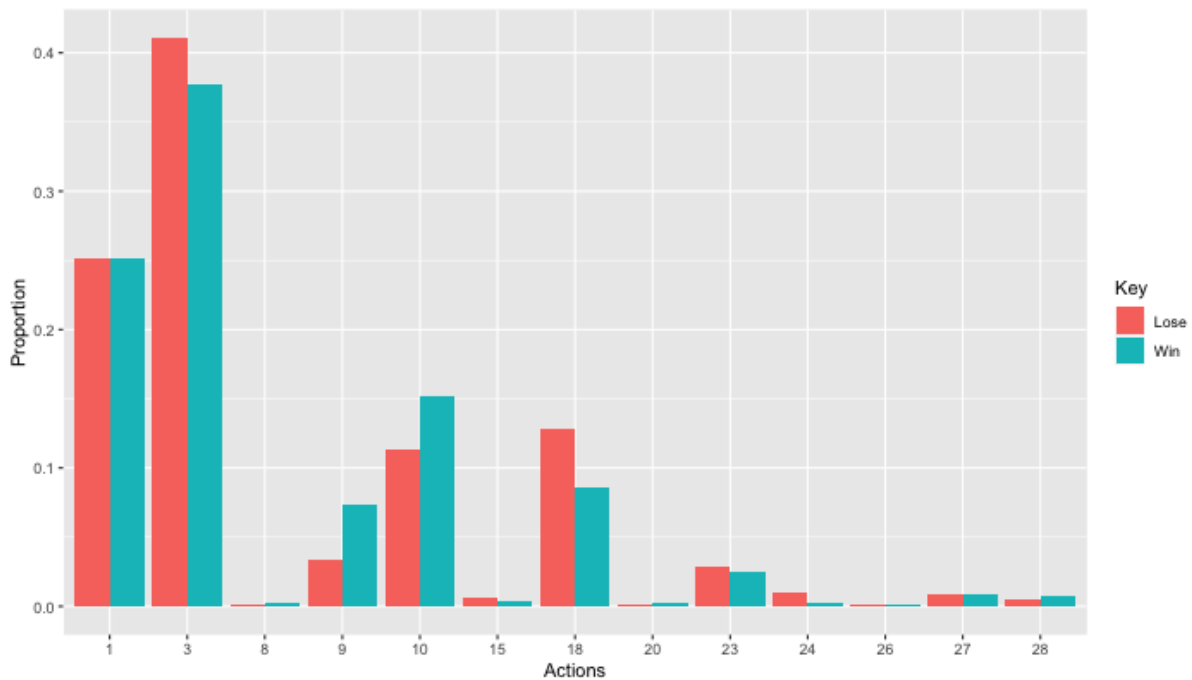


Figure 6.14: Actions before attacking qualities: Winning teams vs. Losing teams.

The actions after attacking qualities were investigated to understand the result of this action for both teams. These next actions are displayed in Figure 6.15. The two distributions displayed appear to be very similar. The differences lie in the ‘carry’ (‘1’) action and the ‘possession’ (‘15’) action. The losing teams tend to carry the ball after attacking qualities more than the winning teams, while the winning teams attacking plays end in a change of possession more often than the losing teams. The high proportion of the carry action in losing teams may indicate that losing teams rely more on individual brilliance than other aspects of the game.

The next action investigated was the ‘kick’ action. This action was analysed as it is a common talking point in South African rugby circles. Figure 6.16 displays the actions that preceded the ‘kick’ action. Winning teams were found to kick more after a ‘ruck’ (action ‘23’) compared to losing teams. The losing teams were found to kick after the ‘pass’ action more than winning teams. Both teams appeared to kick a similar number of times after an offensive scrum.

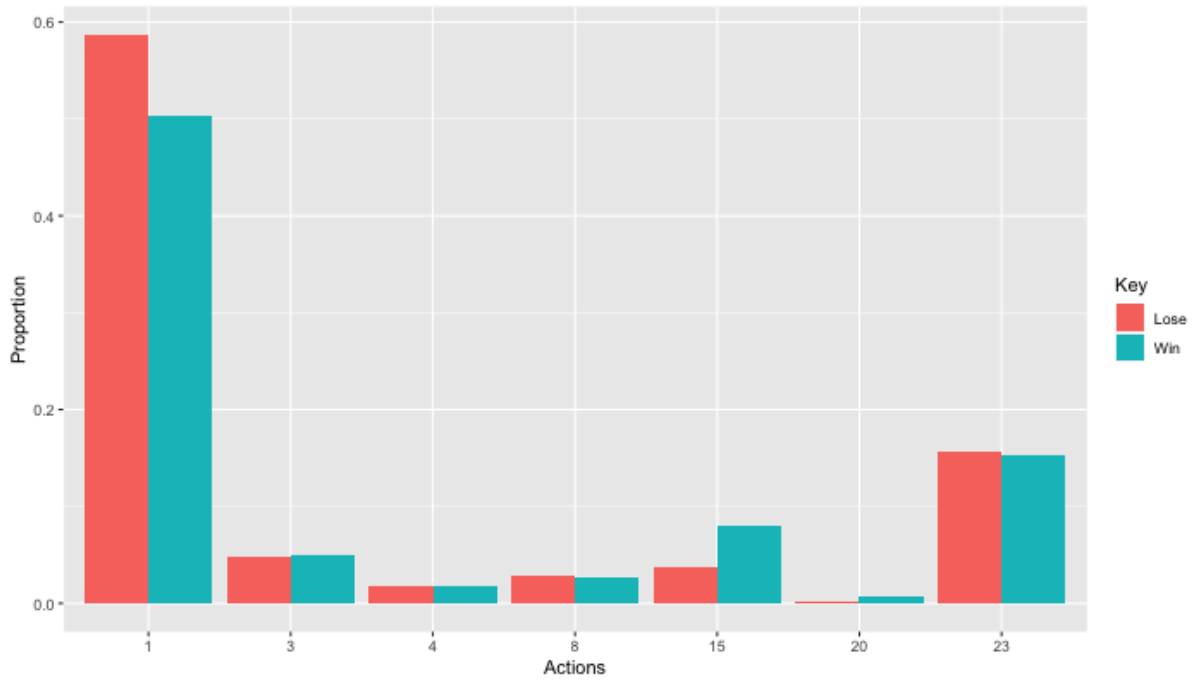


Figure 6.15: Actions after attacking qualities: Winning teams vs. Losing teams.

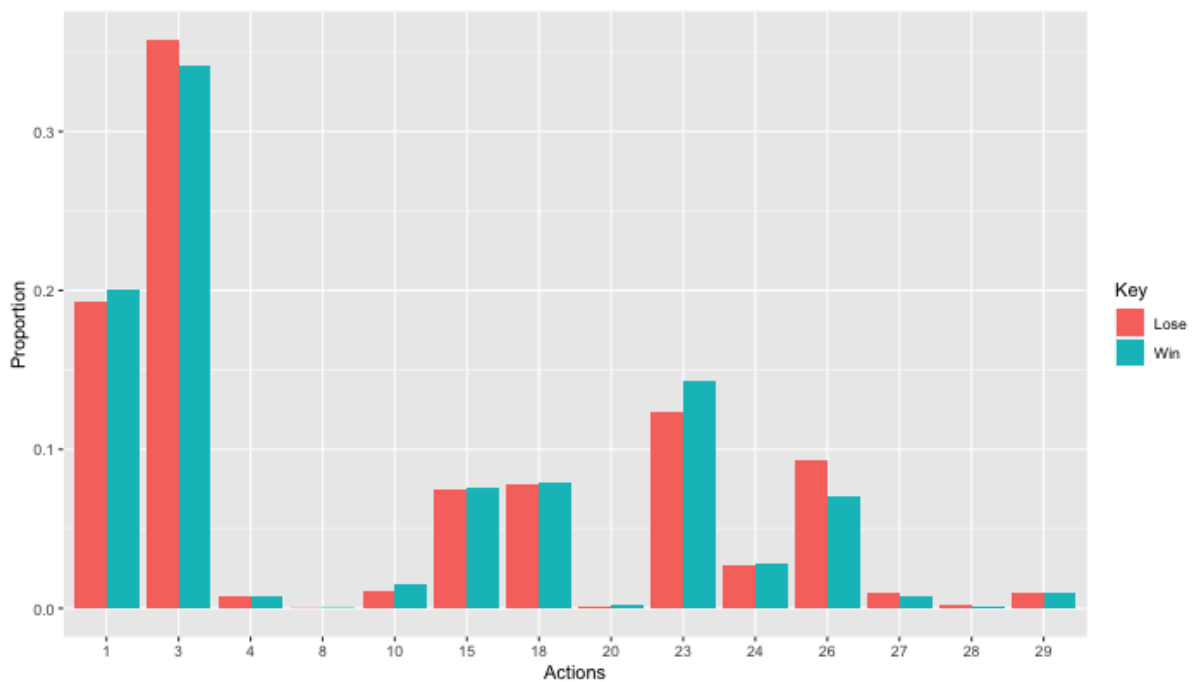


Figure 6.16: Actions before a kick: Winning teams vs. Losing teams.

Figure 6.17 displays the distribution of actions that follow a ‘kick’. This distribution is used to understand the outcome of the ‘kick’ action. Around 70% of the time, kicks were followed by the a change in possession (‘15’). This infers that the sequence ended after a kick. The winning teams collected the ball more, which means they retained the ball after a kick. These characteristics fall in line with what is expected when comparing

winning versus losing teams.

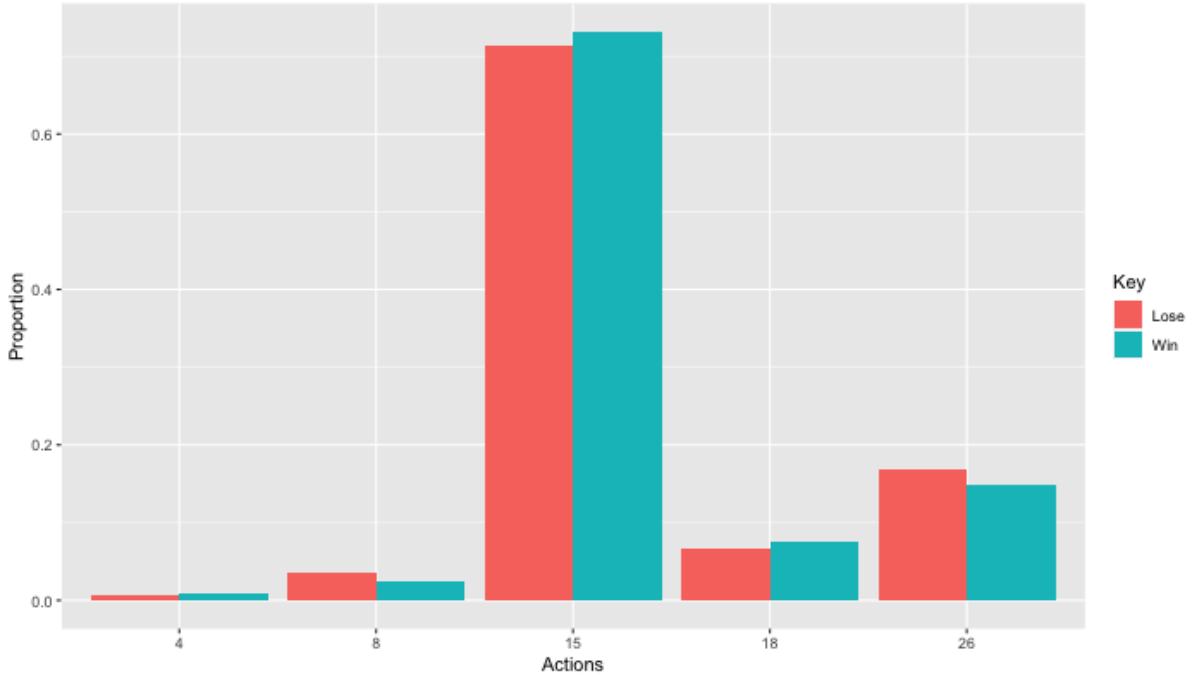


Figure 6.17: Actions before a kick: Winning teams vs. Losing teams.

Using rugby domain knowledge, the results of the winning and losing team actions appear to be logical. This strengthens the argument that the models are capable of learning patterns in rugby sequences and thus can generate realistic sequences of play.

6.2.2 Game Simulation

Games were simulated between winning and losing teams to demonstrate how research of this kind may be applied in the future. On average, each team has 43 possessions (sequences) per game. For the simulation, the average number of sequences in a game are sampled from each of the winning and losing models' distributions. This method assumes possession alternates between teams. For computational purposes, 10000 random samples of 43 from the 10000 generated sequences for each of winning and losing teams are used to create 10000 games.

Points for a try or penalty kick are then allocated to the samples depending on which distribution they originate from. In rugby, a try counts as five points, and a penalty kick counts as three points. If a team scores a try, they are given an opportunity to gain an additional two points by kicking the ball over the posts; this is called a conversion. The sequences generated do not contain indicators of whether a penalty kick or a conversion are successful or not. Nevertheless, the average success rate for conversions and penalty

kicks are calculated from the data set for winning and losing teams. The average kick success rate can be noted in Table 6.16 where both the winning and losing teams have similar conversion success rates of 72.17% and 72.27%, respectively. The winning teams have a much better penalty kick success rate of 79.56% compared to losing teams, that have a 72.13% success rate.

	Winning teams	Losing teams
Conversion kick success rate	72.17%	72.27%
Penalty kick success rate	79.56%	72.13%

Table 6.16: Goal kick success rates from data set.

The total points are calculated, compared, and a winner is determined. After running the simulation for 10000 games, the following results are achieved:

	Model	
Result:	Winning team:	Losing team:
Wins	7467	2267
Losses	2267	7467
Draws	266	266
Avg. Score	23.2474	13.8539
Median Score	23	13
Avg. tries	2.4999	1.2323
Median tries	2	1
Avg. goal kicks	5.8573	4.1792
Median goal kicks	6	4

Table 6.17: Results of winning team model vs. losing team model.

The model trained with the winning team data beat the model trained with the losing team data 74.67% of the time. The two models drew 2.66% of the time, and the losing team model beat the winning team model 22.67% of the time. These results were to be expected as it was noted in Figure 6.12 that the winning model contained more point actions.

Comparing the median statistics to the average (mean) statistics one can note similar results and thus no obvious skewness. The winning team model scored on average 23.25 points per game, and the losing team scored 13.85 points per game. The authors note that these scores are realistic in rugby union. Comparing the average number of tries per game, one can note that the winning team model scored 2.50 tries per game, and the losing team model scored 1.23 tries per game. The winning team model also contained more goal kick opportunities with 5.86 goal kicks compared to the 4.18 goal kicks per game for the losing team model.

It is acknowledged that this demonstration treats the phases independently, and therefore is not a realistic representation of a game. However, this technique provides an elementary illustration of how research of this nature can be used in the future.

Discussion

7.1 Results Summary

Several 1-layer and 2-layer LSTM models were built using various hyper-parameters. Based on the lowest loss criteria, four of these models were selected to conduct further analysis. It was found that using temperature values of 0.2 and 0.4 led to sequences that were often unrealistic for a game of rugby, whereas a temperature of 1 generated more realistic sequences. The models were first assessed by obtaining the distribution of actions from 10000 generated sequences and comparing this to the distribution of actions from the original data set. All four models were able to generate proportions of actions similar to that of the original data set. The single-layer models showed the tendency to under-generate the most prominent actions in the data set and over-generate those actions that occur less. These differences were, however, minimal and showed no impact on how realistic the generated sequences were.

The two-layer models showed distributions that match the original data set distribution more closely; however, it was observed that Model 2.9 generated a higher proportion of action '3' than action '2', which is not seen in the original data set. Model 2.15 displayed a distribution of actions that was the most accurate when compared to the original data.

Next, the models were compared based on how accurately they could generate actions following particular actions in a sequence. Based on the three actions chosen, the four models displayed good results. The distributions of next actions for the four models showed a similar distribution to that of the original data set. The only differences occurred when a model failed to or incorrectly generated next actions. This typically occurred with next actions that were of a very small proportion in the original data set. Models showed no apparent differences in performance on this criteria.

The most frequently occurring short and long sequences of each model were assessed and compared to those from the original data set. The four models displayed the ability to generate realistic short and long sequences. Based on the evaluation criteria, Model 2.15 was deemed to perform best when generating sequences and was the model chosen to conduct analysis on winning and losing teams.

The comparison of winning versus losing teams provided additional evidence of the effectiveness of LSTMs in rugby union sequence generation. The authors used their domain

knowledge of rugby to note that the differences between the winning team model and the losing team model made logical sense in the domain of rugby union. Further, 10000 games were simulated. For 7467 of these simulations, the models based on winning team data generated sequences that resulted in higher scores than those based on losing team sequences. The average scores of these simulations appeared to be realistic.

7.2 Future Considerations

Future work regarding the use of neural networks in rugby union can look into building LSTM models that take into account the location of the play on the field. The models used in this study did not take into account the positional ‘XY’ locations of the action. By taking field location into account, the models could be able to generate more specific sequences. For example, if a defending team has possession of the ball within five metres of their in-goal area, they are more likely to kick than if they were situated in other positions on the field.

The best performing model was a stacked 2-layer LSTM model. Therefore the performance of deeper models could lead to better results. Models using an embedding layer would also have been beneficial to investigate and could be compared to models that use one-hot encoding.

From observing the generated sequences, it was evident that sequences were primarily independent of each other, meaning that a sequence did not make logical sense given the sequence before. In order to conduct effective game simulation, in which sequences are generated depending on the sequence that comes before, one would have to formulate an LSTM generating algorithm that implements the ability for one sequence to depend on the previous sequence.

7.3 Future Applications

This study has shown that LSTMs are effective in generating realistic sequences for rugby union. The ability of a model to distinguish between winning and losing sides can be extended to the assessment of particular teams. This would be of particular interest to rugby coaches and team analysts, as with sufficient data, they could use an LSTM to generate sequences based on their teams’ previous data as well as their opponents’ previous data. Analysis of sequences generated may be able to indicate what their team or other teams are more likely to do in a game of rugby, and based on these results, they

could make various tactical decisions. Additionally, they could build models similar to those suggested in Section 7.2 to understand what opposition teams do dependent on the position on the field where a play is taking place. Therefore this study can be seen as a stepping stone for further rugby analysis.

These results may also be of interest to sports betting companies, as the results of game simulation may offer insight into which teams are most likely to win against other teams. Results based on generated sequences may also be useful in assessing more detailed statistics of a game, such as the number of points scored by specific teams or the number of kicks per game. These statistics would be of use in formulating certain odds of events happening in a game.

7.4 Limitations

7.4.1 Cloud Computing

Amazon Web Services (AWS) provided a secondary platform to train models and generate sequences but it was found that even with a GPU-enabled instance type, processing speed did not improve significantly compared to a modern laptop. Thus the use of AWS only acted as a secondary computer and not a high performance computer. With a larger budget, one can maybe experiment on more powerful instance architectures.

7.4.2 Time Constraints

Running 36 different types of models for the hyperparameter experiments took a significant amount of time. The least complex model took roughly 2 hours to train. Generating the 10000 sequence needed to conduct analysis was also computationally demanding.. The longest model to train, the 2-layer stacked LSTM with a batch size of 32 and 128 units, ran for roughly 20 hours before terminating. Along with deadlines and other required deliverables, there was only enough time to explore two model architectures.

Conclusion

This study set out to determine whether recurrent neural networks, more specifically long short-term memory networks, can be used to generate realistic sequences of play in rugby union. Several LSTM models with varying hyper-parameters were built to achieve this objective. The sequences generated using these models were compared to the sequences obtained from the original data set to assess model performance. It was found that the four models with the lowest loss values could generate the same proportion of action variables as the original data set. The order of actions in these sequences matched what one would expect in a game of rugby. These four models were able to generate short and long sequences that are common in a realistic game of rugby.

After comparing the four chosen models using various criteria, there were minor differences in their ability to generate realistic sequences of play. The model that performed best when generating sequences was a 2-layer stacked LSTM with a 32 batch size and 128 units, using a temperature value of 1. This model had the lowest loss of 0.8719 and showed a distribution of actions almost identical to that of the original data set. Using domain knowledge of rugby, the generated sequences were assessed. They were deemed to generate sequences seen in the original data set and unique sequences of play that make logical sense in a game of rugby.

The secondary objective of determining differences in sequences of play between winning and losing sides found that there were indeed notable differences. The winning teams' attacking sequences contained more point achieving actions and contained more attacking qualities. The simulation of games between the models found that winning teams would win roughly 80% of matches simulated, which makes logical sense. These results proved that an LSTM has the ability to learn actions of play associated with winning and losing teams and generate differing sequences accordingly.

Based on these results, this study has shown that LSTMs can be used to generate realistic sequences of play in rugby union; hence the first objective of this study was achieved. The best performing LSTM model accurately differentiated between data from winning and losing teams and generated sequence accordingly. Therefore, this study has shown that there is a difference between generated sequences based on winning and losing teams, which can be used to assess what sequences of play winning teams are most likely to perform. Hence, the second objective of this study was deemed successful.

Although often overlooked, this study has found that the sequential nature of rugby union

can be assessed using LSTMs. This provides an excellent opportunity to conduct further statistical analysis in the domain of rugby union.

Bibliography

- [1] N. Watson, S. Hendricks, T. Stewart, and I. Durbach, “Integrating machine learning and decision support in tactical decision-making in rugby union,” *Journal of the Operational Research Society*, 2020.
- [2] M. Olsofka, “Why is data analytics so important in sports?” 2018. [Online]. Available: <https://www.samford.edu/sports-analytics/fans/2018/Why-is-Data-Analytics-So-Important-in-Sports>
- [3] J. Lopez, “The role of sports analytics in modern gambling in 2021,” 2021. [Online]. Available: <https://thenationroar.com/2021/03/22/role-of-sports-analytics-in-modern-gambling/>
- [4] Harrod Sport, “Rugby pitch dimensions,” 2019, [Online; accessed November 4, 2019]. [Online]. Available: <https://www.harrod sport.com/uploads/wysiwyg/img/1-5.jpg>
- [5] L. Aslett, “Rstudio server amazon machine image (ami) - louis aslett,” 2020. [Online]. Available: https://www.louisaslett.com/RStudio_AMI/
- [6] A. Graves, N. Jaitly, and A. Mohamed, “Hybrid Speech Recognition With Deep Bidirectional LSTM,” *University of Toronto*, pp. 1–5, 2013.
- [7] D. Soutner and L. Muller, “Application of LSTM Neural Networks in Language Modelling,” *University of West Bohemia*, 2013.
- [8] W. Zou, H. Li, and M. Wang, “Hierarchical LSTM for Sign Language Translation,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.
- [9] M. Chen, G. Ding, S. Zhao, H. Chen, J. Han, and Q. Lui, “Reference Based LSTM for Image Captioning,” *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, vol. 17, no. 1, 2017.
- [10] A. Graves, “Generating sequences with recurrent neural networks,” 2014.
- [11] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, “Social LSTM: Human Trajectory Prediction in Crowded Spaces,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 961–971, 2016.
- [12] M. S. Islam, S. S. Sharmin Mousumi, S. Abujar, and S. A. Hossain, “Sequence-to-sequence bangla sentence generation with lstm recurrent neural networks,” *Procedia Computer Science*, vol. 152, pp. 51–58, 2019, international Conference

- on Pervasive Computing Advances and Applications- PerCAA 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1877050919306775>
- [13] S. Mangal, P. Joshi, and R. Modak, “LSTM vs. GRU vs. bidirectional RNN for script generation,” *CoRR*, vol. abs/1908.04332, 2019. [Online]. Available: <http://arxiv.org/abs/1908.04332>
- [14] A. Karpathy, “The Unreasonable Effectiveness of Recurrent Neural Networks,” 05 2015. [Online]. Available: <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>
- [15] J. G. L.-P. Sebastian Garcia-Valenciaa, Alejandro Betancourtb, “Sequence generation using deep recurrent networks and embeddings: A study case in music,” 2020.
- [16] R. A. Nyquist and D. Pettersson, “Football match prediction using deep learning,” 2017.
- [17] M. Harmon, A. Ebrahimi, P. Lucey, and D. Klabjan, “Predicting Shot Making in Basketball Learnt from Adversarial Multiagent Trajectories,” *stat.ML*, 2016.
- [18] R. Shah and R. Romijnders, “Applying deep learning to basketball trajectories,” *CoRR*, vol. abs/1608.03793, 2016. [Online]. Available: <http://arxiv.org/abs/1608.03793>
- [19] J. Verpalen, “Predicting player movements in soccer using Deep Learning,” 2019.
- [20] S. Goddijn, E. Moshkovich, and R. Challa, “A Sure Bet: Predicting Outcomes of Football Matches,” 2018.
- [21] N. M. Jones, S. D. Mellalieu, and N. James, “Team performance indicators as a function of winning and losing in rugby union.” *International Journal of Performance Analysis in Sport*, vol. 4, no. 1, pp. 61–71, 2004. [Online]. Available: <https://doi.org/10.1080/24748668.2004.11868292>
- [22] C. Wright, S. Atkins, and B. Jones, “An analysis of elite coaches’ engagement with performance analysis services (match, notational analysis and technique analysis),” *International Journal of Performance Analysis in Sport*, vol. 12, no. 2, pp. 436–451, 2012. [Online]. Available: <https://doi.org/10.1080/24748668.2012.11868609>
- [23] N. Watson, I. Durbach, S. Hendricks, and T. Stewart, “On the validity of team performance indicators in rugby union,” *International Journal of Performance Analysis in Sport*, vol. 17, no. 4, pp. 609–621, 2017.
- [24] C. Olah, “Understanding lstm networks,” 2015. [Online]. Available: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

-
- [25] M. Phi, “Illustrated guide to lstm’s and gru’s: A step by step explanation,” Jun 2020. [Online]. Available: <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>
 - [26] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, “Gradient flow in recurrent nets: the difficulty of learning long-term dependencies,” *A Field Guide to Dynamical Recurrent Neural Networks*, 03 2003.
 - [27] S. Sharma, S. Sharma, and A. Athaiya, “Activation functions in neural networks,” *towards data science*, vol. 6, no. 12, pp. 310–316, 2017.
 - [28] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2017.
 - [29] P. Bosch and S. Bhulai, “Predicting the winner of NFL-games using Machine and Deep Learning,” 2018.
 - [30] L. Zhang, C. Xu, Y. Gao, Y. Han, X. Du, and Z. Tian, “Improved Dota2 Lineup Recommendation Model Based on a Bidirectional LSTM,” *Tsinghua Science and Technology*, vol. 25, no. 6, pp. 712–720, 2020.
 - [31] D. Neil, M. Pfeiffer, and S.-C. Liu, “Phased lstm: Accelerating recurrent network training for long or event-based sequences,” 2016.
 - [32] A. Vaswani, R. Ganguly, H. Shah, S. Ranjit, S. Pandit, and S. Bothara, “An autoencoder based approach to simulate sports games,” 2020.
 - [33] F. Chollet and J. Allaire, *Deep learning with R*. Manning, 2018.
 - [34] J. Allaire and Y. Tang, *tensorflow: R Interface to 'TensorFlow'*, 2021, r package version 2.6.0. [Online]. Available: <https://CRAN.R-project.org/package=tensorflow>
 - [35] J. Allaire and F. Chollet, *keras: R Interface to 'Keras'*, 2021, r package version 2.6.1. [Online]. Available: <https://CRAN.R-project.org/package=keras>
 - [36] H. Wickham and J. Bryan, *readxl: Read Excel Files*, 2019, r package version 1.3.1. [Online]. Available: <https://CRAN.R-project.org/package=readxl>
 - [37] H. Wickham, *stringr: Simple, Consistent Wrappers for Common String Operations*, 2019, r package version 1.4.0. [Online]. Available: <https://CRAN.R-project.org/package=stringr>
 - [38] S. Andri et mult. al., *DescTools: Tools for Descriptive Statistics*, 2021, r package version 0.99.43. [Online]. Available: <https://cran.r-project.org/package=DescTools>
 - [39] L. Henry and H. Wickham, *purrr: Functional Programming Tools*, 2020, r package version 0.3.4. [Online]. Available: <https://CRAN.R-project.org/package=purrr>

- [40] H. Wickham, R. François, L. Henry, and K. Müller, *dplyr: A Grammar of Data Manipulation*, 2021, r package version 1.0.7. [Online]. Available: <https://CRAN.R-project.org/package=dplyr>
- [41] H. Wickham, *tidyr: Tidy Messy Data*, 2021, r package version 1.1.4. [Online]. Available: <https://CRAN.R-project.org/package=tidyr>

Appendix A

Examples of Sequences Generated

A.1 Single-layer LSTM Sequences

Model	Sequence Example
1.1	26 6 27 3 3 3 1 3 2 1 2 23 3 1 2 2 23 3 3 1 2 23 3 1 2 2 23 3 1 2 23 3 3 3 1 2 23 3 1 2 2 23 3 4
1.2	26 6 27 3 1 2 23 3 1 2 23 3 1 2 23 3 1 2 2 23 3 1 2 2 23 7
1.3	26 11 14 18 1 2 23 3 3 8
1.4	18 1 2 23 3 3 1 2 23 3 1 2 23 3 1 2 23 3 1 2 23 3 3 1 2 23 3 1 2 23 3 1 2 23 3 3 1 2 23 3 8
1.5	26 6 27 18 1 2 23 3 1 2 23 3 3 1 2 23 3 1 2 23 3 3 1 2 2 23 3 1 2 23 3 3 1 2 23 3 1 2 2 23 3 1 2 2 23 18 7
1.6	26 26 11 14 18 2 2 23 4 8
1.7	26 6 27 3 3 1 2 23 3 1 2 23 3 1 2 23 3 1 2 23 3 1 2 23 3 1 2 23 3 1 2 23 3 1 2 2 23 3 1 2 23 3 1 2 23 7
1.8	18 1 2 23 3 1 2 2 23 3 4
1.9	26 5 28 28 29 29 26 5 28 28 29 29 26 7 4 15
1.10	26 17 5 29 29 28 28 26 5 29 29 28 28 26 7 4 26 6 27 24 1 2 23 3 1 2 23 3 1 2 23 3 1 2 23 3 1 2 23 3 1 2 23 3 3 8
1.11	18 10 1 2 12 2 23 3 4
1.12	18 1 2 23 3 1 2 23 3 1 2 2 23 3 1 2 23 3 1 7
1.13	26 6 27 3 3 1 2 23 3 1 2 23 3 1 2 23 3 1 2 23 3 3 1 2 23 3 1 2 23 3 1 2 23 3 1 2 23 3 1 2 23 3 1 2 23 3 3 8
1.14	26 4 26 6 27 3 3 3 1 8 2
1.15	26 6 27 3 1 2 7 23 18
1.16	26 17 5 29 29 28 28 26 7 4 26 6 27 24 1 2 23 3 1 2 2 23 3 1 2 2 23 3 3 1 2 23 3 3 1 2 23 3 1 2 23 3 1 2 2 23 7
1.17	18 1 3 2 1 2 23 3 3 3 3 2 1 23 3 3 1 2 2 23 3 3 3 1 3 2 2 18 1 2 23 3 1 2 23 3 3 1 2 23 8
1.18	26 5 28 28 29 29 1 2 23 1 2

A.2 2-layer stacked LSTM Sequences

Model	Sequence Example
2.1	26 6 27 3 3 1 2 23 3 1 2 23 3 1 2 23 3 1 2 2 23 3 1 2 23 3 1 2 2 23 3 3 3 1 2 23 3 1 2 2 23 3 3 1 3 8
2.2	18 1 2 23 3 3 1 3 1 4 18 8
2.3	26 6 27 24 3 3 10 1 2 23 1 8
2.4	26 6 27 3 3 1 2 23 3 1 2 23 3 1 2 23 3 3 1 2 23 3 1 2 23 3 1 2 23 3 3 1 2 23 3 1 2 23 3 3 1 2 23 3 3 8
2.5	18 18 1 2 23 8
2.6	26 4 26 17 6 27 3 3 3 3 4
2.7	26 6 27 3 3 1 2 2 23 3 1 2 23 3 3 1 2 2 23 3 1 2 2 23 3 1 2 23 3 1 2 23 3 1 2 23 3 1 2 23 3 3 1 2 23 3 1 2 23
2.8	18 1 2 23 3 3 1 2 23 3 1 3 2 1 2 23 1 2 23 18 7
2.9	4 3 1 18 1 10 2 2 23 3 3 3 10 10 10 1 10 2 12 2 12 2 12 2 23 3 8
2.10	26 6 27 3 3 3 1 2 23 3 3 1 2 23 3 3 1 2 23 3 1 2 23 3 1 2 2 23 3 1 2 2 23 3 1 2 23 3 1 2 23 3 1 2 23 3 3 8
2.11	26 17 17 5 28 28 29 29 3 3 1 2 2 23 3 1 2 23 3 1 2 23 3 1 2 23 3 1 2 23 3 4 18 1 2 2 23 3 1 2 23 3 3 1 2 23 18 7
2.12	26 5 28 28 29 29 1 8 2 24 2
2.13	26 17 5 29 29 28 28 26 7 4 26 6 27 3 3 3 1 8 2
2.14	18 3 1 2 23 3 1 2 2 23 3 3 10 1 10 2 12 2 23 3 1 2 23 3 1 2 23 3 1 2 23 3 1 2 23 3 3 10 1 2 12 2 7
2.15	26 6 27 3 3 1 2 2 23 3 1 10 2 2 23 3 1 2 2 23 3 1 2 2 23 3 3 1 10 2 12 2 23 8
2.16	1 1
2.17	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 2 2 1 1 2
2.18	1 2 1 1 1 1 1 2 1 1 1 2 1 2 1 2 1 1 2 1 1 24 1 1 1 1 18 2 1 1 1 1 2 1 1

Appendix B

Link to GitHub Repository

The code used for this paper can be found in the following GitHub repository:

`https://github.com/grahamdavies15/Stats-Rugby-Project`