Predicting social unrest events in South Africa using LSTM neural networks

Samantha Zambezi

ZMBSAM001

Supervisor:

Juwa Nyirenda

Submitted in partial fulfilment of the requirements of the degree of Master of Data Science

Department of Statistical Sciences

July 2020

Abstract

This thesis demonstrates an approach to predict the count of social unrest events in South Africa. A comparison is made between traditional forecasting approaches and neural networks; the traditional forecast method selected being the Autoregressive Integrated Moving Average (ARIMA model). The type of neural network implemented was the Long Short-Term Memory (LSTM) neural network. The basic theoretical concepts of ARIMA and LSTM neural networks are explained and subsequently, the patterns of the social unrest time series were analysed using time series exploratory techniques. The social unrest time series contained a significant number of irregular fluctuations with a non-linear trend. The structure of the social unrest time series suggested that traditional linear approaches would fail to model the non-linear behaviour of the time series. This thesis confirms this finding. Twelve experiments were conducted, and in these experiments, features, scaling procedures and model configurations are varied (i.e. univariate and multivariate models). Multivariate LSTM achieved the lowest forecast errors and performance improved as more explanatory features were introduced. The ARIMA model's performance deteriorated with added complexity and the univariate ARIMA produced lower forecast errors compared to the multivariate ARIMA. In conclusion, it can be claimed that multivariate LSTM neural networks are useful for predicting social unrest events.

Acknowledgements

This journey was not easy and required commitment, patience and some sacrifice. I would not have been successful if it was not for the motivation and support that I received from my husband Johann van Der Merwe. I would also like to thank my supervisor Juwa Nyirenda for making me better at research and for the guidance that I received.

Table of Contents

List of Figures

List of Tables

Chapter One

Introduction

1.1 Context and Background to the study

Over the past few years, deep learning has received significant hype and this is due to state-of-the-art performance in applications such as computer vision and machine translation. More recently, what has been bought up as an area still in need of establishment is the application of sequence processing neural network architectures to time series forecasting. This thesis demonstrates how predictions for several thousand incidents of social unrest activity in South Africa can be generated many days in advance, using news media event data.

This section aims to provide background and context to the problem that this thesis aims to solve (i.e. predicting social unrest events in South Africa using Long Short-Term Memory (LSTM) neural networks). This chapter has been divided into four sections:

- Social Unrest: a look at social unrest and the significant role it has played in shaping societies.
- Social Unrest in South Africa: a look at the social unrest climate in South Africa.
- The importance of predicting social unrest events: this section provides motivation as to why it is important to be able to predict these events before they occur. Also included is a high-level literature survey of publications in the domain of social unrest event forecasting, a summary of the shortcomings and how this thesis aims to contribute to the existing literature.

- High level approach: this section provides details on why this thesis hypothesizes that LSTM is the most suited machine learning technique to solve this problem.
- Summary of thesis objectives: this section summarises the problem statement and thesis objectives.

1.2 Social Unrest

Conflict and controversy are a normal part of social life (Jill, 2017). All societies experience a certain degree of social conflict, often in the form of class, ethnic, gender, religious, or other cleavages (Idean et al., 2012). Social conflict can manifest as social unrest events like protests, riots, armed conflict and civil war. Social unrest events have led to significant societal changes throughout history (Dan, 2012). The French Revolution and more recently the Arab spring are examples of social unrest events that marked key turning points in the history of the world. The French Revolution was a series of military conflict that lasted from 1792 to1802. The event overthrew the French monarchy and inspired liberal and radical ideals. The Arab spring began as waves of protests across several Arab countries against repression, corruption, youth unemployment and the rising cost of living. The protests soon evolved into civil unrest that led to the overthrowing of governments in Tunisia, Egypt, Libya and Yemen on the one hand and devastating civil wars in Syria, Libya and Yemen on the other. These events commenced in 2011 and their impact remains significant today.

Not all social unrest events manifest into armed conflict against government authorities and civil wars, others can take the form of violent riots. For instance, China saw a fifty percent rise in “public order disturbances” between the years 2003 and 2005. In December 2005, two to thirty villagers in China were reportedly killed. The event symbolized the depth of anger that those with grievances had and the inability of the Chinese administration to resolve disputes peacefully (Zhang and Fung, 2005). The National Patriotic Front in Liberia also

used violent riots to campaign against the state (Idean et al., 2012). Other social unrest events can manifest as peaceful demonstrations; For example, the Green Belt Movement in Kenya was a peaceful demonstration to contest the destruction of forests (Taylor, 2013).

In summary, social unrest can manifest in different ways and range in its nature from non-violent demonstrations, violent riots, armed conflict against government authorities, civil wars to revolutions. Social unrest can bring about major changes in political regimes and societies. It is therefore an important topic.

In this thesis, the country of focus is South Africa. The focus is on social unrest events that South Africa has experienced with the aim of establishing a model that can predict these events before they occur. This thesis successfully achieves the objective of predicting social unrest events. The outputs (count of social unrest events) of the model can be used by the government as an early warning sign that social unrest is impending. The definition of social unrest in this thesis includes peaceful demonstrations, reports of and acts of violence.

1.3 Social Unrest in South Africa

Social unrest in South Africa is not a recent phenomenon (Atkinson, 2007). During Aparthied (1948 – 1994), protests were mainly led by various organisations such as the South African National Civics Organisation (SANCO), the United Democratic Movement (UDF) and the African National Congress (ANC) (Narsiah and Maharaj, 2002). The protests were in favour of a democratic dispensation.

In 1994, the ANC was elected as the new democratic government; committing itself to the reconstruction of civil society and the achievement of more balanced policies. However, with unemployment, inequality, ecological degradation and persistent xenophobia on the rise, citizens are beginning to lose faith in the fulfilment of these promises and ask the question “what difference has democracy made

to development?" (Gumede, 2009). More specifically, citizens ask whether democracy has helped address the severe social and economic inequalities that once characterized the nation. The frustrations of citizens are being expressed in the form of protest action against the state. Figure 1.1 depicts the number of social unrest events in South Africa from 2004 to 2017 that received media coverage.

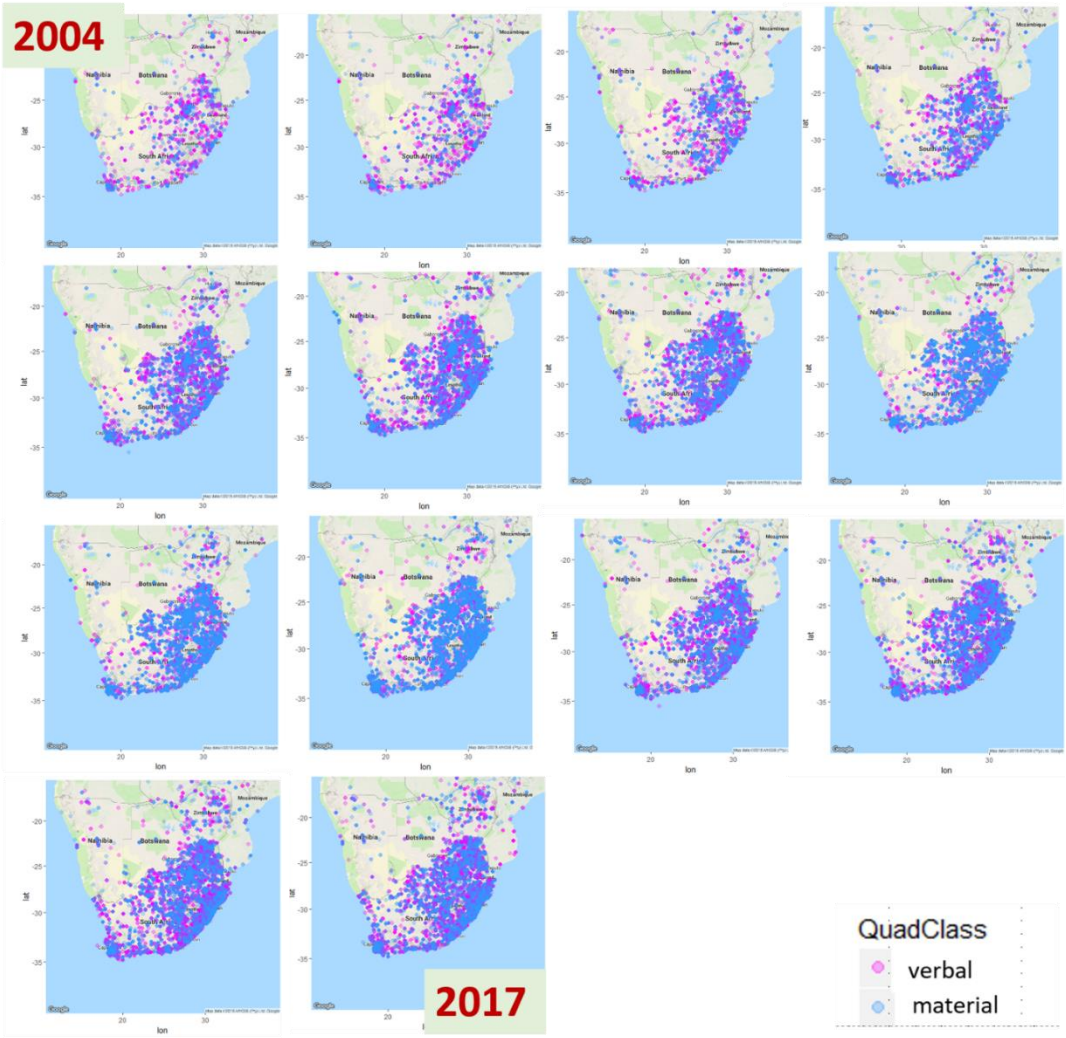

Figure 1.1: From left to right unrest events in South Africa 2004-2017.

In Figure 1.1 the blue dots indicate events reported as violent, resulting in the damage of property and in some cases, individuals. The pink dots highlight peaceful social unrest activities. Clearly, violent protest activity has exceeded the more peaceful acts over the years. Figure 1.1 reports an average of more than 8000 social unrest incidents per year.

This rate is one of the highest in the world, per person (Patrick Bond and Mottiar, 2012).

Peter Alexander (2010) describes protests in South Africa since 2004 as a “rebellion of the poor”. Trevor Ngwane (2010) characterized these new waves of protest as “a diverse set of organizations with the objective of organizing and mobilizing marginalized people to raise state awareness on failed policies and implementation”. In a more recent paper, Chigwata (2016) analysed protest events in South Africa occurring between 2007 and 2016. The study discovered that more than 90% of social unrest events during this time were associated with violence and intimidation. Other social unrest events that made headlines in South Africa and began around 2015, were the "fall-ism" protests. The three key events that dominated this space were: the student protests “fees must fall” and “Rhodes must fall” in 2015 and 2016, thereafter “Zuma must fall” in 2017.

1.4 The importance of predicting social unrest events

High social unrest activity in South Africa suggests a significant amount of social discontent. Social unrest negatively affects economic growth; driving away foreign and domestic investment, reducing social cohesion, paving the way for more populist policies and isolation from the international community (Abdul and Mansur, 2017).

Anticipating social unrest activities before they occur has benefits to both the public and government. The ability to forecast violent strikes can provide the government with the ability to formulate more effective mitigation plans, mobilize resources, coordinate responses and prioritize citizen grievances (Qiao, 2017). The public can be made aware beforehand on the areas to avoid. Being able to anticipate these events can assist decision makers in making more effective public policies.

Decision makers struggle to anticipate the evolution of these events and to initiate effective preparation and mitigation efforts. These difficulties motivate research into predicting these activities. The literature around social unrest in South Africa consists mainly of contributions from the political science field. These studies focus on identifying why protest activity is on the rise. Publications that focus on predicting social unrest events in South Africa have not been identified in the literature (discussed in Chapter 2). Given the social unrest climate in South Africa, the nation would significantly benefit from the improved capability to predict social unrest events before they occur and subsequently, be warned beforehand.

1.5 High level approach

Majority of the studies in the event forecasting domain of social unrest have used social media data, with Twitter being the most popular platform. The most commonly used machine learning techniques were found to be keyword filtering, clustering and logistic regression. Such studies among others include Hua et al. (2013), Ramakrishnan et al. (2014), Compton et al. (2014), Korolov (2016) and more recently Congyu et al. (2018).

Using social media data presents technical challenges. Social media platforms are dynamic, with users contributing to information every second. Identifying the relevant information, finding users and predicting events using keyword-based searches can become impractical and computationally expensive. For this reason, interest has increased in the use of machine coded data sets. One such data set is the Global Database of Events Language and Tone (GDELT). This database has recently become widely accessible and has attracted the attention of social and data science researchers interested in event forecasting.

Yonamine (2013) was the first to "use open-source, machine-coded event data to build forecasts of political violence at a sub-state level of geospatial aggregation" (Yonamine, 2015). The research of Yonamine was followed by that of Qiao et al. (2017) and Smith et al. (2017). Smith et al. (2017) proposed the use of multivariate LSTMs on GDELT data. Although the study only explored the predictive power of numerical indicators from GDELT, it reported superior results to traditional time series forecasting methods such as the autoregressive integrated moving average (ARIMA) models.

This thesis uses GDELT data and expands on the work of Smith et al. (2017). GDELT contains more variables and information than that used by Smith et al. (2017). Furthermore, Smith et al. (2017) neglected the cascading effect that other events may have on social unrest events. The objectives of this study therefore are as follows:

1. Predicting social unrest events in South Africa using LSTM.
2. Explore the predictive power of the other features present in GDELT data.
3. Explore the performance of other LSTM architecture and configurations.
4. To further establish LSTMs as a competing approach in the domain of time series forecasting.

The features of LSTM neural networks that make them preferable to traditional time series models such as ARIMA include:

1. Requires less formal statistical training as they are data driven and self-adaptive in nature. There is no need to specify model form or make any prior assumptions i.e. normality and stationarity assumptions made by traditional methods.

2. More practical for modelling more complex patterns. Neural networks have greater capability to detect non-linear relationships

between dependent and independent variables. With traditional models, the current values predictions are assumed to be a linear combination of historical values and past errors. Not all time series will meet this assumption.

3. There is no need for an equidistant time series. With traditional time series analysis, the distance between observations needs to be constant in order for the algorithm to be executable. This is not a limitation for training an LSTM neural network as they predict the next sequence as opposed to predicting a specified time horizon.

These capabilities suggest that LSTM neural networks are a promising competitive approach in the domain of time series forecasting. More specifically for practical situations where little to no theoretical guidance is provided about the problem.

1.6 Summary of thesis objectives

The social unrest climate in South Africa has been on the rise, highlighting the high levels of discontent in the country. This evidence suggests that the country will benefit significantly if these events can be anticipated in advance and provide guidance to decision makers. This thesis hypothesizes that the LSTM neural networks are the most suited machine learning technique to solve this problem given their primary capability to detect complex patterns and process data in sequence effectively. The aim is to therefore demonstrate the promise of LSTM neural networks as a real time scalable approach to forecasting social unrest events in South Africa. A forecast of the count of social unrest events will be conducted using historical event data provided by GDELT.

In order to fully assess the capabilities of LSTMs, a baseline approach to compare with needs to be established. ARIMA is selected as the

baseline approach for comparing the predictive power. This thesis hypothesizes that LSTM will be a superior method. Though Smith et al. (2017) conducted a similar comparative study, the study was limited to multivariate LSTM architecture and used limited features. This thesis will assess the performance of univariate LSTM and multivariate LSTM against traditional approaches and expand on the feature space used.

Chapter Two

Literature Review

Event forecasting has been used in a vast number of applications such as: forecasting floods (Damle, 2005), predicting disease outbreaks (Ford, 2009), box office sales (Asur, 2010), elections (Tumasjan, 2010), stock market movements (Zhan, 2011) and crime (Wang, 2017). Appendix A is a summary of the literature on event forecasting. The summary shows that there is a significant amount of literature on "detecting and predicting real-world events" (Singh, 2018). For this study, the literature review will be restricted to event forecasting in the domain of social unrest events, in which machine learning approaches are applied.

To facilitate the review, the relevant papers have been categorised into three groups based on data types: conflict databases, social media data, and machine-coded data sets.

2.1 Conflict Databases

Conflict databases only report on armed conflict events. Armed conflicts are sparse events in most geographies. Therefore, much of this work is focused on making predictions that are well into the future using macro-level indicators. For example, Ward et al. (2010) used Uppsala Conflict Data Program (UCDP) data and logistic regression to predict conflict events across various geographies. Predictions were made at a monthly level for six-month periods into the future. The model produced quality results only for countries that reported high civil unrest activity. Huang et al. (2015) used SSPNet Conflict Corpus and compared the performance of multiple machine learning methods,

and this included: an ensemble method of random k-nearest neighbours, support vector regression, and a simple logistic regression model to predict conflict events in Switzerland five years into the future. Huang et al. (2015)'s study established that the use of more complex machine learning methods produced better results when working with sparse event data. Complementing the findings of Huang et al. (2015) is the works of Muchlinski et al. (2015). Muchlinski et al. (2015) used Civil War Data (CWD) and compared the performance of a random forest with logistic regression in predicting civil war in Africa and the Middle East. In Muchlinski et al. (2015), a random forest produced more accurate results compared to logistic regression. Weidman and Ward (2010), experimented further with geolocation variables to improve the forecast accuracy of sparse events. Weidman and Ward (2010) used the Armed Conflict Location and Events Data set (ACLED) and the autoregressive discrete regression to predict armed conflict events in Bosnia. The study showed that including special data improved the forecast accuracy.

The works of Ward et al. (2010), Weidman and Ward (2010), Huang et al. (2015), and Muchlinski et al. (2015) establish a foundation for social unrest prediction. However, the focus is on violent events like civil wars which are very sparse with no recording of simple demonstrations and peaceful protests that can provide valuable information. There is significant literature in the domain of social unrest forecasting towards making real-time predictions for more frequent occurring events, such as simple demonstrations and peaceful protest. An interesting observation (see Appendix A Figure A3) is a decrease in the use of conflict databases and a rise in the use of social media data.

2.2 Social Media Data

Social media platforms such as Twitter, Facebook, and Tumblr provide users with the ability to create and share information across the world

with ease instantly. Twitter alone has on average over 126 million daily users publishing information (Kastrenakes, 2019). Therefore, due to high reach, popularity, speed, and the ease of publication, social media platforms have played an instrumental role in the organisation and announcement of protests, strikes, and other demonstrations. Howard et al. (2011) was the first to demonstrate the role social media played in shaping the political debate of the Arab spring (a series of protests in the middle east). The findings in Howard et al. (2011) grabbed the attention of many researchers, who proceeded to start investigating the predictive power of social media data in the domain of social unrest prediction.

The literature in the domain of social unrest prediction using social media data has two focus areas: event detection and event prediction. Event detection focuses on mining data from social media by either using keyword extraction methods as in the study of Compton et al. (2013) or using more advanced Natural Language Processing (NLP) methods and clustering techniques as in the study of Korolov et al. (2016). Compton et al. (2013)'s keyword extraction method places a strong reliance on constructed vocabulary, whereas Korolov et al. (2016)'s topic modelling approach relaxes this dependency. Results from event detection can only discover events after they have occurred. Event prediction focuses on applying machine learning techniques that use historical event patterns of the data to predict the likelihood of an event occurring in the future.

Compton et al. (2014) used logistic regression classifiers on data containing the mention of unrest and future dates to predict the likelihood of a social unrest event occurring in various Latin American countries. To validate the results, Compton et al. (2014) used news media reports as the ground truth in which out of the 2859 detected events, only 583 aligned to the news articles. Compton et al. (2014)'s study established that massive protests were much easier to predict using social media data as they generate substantial signals on

Twitter. Compton et al. (2014) further expresses significant computational challenges when working with millions of unprocessed text data. Korolov et al. (2016) applied the same forecasting and validation approach as Compton et al. (2014) on twitter data however, the study produced significantly stronger correlations to the ground truth data. Korolov's work focused on predicting the 2015 Baltimore social unrest events. The study was location and event specific. Korolov et al. (2016) used a smaller sample than Compton and found it easier to detect and extract the relevant events from social media data.

The works of Cadena et al. (2015) and Goode et al. (2015) focused on predicting large scale social unrest events at multiple locations using the notion of activity cascades derived from Twitter. This approach mimics abnormality detection. Cadena et al. (2015) and Goode et al. (2015) first use NLP methods to identify large activity cascades. A large cascade was associated with an event. A logistic regression classifier was trained on the tweets to identify large activity cascades. The studies reported good model accuracy and established a good quality machine learning approach for detecting large-scale protests.

What can be gathered from the works of Compton et al. (2014), Korolov et al. (2016), Cadena et al. (2015) and Goode et al. (2015) is that methodologies that are focused on specific events and location with significant post volumes are more likely to produce quality forecasts. Another approach to improve forecast accuracy is in the combination of data from different social media platforms.

Korkmaz et al. (2013) used a combination of data from social media platforms (Twitter and blogs) and news sources to predict social unrest in six Latin American countries. Korkmaz et al. (2013) used logistic regression models with Lasso. They showed that "combining different data sources is effective in predicting next-day social unrest as measured by the F1-score (a measure of the quality of the forecast that balances precision and recall)". The F1-score ranged from 0.68 to 0.95.

Muthiah et al. (2014) used Twitter and Facebook data to predict social unrest events in various Latin American countries using an automatic protest forecasting system that he developed called “EMBERS (Early Model Based Event Recognition using Surrogates)”. The study reported better prediction accuracy. Similarly, Compton et al. (2014) used Twitter and Tumblr data together and reported better results.

Collectively, the studies using social media data to predict social unrest events reported significant challenges in dealing with social media data. Data mined from social media platforms is raw text and this data format tends to be very noisy. Therefore, standard NLP techniques may not produce the desired results and additional human validation steps are required to check the data. For example, Compton et al. (2014) developed the logistic regression classifier with the help of “Amazon Mechanical Turk” services. “Amazon Mechanical Turk” allows individuals to crowd-source workforce that can manually annotate posts to indicate if they relate to social unrest or not. This manual step is expensive in terms of both time and resources. Further, social media data is big data that takes many forms such as videos, images, and text and can therefore take a large amount of time to collect and process. These data formats may require high-performance computing to perform the necessary data transformations and machine learning tasks efficiently. Setting up the infrastructure to process high volumes of data can be costly.

The limitations of using social media data have given the need to explore other data sets, such as machine coded data sets.

2.3 Machine coded data sets

The increase of academic interest in political violence has led to the need to explore more efficient ways to gather event data. Event data is one that lists individual acts or interactions along with the geospatial

coordinates. Examples of event data date back a few decades and include the World Event Interaction Survey (WEIS) (McClelland, 1976), the Conflict of Peace Data Bank (COPDAB) (Azar, 1980) the Armed Conflict Location and Event Dataset (ACLED) (Raleigh et al., 2010), the Geo-referenced Event Data Set (GED), the Social Conflict in Africa Database (SCAD) (Salehyan et al., 2012). These event data sets are similar to, conflict database and social media data in that they rely largely on human coding of news reports, and thus require significant effort and time to process.

Schrodt et al. (1994), developed machine-assisted approaches to generate the same kind of data from news sources. These machine-assisted approaches have been used in the creation of large international data sets such as Kansas Event Data System (KEDS) (Gerner et al., 1994), and more recently GDELT (Leetaru and Schrodt, 2013).

GDELT is the most popular machine coded data set identified in the literature (see Appendix A Figure A3). GDELT is an open source event data repository based on news articles and currently contains over a quarter-billion of event records starting from the year 1979. The database is updated daily through automated coding and includes over 300 types of events (Leetaru and Schrodt, 2013). The GDELT project applies automated coding techniques to generate event data from news media reports and apply automated geocoding algorithms. This results in a data product that relies on the same sources as the human-coded datasets (local and international sources from newspapers and newswires, usually provided through repositories such as LexisNexis and Factiva), but whose creation is fully automated. The machine-coding approach uses keyword search algorithms to define lists of actors and events. News reports are fed into a parser that identifies events based on the actor and event keywords given in the dictionaries (Gerner et al., 2002).

Several advantages that machine coded data sets offer in comparison to human coded ones include: the ability to process a large amount of reports in little time; the data is made available in a database therefore it is ready-to-use for analysis and has eliminated the need for researchers to deal with language complexity. Some concerns of using machine coded datasets include: automation is never perfect, the automated coding process may fail to pick up event, or code others that the human coder chose not to include; automated geo-referencing is complex and can report mentions of different locations, and it may be difficult to determine which locational information in a report accurately describes the event's location.

Earlier work has analyzed the validity of machine coding. Best et al. (2013), performed validity checks that indicated that machine-coded data can match human-coded data with a high degree of accuracy. Hammond and Weidmann (2014) conducted an analysis that compares GDELT to two other well established human coded data sets, the ACLED (Raleigh et al., 2010) and GED (Sundberg and Melander, 2013). Hammond and Weidmann's study reported modest correlation between GDELT and two other data sets ACLED and GED. Further their analysis revealed that GDELT seems to track temporal ups and downs in violence as identified by the human-coded datasets accurately at a national level.

Researchers who have motivated the use of GDELT in social unrest prediction include: Arva (2013), Yonamie (2013), Korkmaz (2015), Qiao (2017), and Smith et al. (2017). These studies demonstrated the predictive power of GDELT, and the studies were all performed at a national level.

Arva (2013) compares the forecasting utility between GDELT and ICEWS (Integrated Conflict Early Warning System, another machine coded data set) by feeding a time series to the following machine learning techniques: random forest, ADABoost and Bayesian model

averaging. The results of the analysis revealed that GDELT produced "dramatically lower false negative rates" and therefore made it more suited for statistical forecasting purposes. Arva (2013) was the first to formalize social unrest event prediction as a sequence time series problem.

Yonamine (2013) used GDELT data and the autoregressive fractionally integrated moving average (ARFIMA) model to predict material conflict in Afghanistan. ARFIMA is a variation of the popular time series forecasting method, ARIMA. Yonamine (2013) compared the performance of ARFIMA to that of a naïve model. A naïve model is an estimation method in which the last period's actual values is used as the current period's forecast. The study focused on motivating the use of GDELT to predict political violence at specific locations. The ARFIMA approach outperformed the naïve model. The study of Yonamine (2013) was the first to apply ARIMA methods to social unrest event forecasting and demonstrated useful results. Qiao et al. (2017) built on Yonamine's study, further motivating the use of GDELT in statistical forecasting. Qiao et al. (2017) "developed a Hidden Markov Model (HMM) for predicting social unrest in five separate south-east Asian countries" using GDELT. The study was "formalized as a sequence classification problem". Qiao's study concluded that "the GDELT data set does reflect some useful precursor indicators that reveal the causes or development of future events". Smith et al. (2017) compared the performance of Yonamine's ARFIMA model to that of a neural network-based model known as LSTM in predicting social unrest events in Afghanistan using GDELT data. Smith et al. showed that the LSTM model produced lower error rates than the ARFIMA model. Smith et al. (2017) however, did not explore the impact of prediction performance of all the data features provided by GDELT and other possible LSTM architectures.

This thesis expands on Smith's approach by incorporating additional features provided by GDELT; comparing univariate and multivariate

performance of ARIMA and LSTM. Further, the predictions are made in the context of South Africa; a geography that has not yet been explored in literature.

Very little has been published on using LSTMs to predict social unrest. However, there is a sizable number of studies that use LSTMs for time series event forecasting outside the domain of social unrest. These include: Duan et al. (2016) for travel time prediction, Chen et al. (2015), Nelson et al. (2017) and Qin et al. (2017) use LSTM and LSTMs with attention layer for stock market prediction, while Zhu and Laptev (2017) use LSTMs to predict the number of trips Uber taxis (a large transport company) made during special events. LSTMs were used in these studies as they are more powerful at detecting complex patterns and spikes in data when compared to traditional statistical learning approaches. For this reason, LSTMs have been selected as the machine learning technique of choice in this thesis.

Chapter Three

Theory of Models

This chapter looks at the theoretical foundation of ARIMA and neural networks used to solve the social unrest prediction problem.

3.1 Time Series Forecasting With ARIMA

When it comes to time series forecasting, ARIMA models are the most frequently used in the literature. ARIMA models have been selected as the baseline approach for the following reasons:

1. Simple to understand and their methodology is well established in the literature of time series forecasting.

2. Ease of implementation. Automated R packages that perform the necessary parameter optimization and data transformations are well established and exist.

3. They make an excellent comparative approach to neural networks given the specific contrast in model capabilities that were discussed in Section 1.

3.1.1 Autoregressive and Moving Average Models

The ARIMA models are made up of three components: an autoregressive component AR(p), a moving average component MA(q) and a differencing component d. In the AR(p) component, the current value of a time series is assumed to be "a linear combination of p past observations and a random error (sometimes called white noise)

together with a constant term" (Adhikari and Agrawal, 2013). Mathematically the AR(p) model can be expressed as:

$$y_t = c + \sum_{i=1}^{p} \varphi_i y_{t-i} + \epsilon_t \quad (1)$$

where:

y_t is the actual value of Y the dependent variable at time t;

ϵ_t is the random error (also known as random shock) at time period t;

$y_{t-i}(i = 1,2,\ldots,p)$ is the realization of the lag dependent variables;

$\varphi_i(i = 1,2,\ldots,p)$ are the unknown model parameters;

c is a constant; and

p is the number of lagged values of Y and represents the order of the AR process.

Figure 3.1 below is an example of a simulated AR process of order one where φ is 0 and 1. The AR process with a high φ value produces a time series with less white noise giving the time series structure. This means the prediction of the current value is strongly dependent on its previous values. If φ is 0, then the time series only constitutes of white noise. "If a time series is white noise, it is a sequence of random numbers, and it cannot be predicted well" (Jason, 2017).

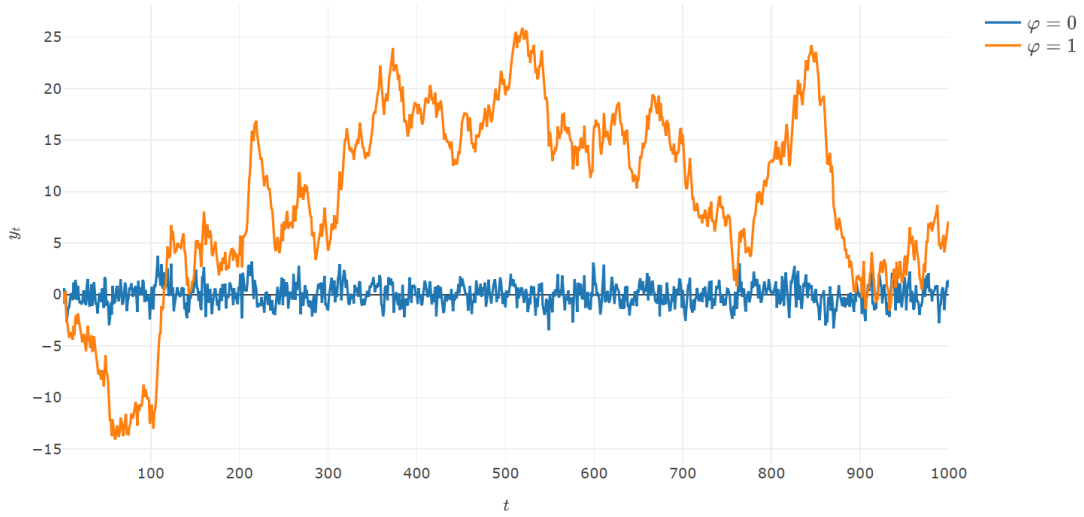

Figure 3.1: Autoregressive process for φ = 0 and φ = 1.

The second component of the ARIMA(p,d,q) model is the moving average MA(q) process. A moving average represents "a process where the current value of a time series is described as a function of q past errors" (Adhikari and Agrawal, 2013). Mathematically the MA(q) model can be expressed as:

$$y_t = \mu + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_t \qquad (2)$$

where:

y_t is the actual value of Y the dependent variable at time t;

ϵ_t is the random error (also known as random shock) at time period t;

μ is the mean of the series;

$\theta_j (j = 1,2,\ldots,q)$ are the unknown model parameters; and

q is the number of lagged values of Y and represents the order of the MA process.

Figure 3.2 is a plot of an MA process of order two. It is rather difficult to view the characteristics of the MA(q) process even when varying θ_j as the data appears to have no structure other than the presence of a constant mean. The characteristics of MA(q) process can be better described using autocorrelation function (ACF) plots. These will be discussed in the sections to follow.

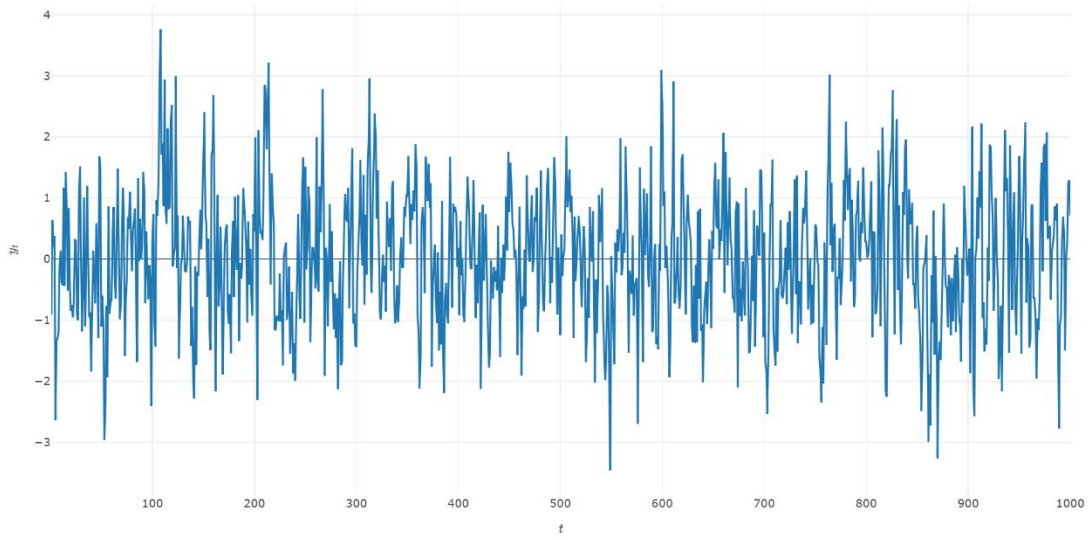

Figure 3.2: The moving average process.

A model that involves both AR(p) and MA(q) is denoted as ARMA(p, q). ARMA represents "the current value of a time series as a linear function of its past values and previous residual values (the past errors)" (Adhikari and Agrawal, 2013). Mathematically the ARMA(p, q) model can be expressed as:

$$y_t = c + \epsilon_t + \sum_{i=1}^{p} \varphi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j}. \qquad (3)$$

Figure 3.3 is a simulated data set of the ARMA(2,1) process where: $\varphi_1 = 0.7$, $\varphi_2 = 0.2$ and $\theta = 0.4$. Here, the time series appears to have more structure, and the values are fairly centered around the mean.

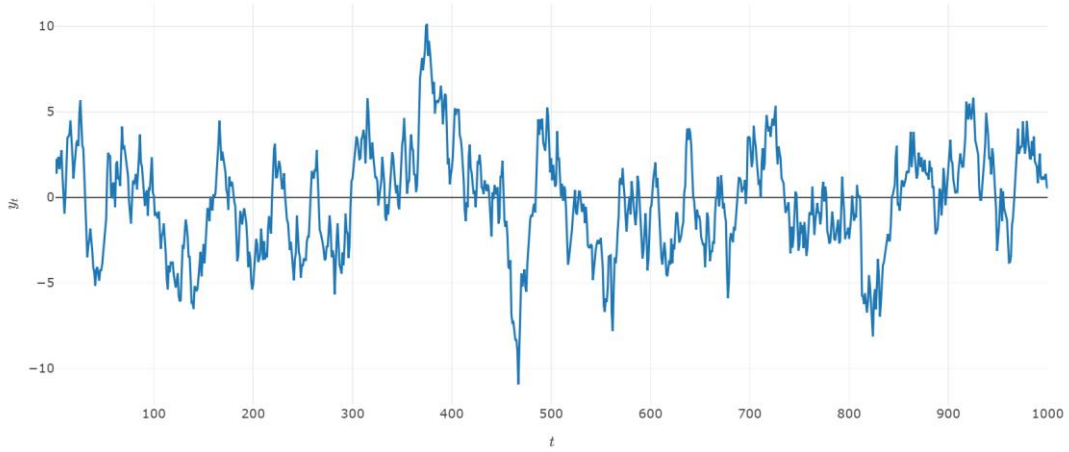

Figure 3.3: The autoregressive moving average process for φ = 0.7, φ_2 = 0.2 and θ = 0.4.

3.1.2 Stationarity

A key assumption that is made by AR, MA, and ARMA models is stationarity. If a time series y_t is stationary, then for all time periods k, the distribution of $(y_t, \ldots, y_{t+k})$ does not depend on t. Figure 3.4 is an example of a non-stationary time series because of the increasing trend. A non-stationary time series y_t can be made stationary through differencing, resulting in a new series x_t given by $x_t = y_t - y_{t-1}$ for all $t \epsilon N$ where $N = \{1,2, \ldots, n\}$ is the sample size of the data set. Differencing may have to be applied more than once to make the time series stationary.

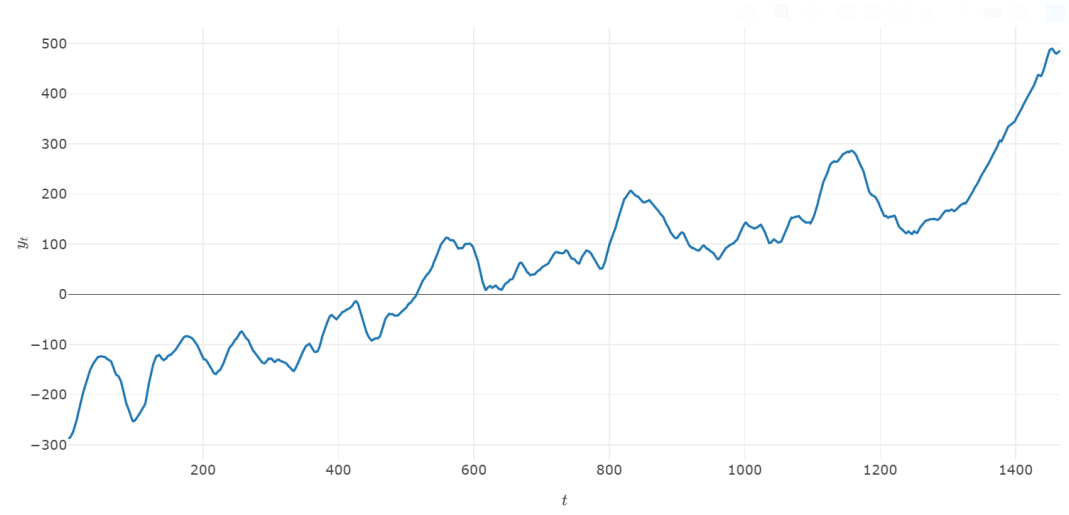

Figure 3.4: A non-stationary time series.

Assuming that the ARMA series is stationary, the model can be simplified by introducing the Box-Jenkins backshift operator L, defined by the following relationship: $L^k y_t = y_{t-k}$, such that $y_1, y_2, \ldots, y_t$ is any time series and $k < t$. Applying the backshift operator to Equation 3, when c is assumed equal to zero, yields the following:

$$\left(1 - \sum_{i=1}^{p} \varphi_i L^i\right) y_t = \left(1 - \sum_{j=1}^{q} \theta_j L^j\right) \epsilon_t$$

which is often written as:

$$\varphi(L) y_t = \theta(L) \epsilon_t \tag{4}$$

where:

$$\varphi(L) = 1 - \sum_{i=1}^{p} \varphi_i L^i$$

$$\theta(L) = 1 + \sum_{j=1}^{q} \theta_j L^j .$$

If a time series is not stationary, then the ARMA model in Equation 4 can be modified and written in terms of differences. Suppose that w is the first difference of y. Then, for any t: $w_t = y_t - y_{t-1} = y_t - L y_t = (1 - L) y_t$. Thus, the differenced series w is obtained from the original series y by multiplying by a factor of $1 - L$. Now, if z is the first difference of w, that is, z is the second difference of y, then: $z_t = w_t - w_{t-1} = (1 - L) w_t = (1 - L)(1 - L) y_t = (1 - L)^2 y_t$. The second difference of y is therefore obtained by multiplying by a factor of $(1 - L)^2$, and in general the d^{th} difference of y would be obtained by multiplying y by a factor of $(1 - L)^d$. Thus, replacing y_t with $(1 - L)^d y_t$ in the ARMA model in Equation 4 yields the formal ARIMA(p, d, q) model given by:

$$\varphi(L)(1-L)^d y_t \;=\; \theta(L)\epsilon_t \qquad (5)$$

where: $(1-L)^d y_t$ is the new time series obtained by differentiating the initial series y_t, d times.

The ARIMA model uses historical data of a univariate time series to predict future series values. It is possible to use more than one time series as the input to the ARIMA model in order to predict the value of another series. When an ARIMA model includes other time series as input variables, the model is known as the ARIMAX model. The following is an example of the ARIMAX model:

$$y_t \;=\; \mu + \sum_{i=1}^{k} \frac{\varphi_i(B)}{\theta_i(B)}\, B^{li}\, x^{it} \;+ \in_t \qquad (6)$$

where:

$\varphi_i(B)$ denotes the autoregressive coefficients' multinomial of the i^{th} input time series;

$\theta_i(B)$ denotes the moving average coefficients' multinomial of the i^{th} input time series;

li denotes the lag degree of the i^{th} input variables;

ϵ_t represents the regression residual time series;

x^{it} represents a covariate at time t of the i^{th} input variables;

B is the coefficient of an explanatory variable.

The detailed mathematical background of the ARIMAX model is discussed in Fan et al. (2009).

3.1.3 Identifying Time Series Properties

ARIMA methods require that the features or pattern of the time series be formally identified so that the appropriate model order (p, d, q) is used. Features include: trend, evidence of seasonality, white noise,

relationships between multiple series, relationships between lags, etc. These features can be identified using plots and methods such as:

1. Decomposition plots: Time series decomposition involves splitting the time series into components, i.e., trend, cycles, seasonal and white noise. Each component exhibits a pattern, and this can be used to understand the nature of the time series and help determine which forecasting method to apply.

2. Scatter plots: Used to explore relationships between multiple time series. The effects of other time series can be incorporated into the forecasting model to help better explain the random variations (i.e., white noise).

Furthermore, correlation coefficients can be used to measure the strength of the linear relationship between values of a time series. The correlation plots include:

1. Autocorrelation plot: Autocorrelation measures the linear relationship between lagged values of a time series.
2. Partial autocorrelation plots: This is an extension of autocorrelation that considers the dependency structure among the intermediate variables.

The correlation plots and their definitions are discussed in more detail in Sections 3.1.5 and 3.1.6.

3.1.4 Time Series Decomposition

Time series decomposition components are defined as follows:

1. Trend: The long-term movement of a time series, either growth or decline. Not all series will have a trend component.

2. Cycles: These are medium to long-term changes in a series. An example of a cycle is “a business cycle that consists of four phases: prosperity, decline, depress, and recovery”.

3. Seasonal: Defined as fluctuations within a year. An example of a factor that causes seasonal variation is the climate, which impacts ice-cream sales in summer vs winter.

4. White noise: Irregular fluctuations in the time series.

Time series decomposition is commonly represented as an analysis step prior to generating predictions. However, it can also be used as a forecasting method. To forecast a time series using a decomposition model and assuming an additive decomposition, the mathematical expression is as follows:

$$y_t = T_t + C_t + S_t + R_t \quad (7)$$

where:

y_t is the actual value of Y the dependent variable at time t;

T_t is the trend component;

C_t is the cycle component;

S_t is the seasonal component;

R_t is the random variation component.

Assuming a multiplicative decomposition, the mathematical expression is as follows:

$$y_t = T_t \times C_t \times S_t \times R_t \quad (8)$$

The multiplicative models assume that the components are not independent, whereas the additive assumes the components are independent.

A time series decomposition approach to forecasting does not allow for a flexible specification of the seasonal component (i.e., the method is not flexible and does not consider possible variations in the seasonal pattern). ARIMA and ARIMAX extend on the decomposition approach. In this thesis, decomposition is used as an exploratory method to get an understanding of the social unrest time series components. An additive model with independent T, C, S and R is assumed.

3.1.5 Autocorrelation Function

Autocorrelation measures "the strength of the linear relationship between lagged values of a time series" (Froude, 2018). For example, let y_t "denote the value of a time series at time" t. The autocorrelation function "(ACF) of the series gives correlations between" y_t and y_{t-1} for $k = 1,2,3$ etc. Mathematically the autocorrelation between y_t and y_{t-k} is expressed as:

$$r_k = \frac{\sum_{t=k+1}^{T}(y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^{T}(y_t - \bar{y})^2} \quad (9)$$

where:

r_k is the "measure of the strength of the linear relationship" (Froude, 2018);

T is the length of the time series.

Figures 3.5 and 3.6 are examples of ACF plots for a time series generated by an AR process. Here, "the dashed blue line indicates a significance threshold. When lags are consistently outside of the pair of dashed blue lines, the trends are considered non-stationary" (Revels,

2017). In Figure 3.5 where φ is 0.4 after lag 4 all the lags consistently fall within the dash blue lines. The time series may not have to be made stationary. In Figure 3.6 when φ is 1, all lag values are above the dashed blue line indicating that the time series must be made stationary before ARIMA methods can be applied.

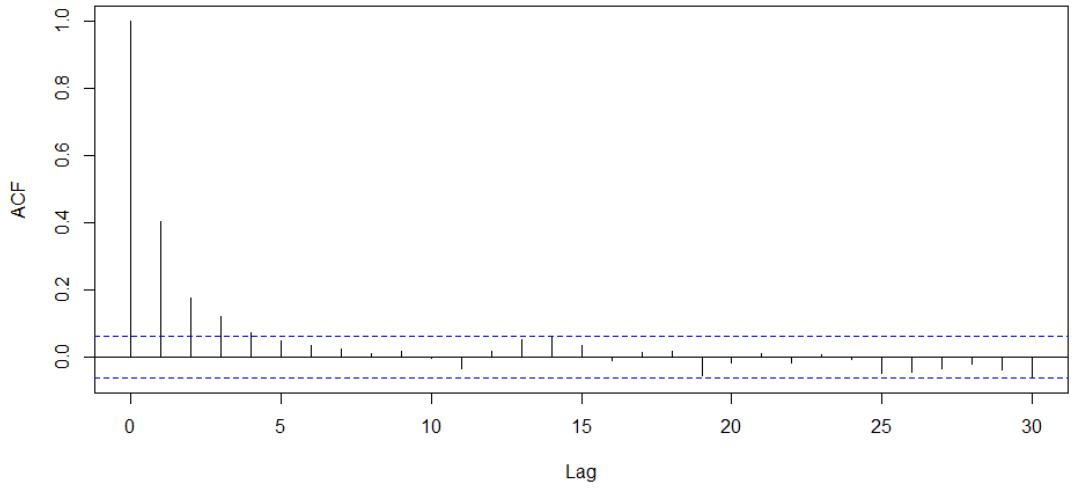

Figure 3.5: ACF for AR process for φ = 0.4.

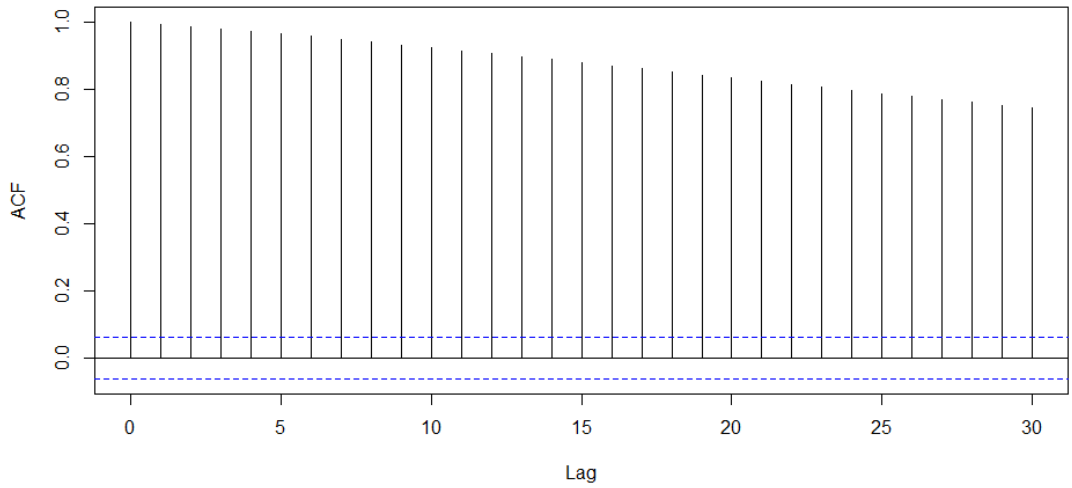

Figure 3.6: ACF for AR process for φ = 1.

Figures 3.7 and 3.8 compare the two AR models one of order 1 and the other of order 2 (i.e. AR(1) and AR(2) respectively). For the AR(2) model the autocorrelations diminish at a slower rate than the AR(1) model.

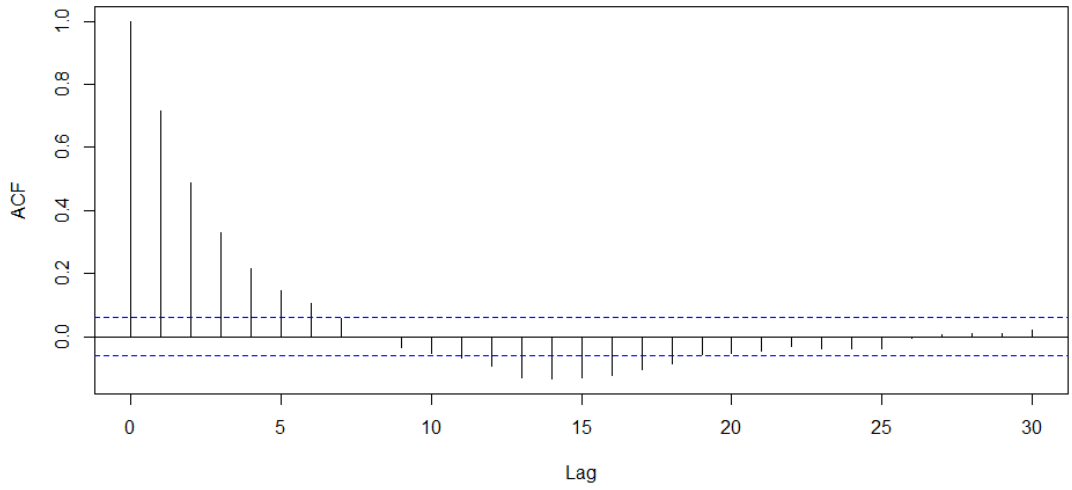

Figure 3.7: ACF for AR process of order one.

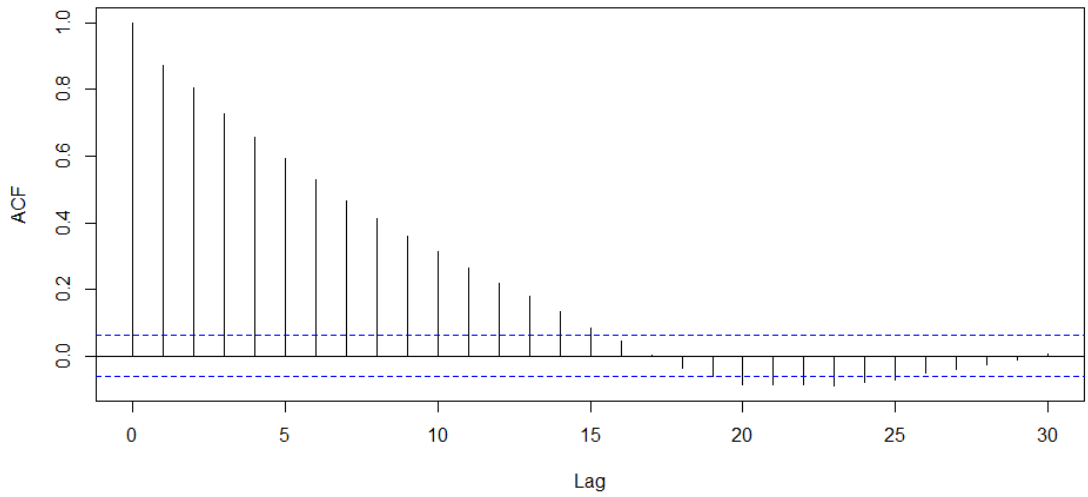

Figure 3.8: ACF for AR process of order two.

Figures 3.9 and 3.10 are examples of ACF plots for the MA process for MA(1) and MA(2). For MA(1), the autocorrelation cuts off at lag 1 and for MA(2), the autocorrelation cuts off at lag 2. This is the characteristics of the MA(q) process. The correlation between y_t and y_{t-n} when $n > q$ is always zero. In contrast the autocorrelation for the

AR(p) model persists even at values of $n > p$. This difference in the behavior of the two models with respect to ACF when $n > q$ and p is used when deciding whether to fit an AR or MA model to the data. Thus, when the ACF of a process suddenly becomes zero at a specific lag, the process is deemed to be MA.

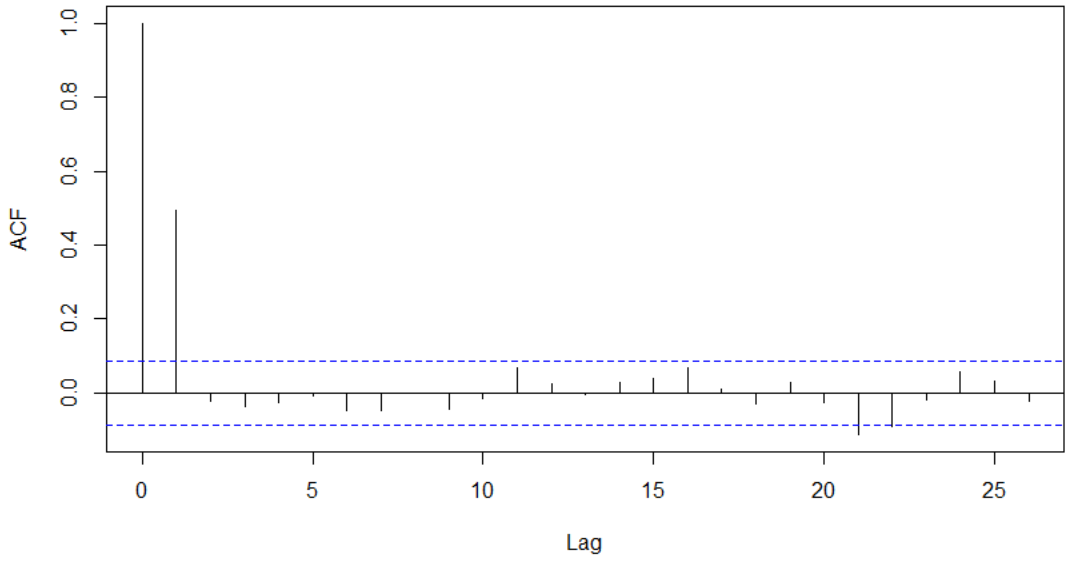

Figure 3.9: ACF for MA process of order one.

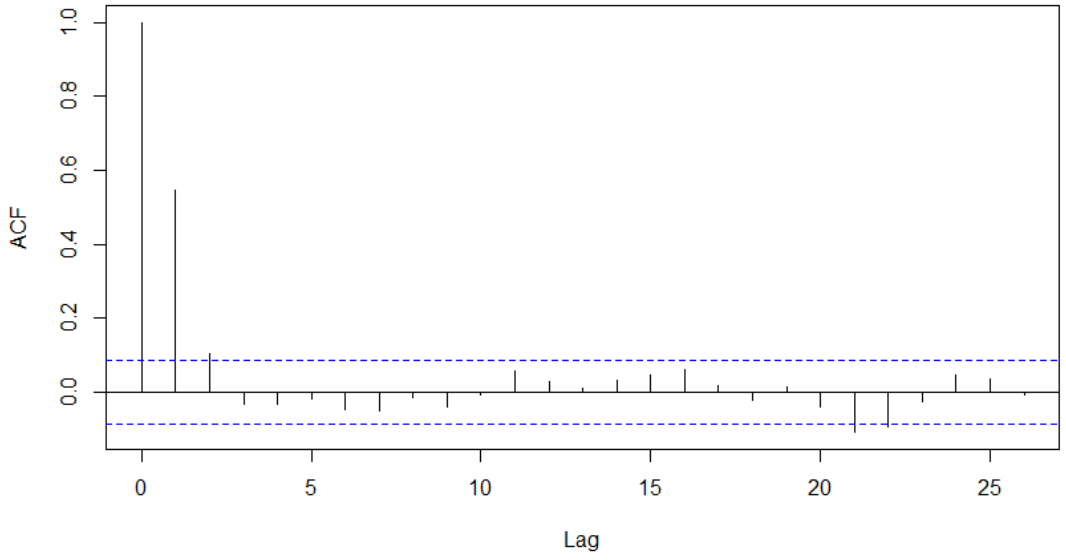

Figure 3.10: ACF for MA process of order two.

We have seen that it is easy to estimate the order of a MA process from the ACF. However, the same is not true for the order of the AR process. In order to estimate the p in AR(p), the PACF is used.

3.1.6 Partial Autocorrelation Function

PACF is another way to evaluate the relationship between y_t and its lagged values. The partial autocorrelation of y_t at lag k is the coefficient of y_{t-k} in a regression of y_t on $y_{t-1}, y_{t-2},\ldots$, up to y_{t-k}. Thus, the partial autocorrelation of y_t at lag 1 is the same as the autocorrelation of y_t at lag 1. The partial autocorrelation of y_t at lag 2 is the coefficient of y_{t-2} in a regression of y_t on y_{t-1} and y_{t-2}, and so on. The way to interpret the partial autocorrelation at lag k is that it is the amount of correlation between y and y_{t-k} that is not explained by lower-order autocorrelations.

Figures 3.11, 3.12 and 3.13 are the plots of AR series, ACF and the PACF for a stationary AR process of order 3 where: $\varphi_1 = 0.9, \varphi_2 = -0.6$ and $\varphi_3 = 0.3$. In the ACF plot, there is a significant correlation at lag 1. The PACF function shows significant autocorrelation at lags $1, 2$ and 3. An autoregressive process of order p has a PACF that cuts off after p lags.

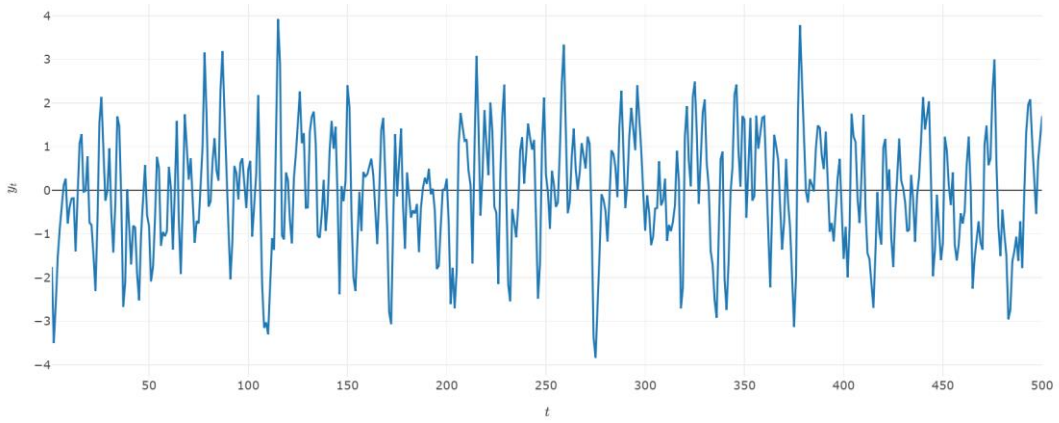

Figure 3.11: Time series AR process of order 3, $\varphi_1 = 0.9, \varphi_2 = -0.6$ and $\varphi_3 = 0.3$.

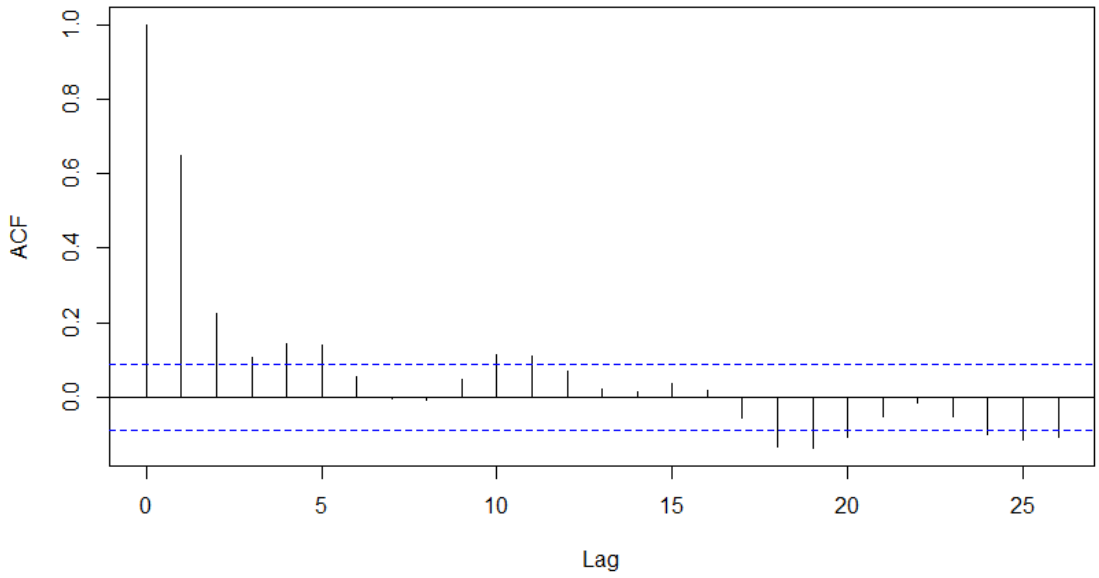

Figure 3.12: ACF for AR process of order 3, $\varphi_1 = 0.9, \varphi_2 = -0.6$ *and* $\varphi_3 = 0.3$.

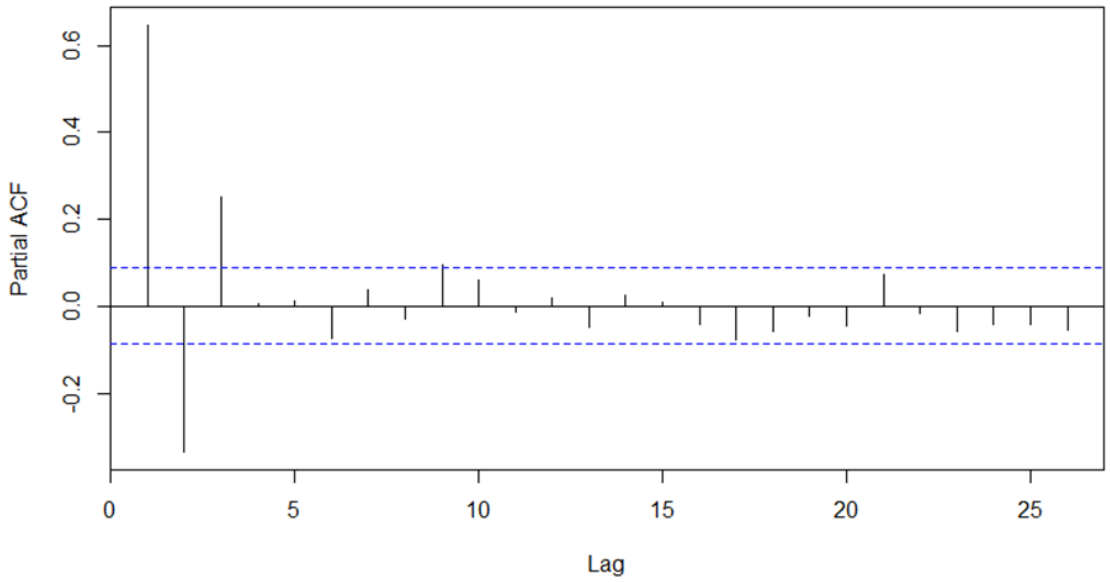

Figure 3.13: PCG for ARMA process of order 3 $\varphi_1 = 0.9, \varphi_2 = -0.6$ *and* $\varphi_3 = 0.3$.

3.2 ARIMA Process Steps

From what has been discussed above, the process of determining the order of an ARIMA model involves: describing the time series using decomposition and correlation plots, applying the necessary transformations to ensure stationarity (i.e., differencing), and thereafter identifying the order of MA and AR. These steps are summarized in Figure 3.14.

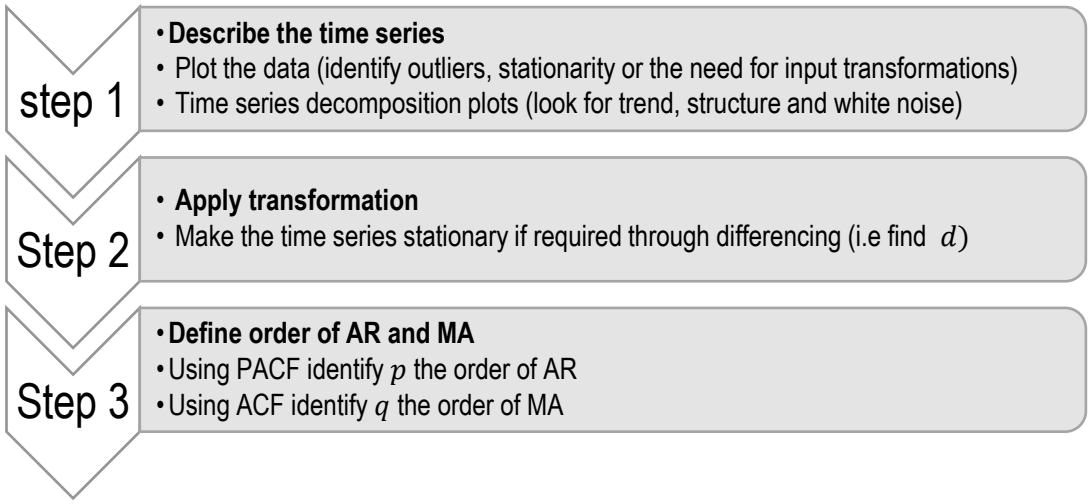

Figure 3.14: Process for determining the order of the ARIMA model.

Table 3.1 below summarizes the behaviour of AR(p), MA(q) and ARMA(p, q) with respect to ACF and PACF values. A pure AR process will trail off when looking at the ACF and cut off at lag p when looking at PACF. A pure MA(q) process cuts of after q when looking at ACF and trails off when looking at the PACF. The mixed process ARMA will trail off when looking at both functions.

Table 3.1: Guideline to identifying ARMA process.

	$\boldsymbol{AR(p)}$	$\boldsymbol{MA(q)}$	$\boldsymbol{ARMA(p,q)}$
ACF	Trails off	Cuts off after lag q	Trails off
PACF	Cuts off after lag p	Trails off	Trails off

When working with more complex models, identifying the order of ARIMA becomes less straightforward. This thesis uses the “auto.arima” function from the R library called “forecast”. This function automates the process of finding the optimal model parameters. The one limitation of this function is that as the time series becomes long, it becomes a computationally expensive process.

3.3 Time Series Forecasting with Neural Networks

ARIMA models have been proven to work well when the data has structure with a linear trend. ARIMA models, however, do not perform well when the trend and seasonal effects are changing through time or when irregular fluctuations exist in the data (Chatfield, 2003). Real-world problems can be complex and will usually have time series that do not fit the ARIMA model assumptions. More sophisticated modelling approaches are desirable in order to improve forecasts. Section 2.3 mentioned studies that established that neural networks outperform traditional methods when working with complex data on a variety of tasks, including forecasting. The next subsections provide a review of three types of neural networks used in forecasting tasks: feed forward, simple recurrent neural networks (RNNs) and the more complex RNN know as the LSTM.

3.3.1 Feed-forward neural networks for Time Series

Figure 3.15 is an example of a feed-forward neural network architecture. The nodes are “processing units called neurons” (Nami, 1997). Neurons are grouped together to form layers. Each layer is identified by the index $l = 0, \dots, L$. "The layers 0 and L are called the “input layer” and “output layer”, and all other layers are called “hidden layers”. Each neuron has an “activation” level a, which is given by the formula $a = f(Z)$. f is known as the "activation function.

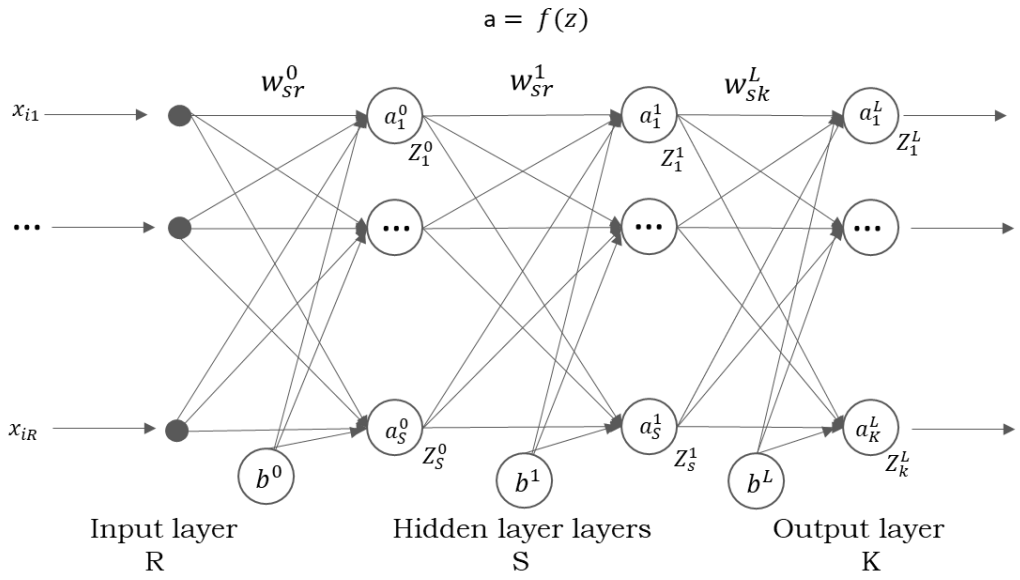

Figure 3.15: A feed-forward neural network with an input layer, hidden layers and an output layer.

An activation function modifies a pattern before outputting it. The most common activation functions include: Sigmoid Function, Rectified Linear Units (ReLu), and Hyperbolic Tangent Function (tanh) (Jason, 2019).

A sigmoid function is a non-linear mathematical function. It has continuous boundaries between zero and one (Figure 3.16). The sigmoid function $f(Z)$ is given as:

$$f(Z) = \frac{1}{1 + e^{-\alpha Z}} . \tag{10}$$

Tanh has a similar shape as the sigmoid function. It has continuous boundaries between negative one and one (Figure 3.16). The tanh function $f(Z)$ is given as:

$$f(Z) = \frac{e^{Z} - e^{-Z}}{e^{Z} + e^{-Z}} . \tag{11}$$

Relu has a different shape to the sigmoid and tanh function (Figure 3.17). The mathematical formulation of relu is simple: $max(0, z)$. Relu is not a linear function and is less prone to the vanishing gradient problem (described in Section 3.3.2).

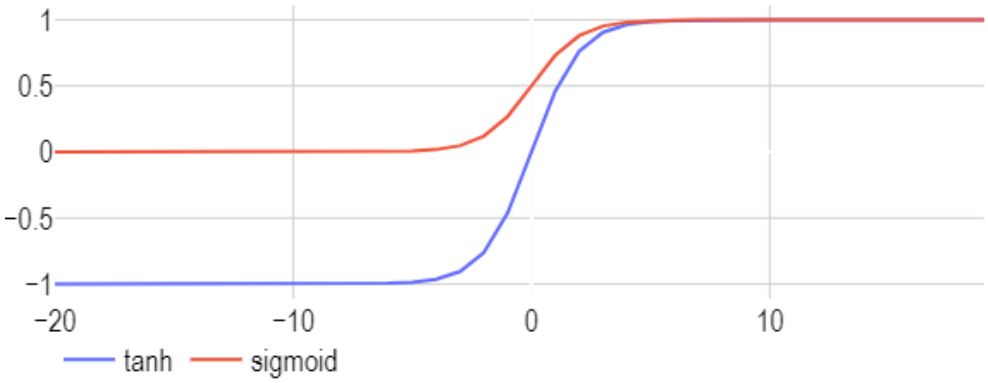

Figure 3.16: Sigmoid and tanh activation functions.

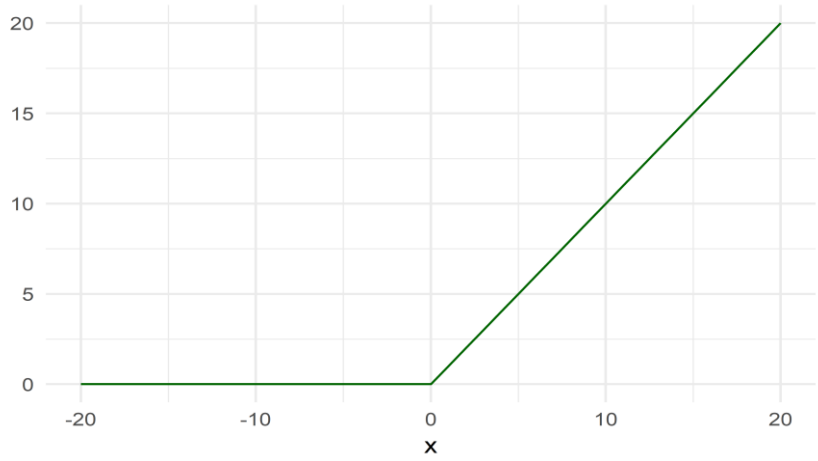

Figure 3.17: Relu activation function.

A neural network exchanges information between layers through the connections (Figure 3.15). Each connection has a value and is represented as w known as the weight. “Each layer has a bias neuron b which does not receive data from the previous layer” (Inzaugarat, 2018). The passing of information through the network starts at the input layer 0 and ends at the output layer L. The process in which a neural network transforms inputs into outputs is known as forward

propagation. The forward propagation for neuron s in layer zero defined as a_s^0 can be written as:

$$a_s^0 = f(w_{s1}^0 x_{i1} + w_{s2}^0 x_{i2} + \cdots + w_{SR}^0 x_{iR} + b_s^0) \tag{12}$$

where i is the input observation.

The forward propagation for neuron s in layer one defined as a_s^1 can be written as:

$$a_s^1 = f(w_{s1}^1 a_1^0 + w_{s2}^1 a_2^0 + \cdots + w_{SR}^1 a_{SR}^0 + b_s^1). \tag{13}$$

In summary, the result of a forward pass for neuron s in layer l is a weighted sum of the inputs with respect to each neuron. Therefore, in general:

$$a_s^l = f\left(\sum_{r=1}^{R} w_{sr}^l a_s^{l-1} + b_s^l\right) \tag{14}$$

where:

$$w_{sr}^l \begin{cases} 1 \leq l \leq L \; number \; of \; layers \\ 1 \leq s \leq S^l \; number \; of \; neurons \\ 1 \leq r \leq R^l \; number \; of \; inputs. \end{cases}$$

For the output layer:

$$w_{sk}^l \begin{cases} 1 \leq l \leq L \; number \; of \; layers \\ 1 \leq s \leq S^l \; number \; of \; neurons \\ 1 \leq k \leq k^l \; output \; classes. \end{cases}$$

The vectorised implementation of Equation 14 is written as:

$$a^l = f(w^l a^{l-1} + b^l). \tag{15}$$

Further:

$[a^l \ldots a^L]$ are the vector inputs given to the neural network;

w_{sr}^{l} are the weight vectors that determine the contribution of inputs a^{l-1} to the outputs a^{l};

b^{l} is the bias term and represents the constant term that does not depend on any input value; and

The outputs Z_s shown in Figure 3.15, are the linear units not transformed by the activation function f.

After forward propagation, the next step is to compare the output produced to the desired/target result. “Error values are assigned to each neuron in the output layer. The error values are propagated back from the output layer to the hidden layers” (Rudorfer, 1997). Error values are determined using a loss function. A loss function measures the squared or absolute error between a network’s output and some target or desired output. Examples of loss functions are the “Mean Squared Error” (MSE) for regression and the “cross-entropy” for classification. The time series problem can be thought of as a regression problem. Therefore, the loss E can be written as:

$$E = \frac{1}{2}\sum_{k \in K}\sum_{i \in n}(t_{ik} - \hat{y}_{ik})^2 \tag{16}$$

where:

t_{ik} is the target value for training observation i of output class k;

$\hat{y}_{ik}$ is the predicted value for training observation i of output class k; and

n is the total number of observations.

The weights of the neural network are updated in the direction that minimizes the loss function. For the output layer, the weights of a neural network are adjusted using the equation below:

$$w_{sk}^{new} \leftarrow w_{sk}^{old} + \Delta w_{sr} \tag{17}$$

$$\Delta w_{sk} = \eta \delta_k a_s \tag{18}$$

where:

η is the "learning rate" and δ is known as the "transfer function". The unit $\delta_k a_s$ is equal to $\frac{\partial E}{\partial w_{sk}}$ and this is known as "the gradient of the output function" (Rudorfer, 1997). The gradient is a measure of how much the weight value needs to be adjusted such that it minimizes the error/loss. The process in which a weight of the neural network is adjusted in proportion to how much it contributes to the overall error is known as backpropagation. The key computation in backpropagation is the calculation of the gradients. Given an output layer with the weights w_{sk}^2, taking the derivative of the error E (Equation 16) with respect to w_{sk}^2 gives:

$$\frac{\partial E}{\partial w_{sk}^2} = \frac{\partial}{\partial w_{sk}^2} \frac{1}{2} \sum_{k \in K} (t_{ik} - \hat{y}_{ik})^2$$

$$\frac{\partial E}{\partial w_{sk}^2} = a_s^1 \delta_k \tag{19}$$

where

$$\delta_k = (t_{ik} - \hat{y}_{ik}) \hat{y}_{ik} (1 - \hat{y}_{ik}). \tag{20}$$

For a hidden layer with weights w_{sr}^1, taking the derivative of error E (Equation 14) with respect to w_{sr}^1 gives:

$$\frac{\partial E}{\partial w_{sr}^1} = \frac{\partial}{\partial w_{sr}^1} \frac{1}{2} \sum_{k \in K} (t_{ik} - \hat{y}_{ik})^2$$

$$\frac{\partial E}{\partial w_{sr}^1} = x_{ir} \delta_s \tag{21}$$

where:

$$\delta_s = a_s^1(1 - a_s^1) \sum_{k \in K} \delta_k w_{ks}^2 \tag{22}$$

updating of the weights:

$$w_{sr}^{new} \leftarrow w_{sr}^{old} + \Delta w_{sr} \tag{23}$$

$$\Delta w_{sr} = \eta \delta_s x_{ir} \,. \tag{24}$$

The equations and notations in this section have been adapted from Tom (1997).

A clear objective of time series modelling is to identify the relationship between successive inputs. Feed-forward neural networks require that the temporal dependencies be known upfront. In many practical applications, temporal dependencies are not known upfront and must be discovered. Processing sequences is a challenge for feedforward neural networks. RNNs are an alternative neural network architecture designed for sequence processing and to overcome this limitation.

3.3.2 Recurrent Neural Networks for Time Series

The primary feature that differentiates RNNs from feed-forward neural networks is the explicit handling of order between observations. This property is emphasised by comparing Figure 3.18 and Figure 3.19.

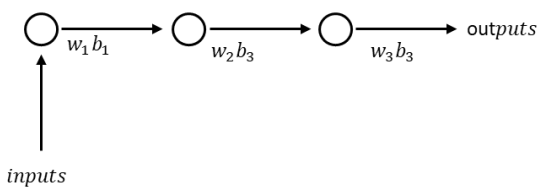

Figure 3.18: A graphical representation of a feed-forward flow.

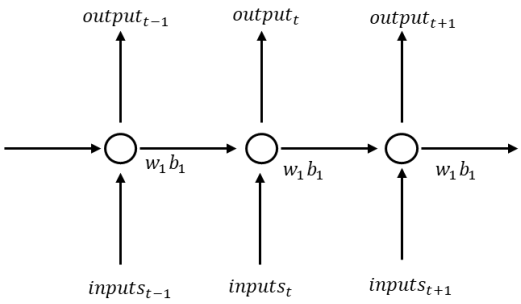

Figure 3.19: A "graphical representation" of a recurrent flow.

Figure 3.18 represents the feed-forward machine learning flow with one hidden layer. In the feed-forward flow:

- The layers receive and process the full sequence (that is the pre-defined temporal dependencies) at once.
- Each layer has different weights.

In the recurrent flow Figure 3.19:

- Subsequent input is passed through the same weights.
- Each layer has the same weights.

In the feed-forward flow, if inputs within a sequence are related or if sequences are related, the neural network would not detect this. A recurrent neural network can relate subsequent inputs to each other by passing the inputs through the same weight. This gives a neural network memory and a new dimension t known as the time step.

Figure 3.20 is a more detailed illustration of the recurrent flow. The RNN has three weight matrices W_{xh} W_{hh} and W_{hy}. W_{xh} is used for input x at time step t. W_{hh} is used for the hidden state h at time step t_{t-1} and W_{hy} is used for the output y_t. The weight matrices are the same at each time step. Therefore, the hidden state gets passed from one time step to the next with a fixed weight parameter W_{hh}, W_{xh} and W_{hy}. Algebraically this can be expressed as:

$$h_o = 0 \tag{25}$$

$$h_t = f(h_{t-1}, x_t)$$
$$h_t = f(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \tag{26}$$

$$y_t = W_{hy}h_t + b_y. \tag{27}$$

In Equations 25 – 27, the weights are matrices and the other variables are vectors, b is a vector of biases. The recurrence relation starts with h_0, a vector of zeros. In the RNN, the next hidden state of a neuron is the weighted sum of inputs from its current state plus the weighted sum of outputs from the previous state with activation and bias applied. Whilst in the feed-forward neural network, the activation a was the weighted sum of inputs from the current state with no additional information from the previous state.

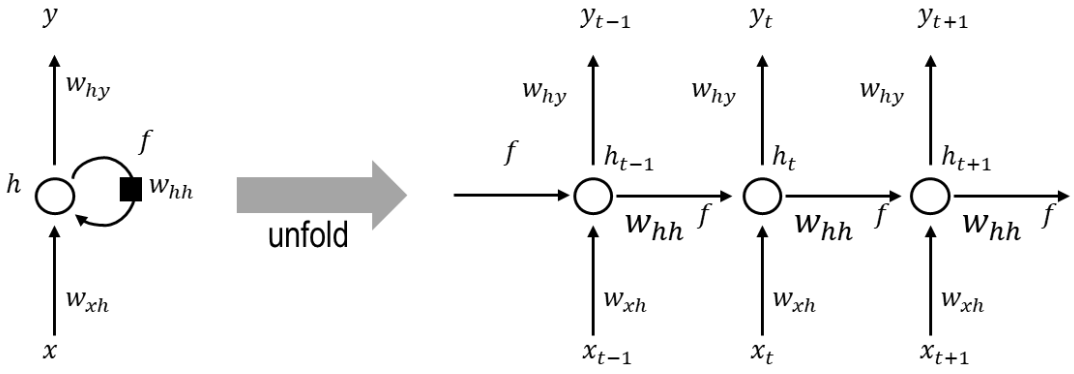

Figure 3.20: RNN unfolded architecture representation.

During the backpropagation of RNNs, the gradients of the weight matrices W_{yh}, W_{hh} and W_{xh} with respect to the error are updated. This is the same as the backpropagation process of a feed-forward neural network, except the backpropagation sums up all the gradients at each time step t for each training example. This is mathematically expressed as:

$$\frac{\partial E}{\partial W} = \sum_{t} \frac{\partial E_t}{\partial W} \tag{28}$$

Like the backpropagation of a feed-forward neural network, the chain rule of differentiation is used to calculate the gradients (see Figure 3.21). In Figure 3.21 starting with W_{hy} to backpropagate the error E_3, the chain rule is as follows:

$$\frac{\partial E_3}{\partial W_{hy}} = \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial W_{hy}}. \tag{29}$$

Because W_{hy} is shared across all t, differentiating at each time step and summing all the steps together gives:

$$\frac{\partial E_t}{\partial W_{hy}} = \sum_{t} \frac{\partial E_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial W_{hy}}. \tag{30}$$

To derive the gradient w.r.t. W_{xh} considering the time step $t = 3$

$$\frac{\partial E_3}{\partial W_{xh}} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial h_3} \frac{\partial h_3}{\partial h_k} \frac{\partial h_k}{\partial W_{xh}}. \tag{31}$$

For all t

$$\frac{\partial E_t}{\partial W_{xh}} = \sum_{i} \sum_{k=1}^{t} \frac{\partial E_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W_{xh}}. \tag{32}$$

To derive the gradient w.r.t. W_{hh} considering the time step $t = 3$

$$\frac{\partial E_3}{\partial W_{hh}} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial h_3} \frac{\partial h_3}{\partial h_k} \frac{\partial h_k}{\partial W_{hh}} . \tag{33}$$

Similarly, for all t

$$\frac{\partial E_t}{\partial W_{hh}} = \sum_{i} \sum_{k=1}^{t} \frac{\partial E_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W_{hh}} . \tag{34}$$

Note $\frac{\partial E_3}{\partial W_{hy}}$ only depends on the current state $h_3, \hat{y}_3 \; and \; y_3$.

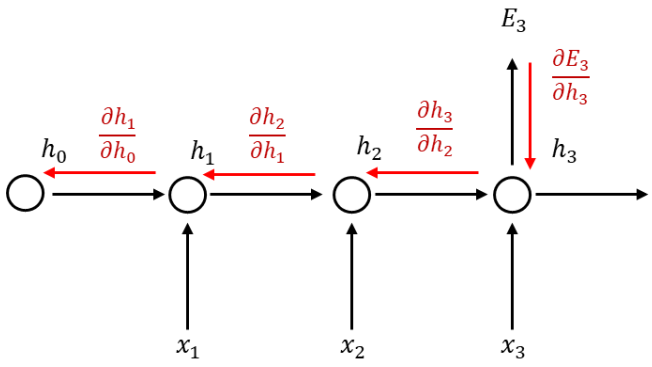

Figure 3.21: The chain rule for a simple RNN.

The RNN architecture, however, has a limitation. A standard RNN cannot bridge more than five to ten time steps. Error signals between subsequent steps tend to vanish. Therefore, updates made to the weights will be very small, and the model may not converge (that is, reach a global optimal solution). This is known as "the vanishing gradient problem". This limitation is expressed in the backpropagation algorithm Equations 31 – 43. For example, $\frac{\partial h_3}{\partial h_k}$ is a chain rule itself as $\frac{\partial h_3}{\partial h_1} = \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1}$. The critical thing to note in the backpropagation equation is that the gradient is calculated from many terms taking the form $f'(z)$ (that is, outputs transformed by the activation function tanh or sigmoid). These outputs are very small, and the chain rule is a series

of vector multiplications of very small numbers. Eventually multiplying many small numbers leads to yielding even smaller numbers continuously. The vanishing gradient becomes even more of a critical problem the deeper the network gets, and RNNs tend to be deep networks. When it comes to time series analysis, the sequences given to the network can be very long and each time step is a hidden layer. The vanishing gradient impacts RNNs ability to model long-term dependencies.

3.3.3 LSTMs for Time Series Forecasting

LSTMs are extensions of RNNs and offer a solution that addresses the vanishing gradient problem and can remember information for more extended periods of time. With LSTMs, rather than each hidden neuron having a simple configuration with a single activation function, each neuron is a cell with several activation functions. Like simple RNNs, LSTMs pass their new cell state to the next hidden layer. The information flow for a standard LSTM is depicted in Figure 3.22.

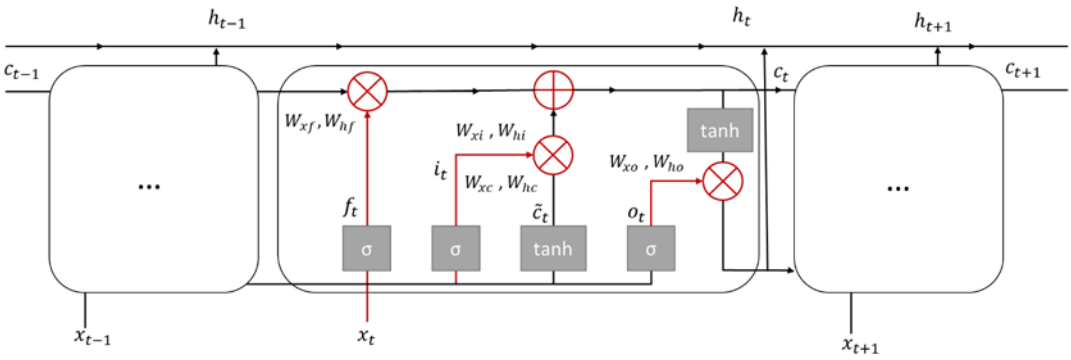

Figure 3.22: LSTM information flow.

The core of the LSTMs' operation relates to how information is regulated at its gates. A gate consists of a sigmoid function and a pointwise operator. Figure 3.22 represents the pointwise operators in red and the sigmoid function as σ. Standard LSTMs have three gates: a forget gate f_t, an input gate i_t and an output gate o_t. c_t is the core memory unit of the LSTM and is known as the memory cell or cell state

at time step t. The forget gate is responsible for removing information from the cell state, the input gate adds information to the cell state, and the output gate further filters out information from the cell state (Fischer, 2018). Given the sequence data $\{x_1, \dots, x_T\}$, the gate operations are defined as follows:

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \quad (35)$$

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \quad (36)$$

$$\tilde{c}_t = tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (37)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (38)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \quad (39)$$

$$h_t = o_t \odot \tanh(c_t), \quad y_t = W_{hy}h_t + b_y \quad (40)$$

where:

$\odot$ is an operation for the element-wise multiplication of vectors;

h_{t-1} is the hidden state of the previous memory cell;

h_t is the hidden state of the current memory cell;

b_f, b_i, b_o, b_y are the bias estimated parameters; and

$w_{xf}, w_{hf}, w_{xi}, w_{hi}, w_{xc}, w_{hc}, w_{xo}, w_{ho}, w_{hy}$ are the weight parameters.

The forget gate (Equation 35) considers h_{t-1}, the hidden state from the previous time step and x_t, the input at time step t and applies the sigmoid function that outputs a vector of numbers ranging from zero to one. The sigmoid function is therefore responsible for deciding which values to keep and those to discard. The input gate, i_t, is responsible for adding information to the cell state. This is a three-step process. The first step is regulating what information needs to be added (Equation 36) by applying the sigmoid function. The second step (Equation 37) creates candidate values $\tilde{c}_t$ that could be added to the

state. This is done by using the tanh function, which outputs values between -1 and +1. The third step updates the old cell state c_{t-1} to c_t (Equation 38). The final operation within an LSTM cell is deciding what to output h_t (Equation 39 - 40). This is a two-step process. The first step is selecting useful information from the current cell state (Equation 39). o_t is the information filter that is applied to the current cell state c_t.

The above Equations (35 – 40) are the key operations that make LSTMs better at managing the vanishing gradient problem. The backpropagation process that involves updating the weight parameters: $w_{xf}, w_{hf}, w_{xi}, w_{hi}, w_{xc}, w_{hc}, w_{xo}, w_{ho}, w_{hy}$ for LSTM can be found in Chen (2018).

3.3.4 LSTM Architectures

LSTMs have five different architectures, and these are shown in Figure 3.23. Here, each rectangle is a vector, and the arrows represent functions (e.g. matrix multiplication). From left to right, the architectures are explained as follows:

1. One to one: This is the standard LSTM architecture and is commonly known as the “vanilla” LSTM. The LSTM takes a fixed-size input and transforms it to a fixed-sized output. An example of a task that uses this architecture is image classification.

2. One to many: The LSTM takes a fixed input and produces an output sequence (example, an image captioning task that takes an image as input and outputs a sentence).

3. Many to one: The LSTM takes a sequence as input and produces a fixed output (example, sentiment classification).

4. Many to many: The LSTM takes a sequence as input and produces a sequence as output (example, a machine translation task that reads a sentence in French and outputs a sentence in English).

5. Many to many: The LSTM takes a synced sequence input and output (example, a video classification task that labels each frame of the video).

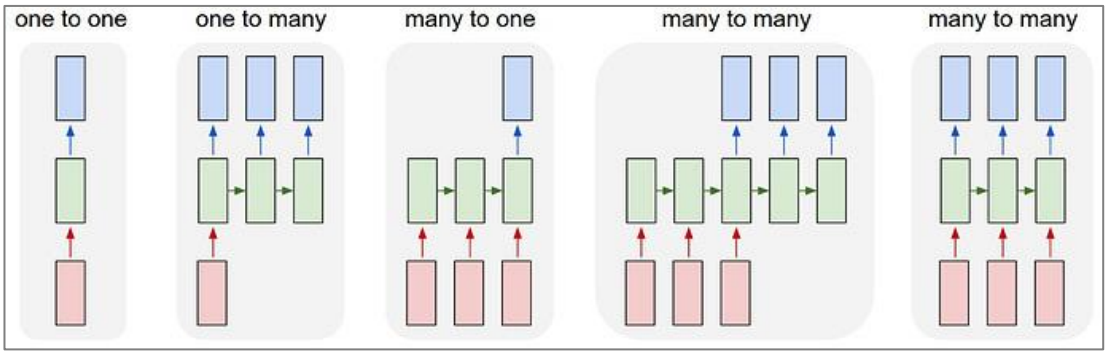

Figure 3.23: LSTM architectures adapted from http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

Note that there are no pre-defined constraints on the length of the sequence. This thesis investigates two types of LSTM sequence processing architectures: 1) one to one for univariate time series forecasting and 2) many to one for multivariate time series forecasting.

3.3.5 Summary LSTM Capabilities

LSTMs offer additional capabilities that are challenging to classical time series and extend the capabilities of feed-forward neural networks and the "vanilla" RNN. These capabilities are summarized as follows:

1. The ability to learn complex non-linear patterns from inputs: the presence of activation functions, and the ability to have multiple hidden layers enables neural networks to learn non-linear patterns.
2. LSTMs do not require the time series to be stationary
3. Ability to process sequences, which is a limitation of feedforward neural networks
4. LSTM support the efficient learning of the temporal dependencies: they can better handle the vanishing gradient as they have a more complex cell structure that does not rely on one activation function.

3.4 Evaluation metrics

The performance of the models needs to be evaluated and compared. For time series forecasting, the five most commonly used evaluation metrics in the literature are: "the root-mean-square error (RMSE), the mean absolute error (MAE), the symmetric mean absolute percentage error (SMAPE), the BIAS, and the correlation function between the forecast, x' and the measured time series, x" (Panigrahi, 2017). These metrics are mathematically expressed as:

$$RMSE(x', x) = \sqrt{\frac{1}{N} \sum_{n=1}^{N} (x'_n - x_n)^2} \quad (41)$$

$$MAE(x', x) = \frac{1}{N} \sum_{n=1}^{N} |x'_n - x_n| \quad (42)$$

$$SMAPE(x', x) = \frac{100}{N} \sum_{n=1}^{N} \left| \frac{x'_n - x_n}{x'_n} \right| \quad (43)$$

$$BIAS(x', x) = \frac{100}{N} \sum_{n=1}^{N} (x'_n - x_n) \quad (44)$$

$$Correlation(x', x) = \frac{\sum_{n=1}^{N} (x'_n - \bar{x}'_n) \times \sum_{n=1}^{N} (x_n - \bar{x}_n)}{\sqrt{\sum_{n=1}^{N} (x'_n - \bar{x}'_n)^2 \times \sum_{n=1}^{N} (x_n - \bar{x}_n)^2}} . \quad (45)$$

The RMSE and the MAE measure the mean deviation between forecasts and observed values. However, because the calculation of the RMSE involves calculating the average of the squared deviations, it is more sensitive to outliers in the data. The SMAPE on the other hand measures the average percentage error while the BIAS is a measure of the mismatch between the model predictions and the observed output of the system. Finally, the correlation is a measure that shows how the predicted values compare with the observed values (Abdullayeva, 2019). Smith et al. (2017) used RMSE, MAE, and SMAPE to evaluate the results of their study. This thesis uses the same forecast evaluation metrics.

3.5 Proposed Approach to Social Unrest Prediction

Figure 3.24 is an overview of the proposed social unrest prediction framework that lays out the structure of the data analysis and implementation component of this thesis. The proposed framework consists of five processes: event data analysis, time series analysis, data preparation, experiments and lastly, prediction results. All these processes are explained in more detail in the respective sections below.

The event data analysis section describes how the data was collected, and the variables used in the machine learning models. The time series analysis section implements the time series exploratory analysis procedures discussed in Section 3.3 (identifying time series properties). The data preparation section discusses all the data transformations and preparation steps, such as scaling and differencing, applied to the data. The experiments section describes all the experiments conducted to train the data on ARIMA and LSTM methods. The results section

compares ARIMA and LSTM model performance using the performance metrics RMSE, MAE, and SMAPE.

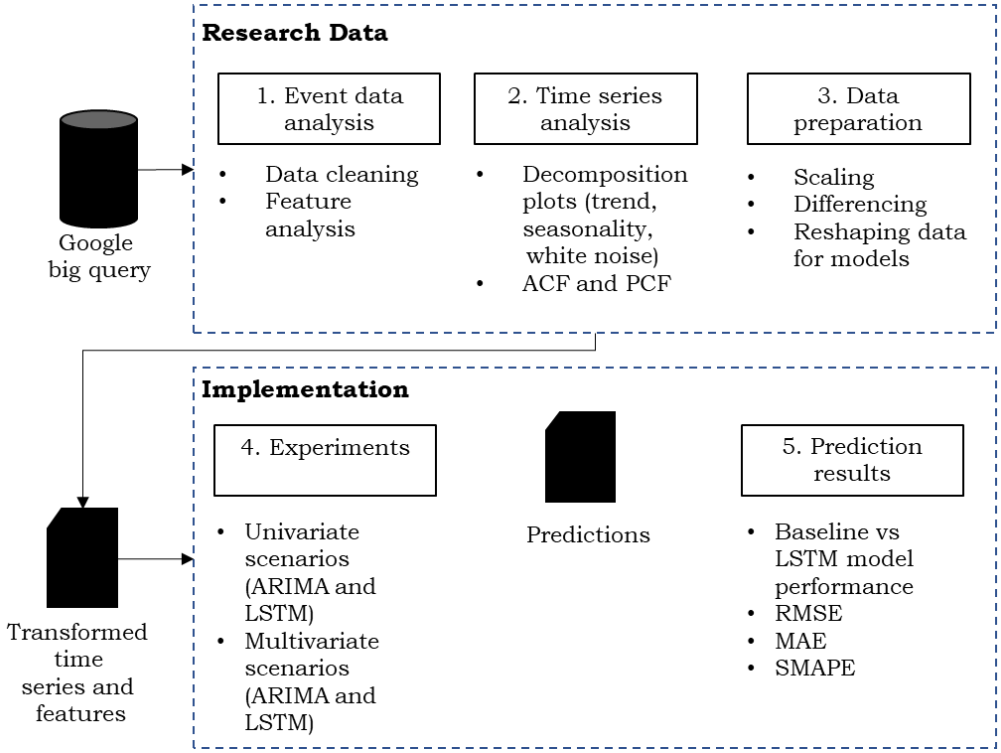

Figure 3.24: The proposed research design framework.

Chapter 4

Research Data

This chapter describes the GDELT data, and the variables used to predict social unrest. It also covers the feature transformations performed on the data.

4.1 Event Data Analysis

GDELT monitors news media across the world to identify events and currently contains a list of over 300 event categories. These events include “protests”, “peaceful appeals”, “diplomatic agreements”, “diplomatic apologies”, “public demands”, and reports on threats. Each event record has 61 fields capturing details of the event, including its georeferenced location and actors mentioned. Actors are entities such as individuals, countries, identity groups, religious groups, political parties, and organizations.

This thesis aims to predict social unrest using the baseline variables in GDELT data used in Smith et al. (2017). These were SQLDATE, GoldsteinScale, NumMentions, NumSources, NumArticles, AvgTone, ActionGeo_Lat, ActionGeo_Long and ActionGeo_CountryCode. In addition, the following variables are included: EventRootCode, EventCode, Actor1Type1Code, and QuadClass.

4.1.1 Selected GDELT variable description

SQLDATE is the date the event took place. The event type data is captured in a special format known as “Conflict and Mediation Event Observations (CAMEO)”. The EventCode contains the event type and the EventRootCode defines the highest-level category. For example,

EventCode 1452 (“engage in violent protest for policy change”) has a root CAMEO code of 14 (PROTEST). This grouping makes it possible to aggregate events to the root code level. Details of all CAMEO codes and their definitions are available on the GDELT website and in Gerner (2002). GoldsteinScale is “a numeric score that ranges from -10 to 10” and is assigned to an event to capture the potential theoretical “impact that type of event will have on the stability of a country”. NumMentions is “the total number of times the event is mentioned in all news media sources”. NumSources is “the total number of unique information sources that have mentioned the event”. AvgTone is “the average tone of all documents containing one or more mentions of this event. The score ranges from -100 (extremely negative) to +100 (extremely positive). Common values range between -10 and +10, with 0 indicating neutral” (GDELT). ActionGeo_Lat and ActionGeo_Long are “the centroid latitude and longitude of the location of the event activity”. ActionGeo_CountryCode is the CAMEO country code for where the event took place. Actor1Code is the primary actor involved in the event. An actor could be a geography, class, ethnic group or religious group. Actor1TypeCode is a grouping of the actor codes based on their affiliations, for example, government, military, and state intelligence. The QuadClass groups events into four categories based on severity: “verbal cooperation, material cooperation, verbal conflict, and material conflict”.

More details on the methods GDELT uses to extract features from news articles are in the codebook available on the GDELT website (https://www.gdeltproject.org/).

4.1.2 Getting the Data

GDELT is available on Google BigQuery and is accessed through the Google Cloud Platform. BigQuery is a distributed data warehouse and is used to store large volumes of data in a structured format. Data is

extracted from BigQuery using Structured Query Language (SQL). The goal of this thesis is to predict social unrest in South Africa using GDELT. The data was filtered for South Africa, where the country code is “SF”, and the actor code is “ZAF”. Data was collected covering the period 01 January 2001 to 31 December 2017 for all CAMEO event codes.

4.1.3 Social Unrest Event Root Codes

The variable analysed in this section is the EventRootCode. The EventRootCode consists of 20 root CAMEO codes. Figure 4.1 shows the count of events by root CAMEO codes over the analysis period (01 January 2001 to 31 December 2017). The most reported event category is consult (CAMEO 04). Events falling under the category consult are diplomatic meetings and mediations - they can be either face-to-face or telephonic conversations. The protest category (CAMEO 14) falls in the bottom 6 of reported events in South Africa. According to GDELT, protests are non-military demonstrations. The protest category events are quite sparse; therefore, similar to Smith et al. (2017) this thesis does not only use the protest category on its own but also includes other root CAMEO codes that involve other forms of material conflict such as military mobilizations and physical attacks. The added CAMEO codes are 15, 17, 18, 19, and 20. The resulting category or group is referred to throughout the document as social unrest.

CAMEO 15 is “exhibit force posture”, and events falling under this category are “armed force mobilization”. Coerce (CAME0 17) means “repression, violence against civilians, and damage to properties”. Assault (CAMEO 18) are “extreme acts of violence and groupings such as terrorism and bombs”. Fight (CAMEO 19) is a category that includes acts of “fighting and killings of any kind”, and the final category is “use unconventional mass violence” (CAMEO 20). CAMEO 16 is called

"reduce relations" and there is no indication of any forms of mobilization in this category; for this reason, CAMEO 16 is excluded.

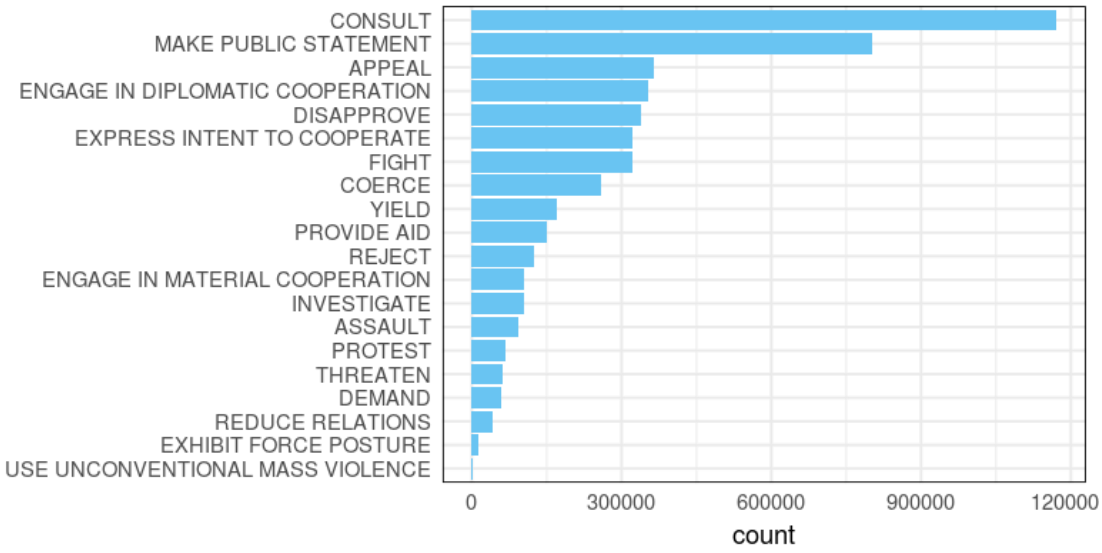

Figure 4.1: Count of events by RootCodeEvents 2001 - 2017 South Africa.

4.1.4 Social Unrest Event Codes

This section takes a closer look at the EventCode variable falling under the root CAMEO codes 14, 15, 17, 18, 19, and 20. Figure 4.2 depicts the count of events by EventCode. Figure 4.2 shows the top 20 occurring social unrest events. Here, the category "use conventional military force" is the most reported event with over 200000 reports over the analysis period (01 January 2001 – 31 December 2017). According to GDELT, "conventional military force" is defined as: "All uses of conventional force and acts of war typically by organized armed groups not otherwise specified."

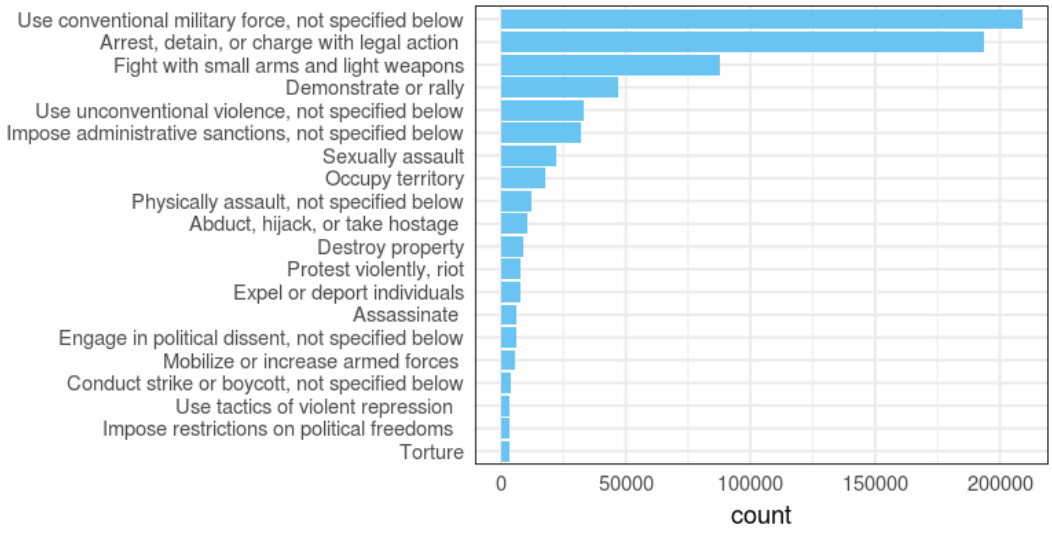

Figure 4.2: Count of the top 20 events by EventCode South Africa 2001-2017.

4.1.5 Social Unrest Actor Types

This section looks at the Actor1TypeCode variable. This variable is a categorical variable not used in Smith et al. (2017) and includes 40 categories. Figure 4.3 represents the top 20 actor types and shows the total number of times the actor type is mentioned in an event for the CAMEO codes 14, 15, 17, 18, 19, and 20 over the analysis period 01 January – 31 December 2017. Here, the key players in the social unrest events taking place in South Africa in the analysis period are police forces and the government.

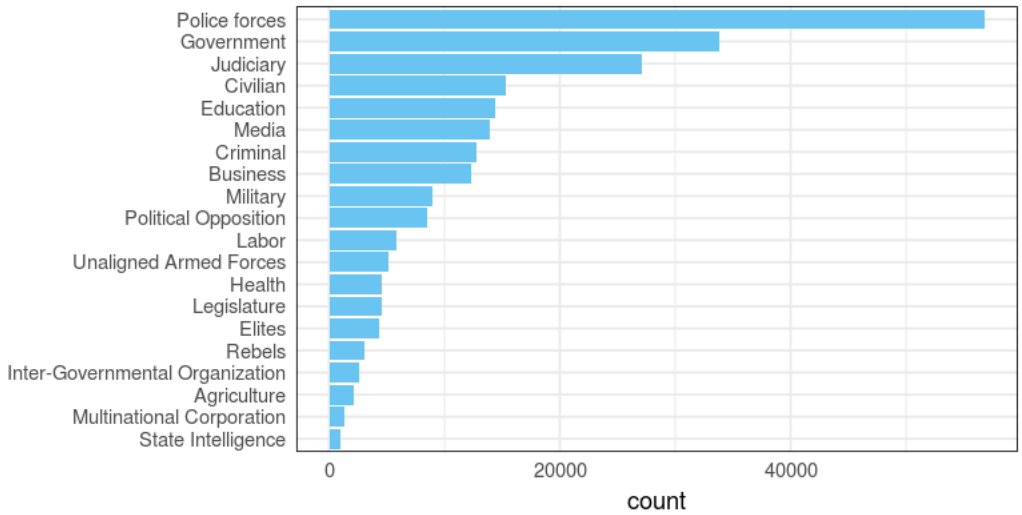

Figure 4.3: Count of actor types South Africa 2001 – 2017.

4.1.6 Other Event Root Codes

This thesis hypothesises that the inclusion of other event categories as explanatory variables will improve the quality of the predictions made by the models. This section looks at the data represented by other root CAMEO codes (that is, CAMEO 1 to 14 and 16). Figure 4.4 is the count of events by EventCode. Figure 4.4 depicts only the top 20 events. Here, the category "make statement" is the most reported event. The statements are reported verbal announcements relating to social, economic and political issues.

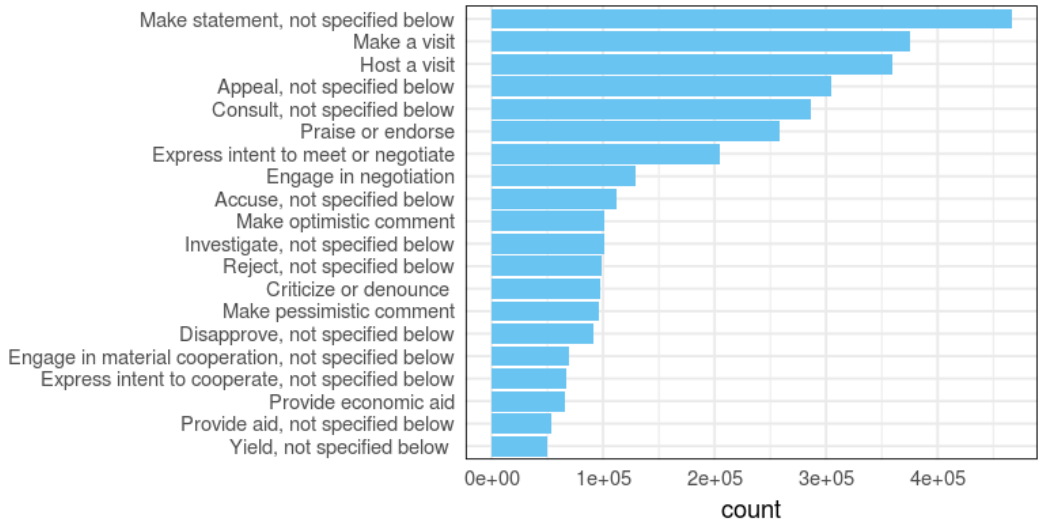

Figure 4.4: Count of other events by EventCode.

When events depicted in Figure 4.4 are analysed individually at the day level over time, their occurrences are sparse. Therefore, in this thesis, the events are aggregated by QuadClass.

4.1.7 Relationship Between Social Unrest and Explanatory Variables

This subsection looks at the linear relationship between social unrest and the explanatory variables. The explanatory variables are:

1. The sum for the number of times an actor type was mentioned (forty actor types).
2. The count of other event categories aggregated by quad class.
3. The sum for NumMentions, NumSources, and NumArticles of social unrest activity.
4. The mean for the AvgTone, Goldstein scale, latitude, and longitude.

The analysis consisted of 51 possible explanatory variables. Pearson correlation is used to determine the strength of the linear dependencies between social unrest and the explanatory variables (see Appendix B Table B2 for full correlations). Table 4.1 provides a summary of the top nine variables highly correlated to social unrest. The highly correlated events to social unrest include material cooperation, verbal cooperation, police forces, and government. These are new variables not included in the study of Smith et al. (2017). It will be worthwhile to investigate how these variables influence forecast accuracy.

Table 4.1: Top nine Correlation between social unrest and other variables in GDELT.

Variables	Correlation
ActionGeo_Long	0.937
Verbal_Conflict	0.927
NumMentions	0.921
NumArticles	0.921
NumSources	0.918
Material_Cooperation	0.911
Verbal_Cooperation	0.899
Police_forces	0.861
Government	0.821

Figure 4.5 shows the events over time for social unrest, material conflict events, material corporation events, verbal conflict, and verbal corporation events. The events appear to be correlated and correspond to the results in Table 4.1.

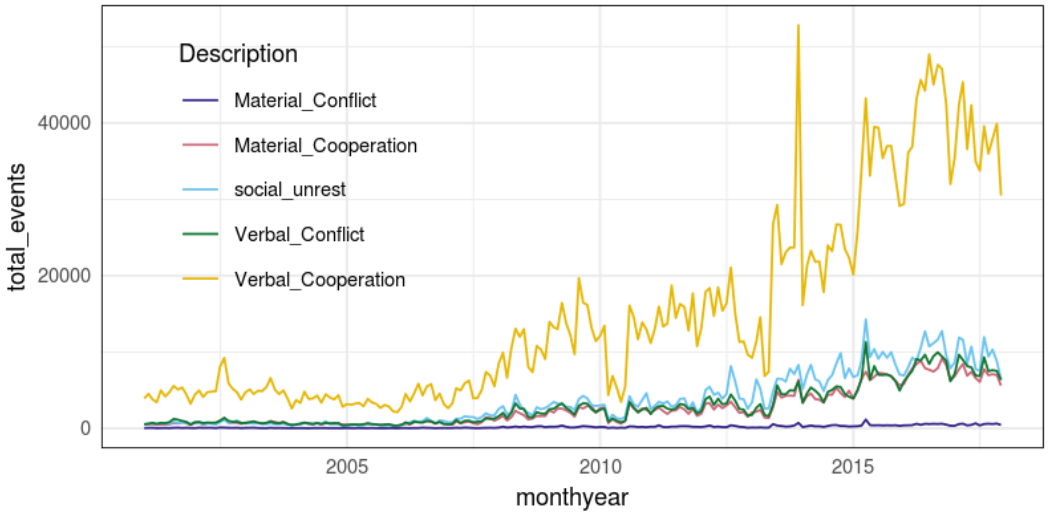

Figure 4.5: The relationship between social unrest and other events by QuadClass (The QuadClass groups events into four categories based on severity: "Verbal Cooperation, Material Cooperation, Verbal Conflict, and Material Conflict").

4.1.8 Data Representation

The social unrest time series is defined as the daily sum of 4491394 events (that is, root CAMEO code: 14, 15, 17, 18, 19 and 20) recorded between 31 January 2001 and 31 December 2017. The daily sum resulted in 6209 observations, 6209 days.

Before training the data on ARIMA and LSTM models, the data was partitioned into a training and test set. During training, the models learn patterns that occur in the time series. The learned patterns are evaluated against the test set. The smaller the deviation between the learned pattern and the test set, the better the model.

Figure 4.6 represents the social unrest event time series. The vertical line represents the cut-off between the training set and the test set. The cut off is on 12 July 2017. The data from 12 July 2017 to 31 December 2017 is the test set (approximately 3% of the data).

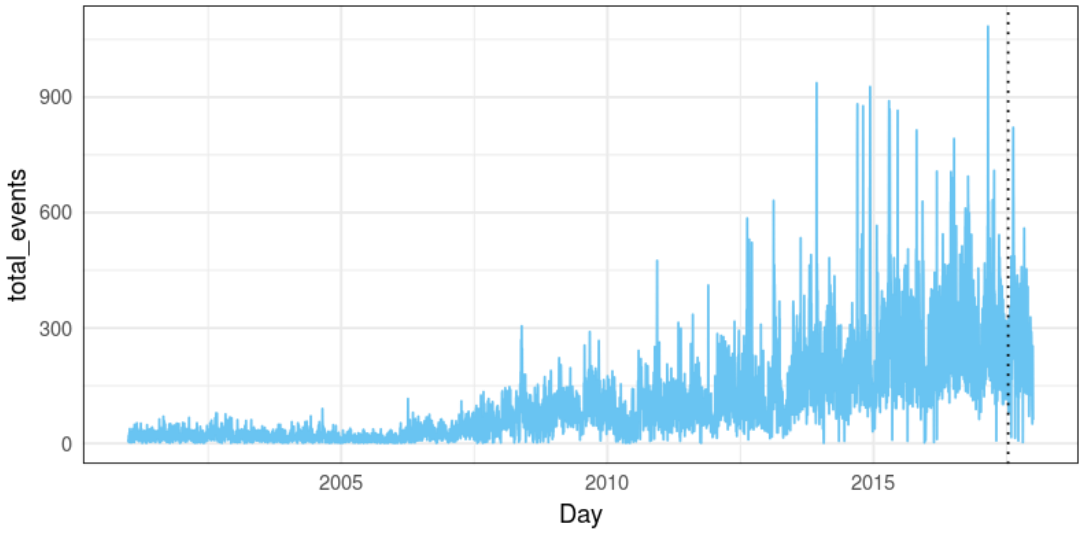

Figure 4.6: Count of Social unrest events by day South Africa 2001 – 2017.

The data is trained at the daily level and predictions are made at the daily level. Figure 4.7 represents the test set. The data has adequate variation to test the ability of the model in predicting peaks and troughs.

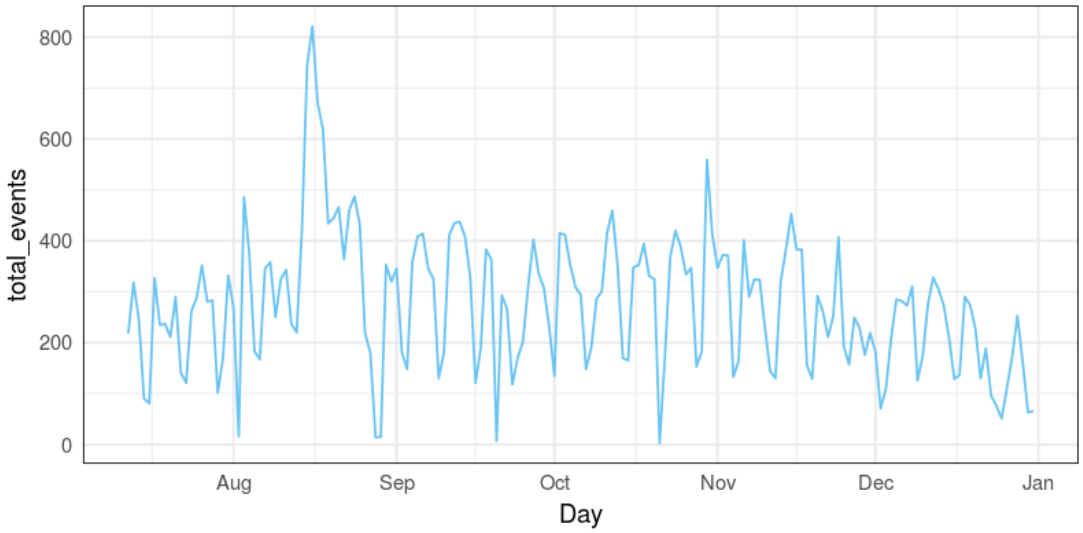

Figure 4.7: Social unrest time series test data South Africa.

4.2 Time Series Analysis

This section implements the time series exploratory analysis methods reviewed in Section 3.3. These methods are used to obtain the statistical properties of the social unrest time series. These properties assess if the time series is fit for ARIMA models. LSTMs are not as dependent on the statistical properties of the time series.

4.2.1 Decomposition Plots

Figures 4.8 to 4.10 are the time series decomposition plots: trend, seasonality, and white noise with year seasonality applied. The trend (Figure 4.8) captures a steep increase in social unrest from 2006 onwards. However, the increasing trend is not consistent, and because of this, the trend of the time series is not linear.

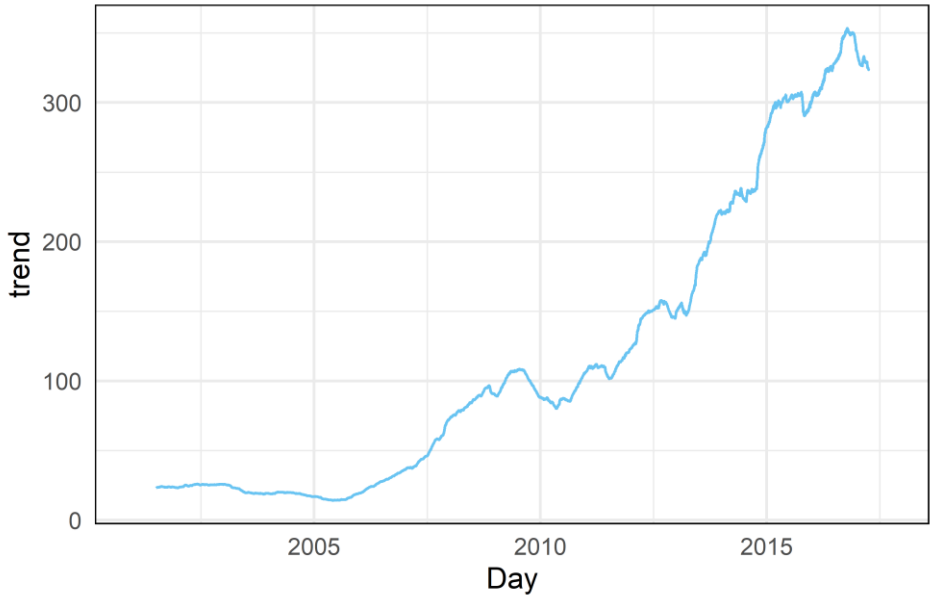

Figure 4.8: Time series decomposition plot: trend.

Figure 4.9 is the seasonal component of the time series. The time series appears to have a seasonal pattern that is constant.

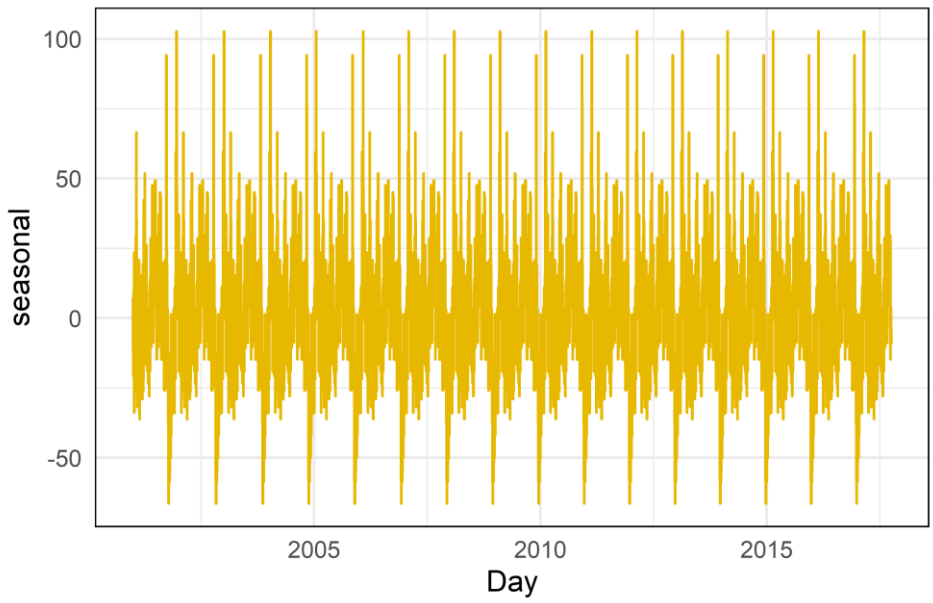

Figure 4.9: Time series decomposition plot: seasonality.

Figure 4.10 is the white noise component. When looking at the values on the y-axis, the time series appears to have a lot of irregular fluctuations.

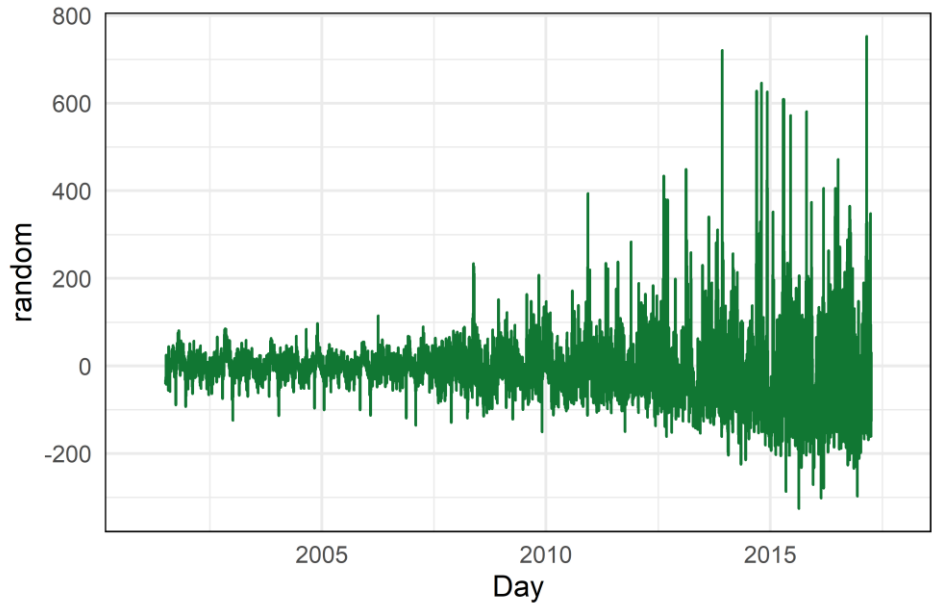

Figure 4.10: Time series decomposition plot: white noise.

From the decomposition plots, the properties of the social unrest time series are summarised as follows:

1. Skewed data as there is a significant increase in the number of protest events;
2. The time series trend is not linear;
3. Presence of a constant seasonal pattern; and
4. The significant presence of irregular fluctuations will make it challenging for a traditional ARIMA model to predict irregular fluctuations.

4.2.2 ACF and PACF

The ACF and PACF plots (Figures 4.11 and 4.12) are used to further verify some of the time series properties observed in the decomposition plots. For example, Figure 4.11 indicates the presence of a constant seasonal pattern. The seasonal patterns are the spikes at every seventh lag depicted in the plot.

Another property identified from the ACF plot is non-stationarity. Recall from Chapter 3 that for a stationary time series, the ACF drops to zero relatively quickly. The ACF for a non-stationary time series decreases slowly. The lags in Figure 4.11 are consistently above the dashed blue line and only decrease slightly.

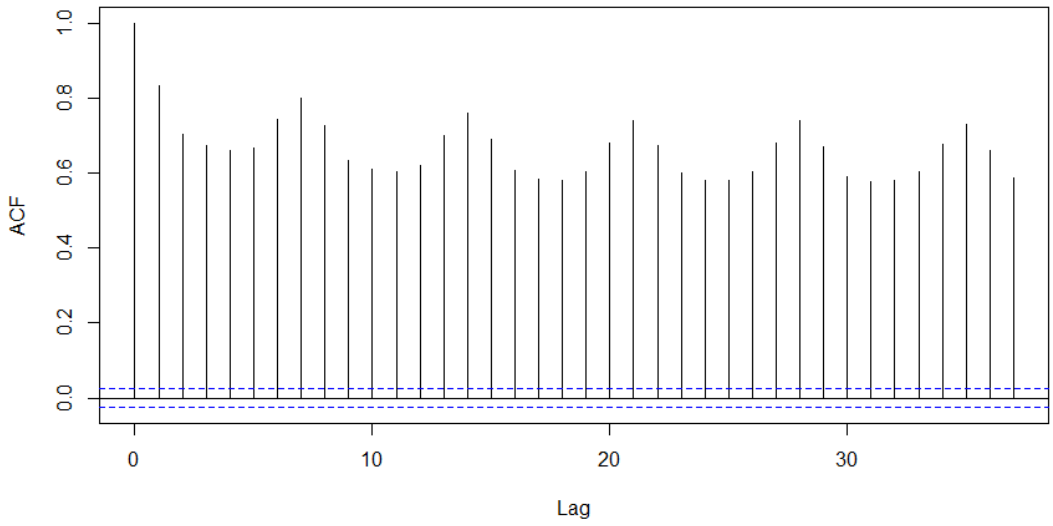

Figure 4.11: Social unrest time series ACF.

Figure 4.12 is the PACF plot. Recall the PACF plots are useful when determining the order of the AR(p) model. Most of the lag values lie outside the blue dotted line and do not trail off (diminish towards zero). AR(p) process for the social unrest time series cannot be identified with much certainty from the PACF plot. This observation further verifies that the time series is non-linear.

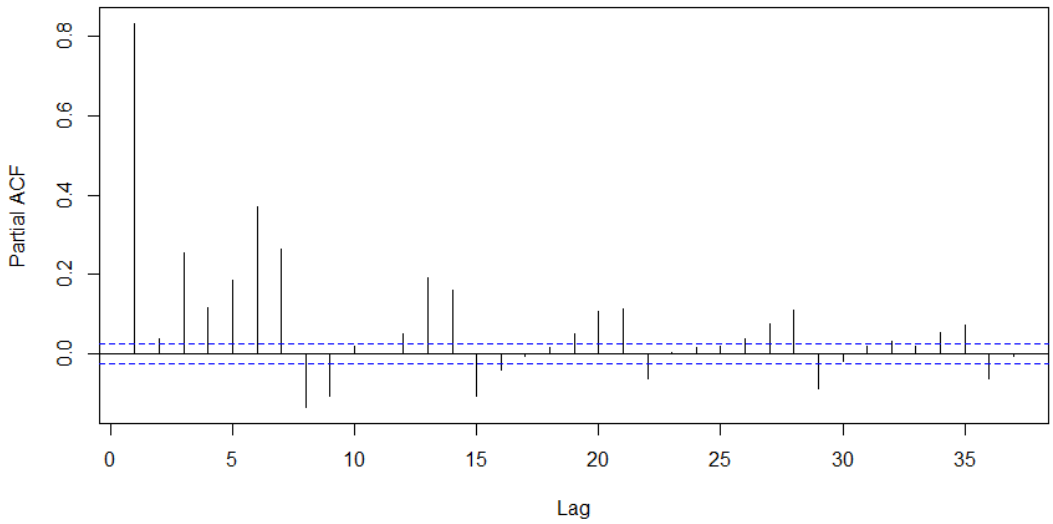

Figure 4.12: PACF plot for the social unrest time series.

4.3 Data Preparation

ARIMA baseline methods and LSTM models have different learning techniques; therefore, different data transformations are applied to the data sets.

4.3.1 Box-Cox Transformation

Box-Cox transformation is a pre-processing step only applied to the data trained on ARIMA. It is widely used on time series to remove skewness and other distributional features that can make analysis more complicated. The goal is "to find a simple transformation that leads to normality" (Box, 1964). Data that is normal can be estimated better with linear models.

At the core of the Box-Cox transformation is the parameter lambda (λ), which varies from -5 to 5. The Box-Cox transformation is expressed as:

$$y(\lambda) = \begin{cases} \dfrac{y^{\lambda} - 1}{\lambda}, & if\ \lambda \neq 0; \\ \log y, & if\ \lambda = 0. \end{cases} \qquad (46)$$

The R package forecast is used to select the optimal value of λ. The optimal value is the one that results in the best approximation of the normal distribution curve.

The social unrest time series has a non-linear trend and contains a significant number of irregular fluctuations. ARIMA models require the time series to be normally distributed, stationary and have a linear trend in order to determine the adequate model parameters p, d, q. Figure 4.13 is the transformed social unrest time series. Here, the Box-Cox transformation with λ = 0.247, together with one level of differencing (d = 1) was applied to the time series.

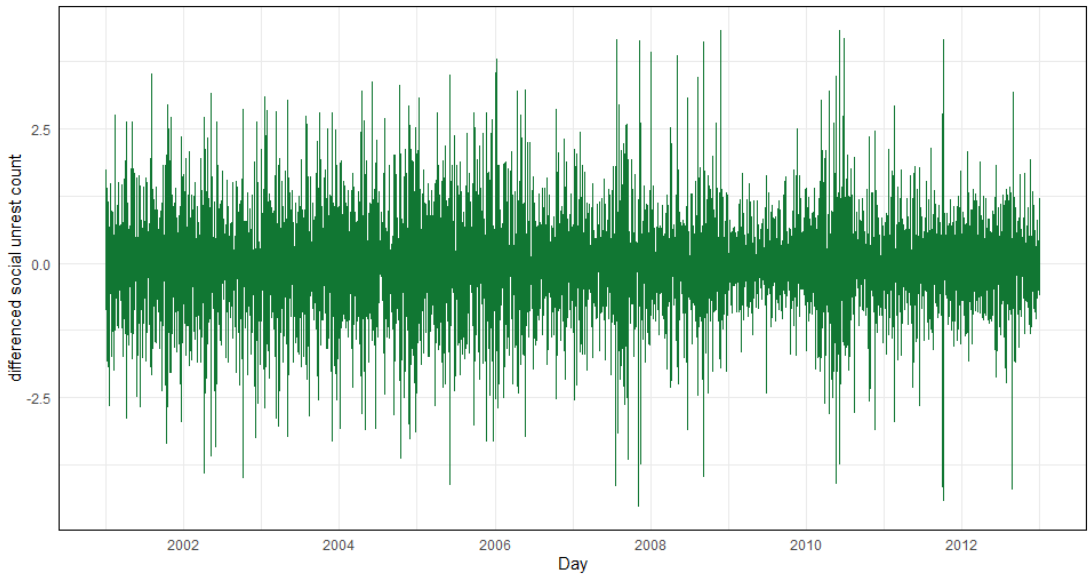

Figure 4.13: Box-Cox transformed social unrest time series with one level of differencing applied.

Figure 4.14 is the ACF plot of the transformed time series. The seasonal pattern has been reduced by the transformation and the ACF drops much quickly after lag one. There are more lag values falling within the dashed blue line and the time series has been made stationary. However, there are still some spikes at lags 7, 14, 21, 28, and 35. A multivariate model will be useful to explain some of the irregular fluctuations present in the data.

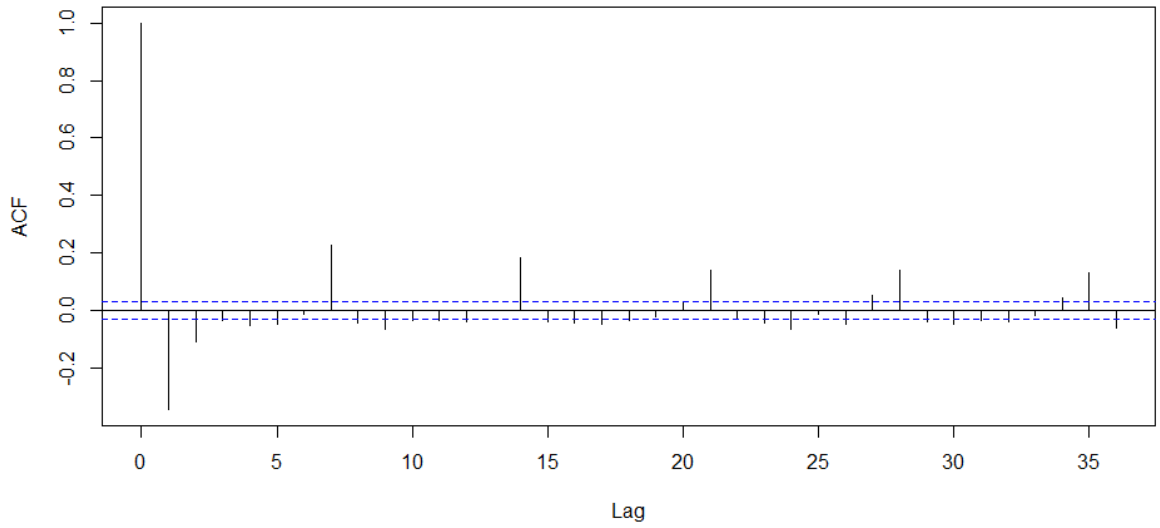

Figure 4.14*:* ACF of Box-Cox transformed time series.

The PACF of the transformed series, Figure 4.15, does not show much improvement from Figure 4.12. Many of the lags still fall outside the dashed blue line. The order of AR(p) is difficult to establish looking at Figure 4.15 alone.

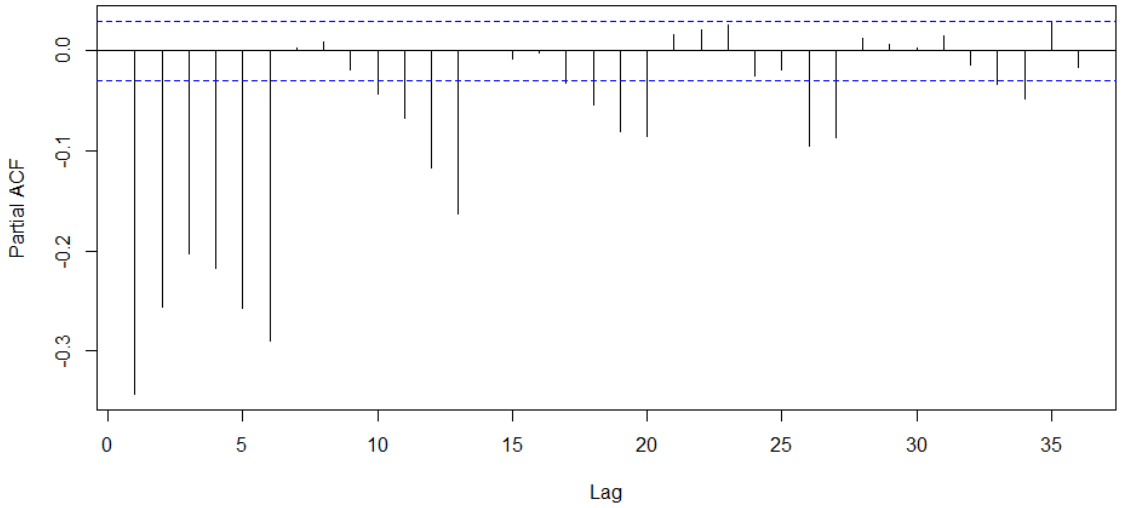

Figure 4.15: PACF of Box-Cox transformed time series.

Similarly, the order of the ARIMA(p, d, q) model cannot be identified with much certainty from the ACF and PACF plots (Figures 4.14 and 4.15) alone. For this reason, the R auto.arima function is used to identify the order of the ARIMA models.

4.3.2 Differencing

Differencing was discussed in Section 3.1.2. It is a method used to make time series stationary. One level differencing is applied to the univariate ARIMA model. The auto.armia function automatically selects and applies the optimal level of differencing.

4.3.3 Min-max Transformation

The research data are numerical variables. One of the challenges of working with numerical variables is the possible presence of variables with large values and variables with small values. Scaling ensures that no variable dominates the others in the analysis.

For the multivariate ARIMA model, the min-max scaling procedure is used to scale values of predictor variables. In contrast no scaling nor

transformation of any sort was applied to data for the multivariate LSTM model.

The min-max transformation applied is expressed as follows:

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{47}$$

where $x = (x_1, \ldots, x_n)$ is the data matrix and z_i is the i^{th} normalized data. More on scaling can be found in NcNelis (2005).

4.3.4 Time Series Data for Supervised Learning

ARIMA with regressors and LSTM neural networks are supervised learning algorithms. These algorithms require labelled data samples where each sample has an input component (x) and an output component (y). In this thesis, the input components (x) are the previous day values of the explanatory variables. The output component (y) is the current day's count of social unrest.

A sliding window approach was applied using the lag shift operator function available in Python pandas' library to turn the data into a form more suitable for supervised learning. Figure 4.16 depicts a sample of the transformation for a many to one LSTM architecture. Here the output component (y) is var9(t). The input components (x) are depicted as var1($t-1$), var2($t-1$) and so forth.

	var1(t-1)	var2(t-1)	var3(t-1)	var4(t-1)	var5(t-1)	var6(t-1)	var7(t-1)	var8(t-1)	var9(t-1)	var9(t)
1	57.0	12.0	51.0	-8.727273	2.333828	-25.617436	26.567282	2270.0	11.0	4.0
2	8.0	4.0	8.0	-6.250000	5.467862	-31.723175	21.186475	2180.0	4.0	24.0
3	100.0	24.0	100.0	-8.308333	4.215596	-28.208446	28.936046	2200.0	24.0	15.0
4	40.0	15.0	40.0	-8.566667	4.956325	-20.496478	15.263329	2280.0	15.0	3.0
5	5.0	3.0	5.0	-6.666667	4.269421	-28.772233	24.861100	2230.0	3.0	12.0

Figure 4.16: Time Series Data for Supervised Learning Problem.

For a univariate one-to-one architecture, the input is a single variable with a single output component (y).

4.3.5 Reshaping the Data for LSTM

LSTMs require the shape of the inputs to be in the following format: [sample size, time step, number of features]. For the univariate implementation, only one feature or independent variable is considered while in the multivariate case at least two other time series variables may be considered. Reshaping the data is achieved using Python's NumPy reshape function. The selected configuration of the time step is discussed in Chapter 5.

Chapter 5

Implementation

This chapter implements the univariate and multivariate models of ARIMA and LSTM. The steps undertaken to develop the models, and their results are discussed. In total, ten experiments were conducted. Table 5.1 provides the details of each experiment's design and the notation used to refer to each of them in the rest of the thesis. The experiments vary in terms of model, data transformations, and explanatory variables used.

Table 5.1: Experiments summary.

Ex no.	**Model description**	**Notation used**	**Variables**	**Total number of variables**
1	Naïve	Naïve_1	Social unrest	1
2	Univariate ARIMA	ARIMA_2	Social unrest	1
3	Univariate ARIMA for a Box-Cox transformed series	ARIMA_3	Social unrest	1
4	Univariate LSTM	LSTM_4	Social unrest	1
5	Multivariate ARIMA	ARIMA_5	Social unrest, AvgTone, ActionGeo_Lat, ActionGeo_Long, NumMentions, NumArticles, NumSources, GoldsteinScale	7
6	Multivariate ARIMA for a Box-Cox transformed series	ARIMA_6	Social unrest, AvgTone, ActionGeo_Lat, ActionGeo_Long, NumMentions,	7

			NumArticles, NumSources, GoldsteinScale	
7	Multivariate ARIMA all variables	ARIMA_7	Includes actor and other event variables mentioned in Section 4.	43
8	Multivariate ARIMA for a Box-Cox transformed series all variables	ARIMA_8	Includes actor and other event variables mentioned in Section 4.	43
9	Multivariate LSTM	LSTM_9	Social unrest AvgTone, ActionGeo_Lat, ActionGeo_Long, NumMentions, NumArticles, NumSources, GoldsteinScale	7
10	Multivariate LSTM all variables	LSTM_10	Includes actor and other event variables mentioned in Section 4.	43

5.1 Forecast Strategy

All models are evaluated using one-step time series forecasting. For example, given the observed social unrest over the past five days in Table 5.2 below, a one-step forecast would make a forecast at time step six only. In contrast, a multi-step strategy makes a forecast for more than one time step e.g., the next two days, that is time step six and seven.

Table 5.2: Sample social unrest data.

Time	**Social unrest**
1	218
2	324
3	105
4	63
5	68

The forecasting models are developed using the training set consisting of $(n - 173)$ observations, where n is the length of the time series (that is, 6209 observations/days). One hundred and seventy three forecasts are produced one time step at a time, and their accuracy is compared to the actual values not used in developing the forecasting model. For all models, actuals are fed back to the test set and used as input for the next forecast. For ARIMA, the model was refit with the new observation and fixed parameters p, d and q. For LSTM, the model was not refit due to extensive computation requirements for this step.

5.2 Experiments

5.2.1 The Naïve Model

The primary goal of the thesis is to predict social unrest events in South Africa. This objective will be achieved provided that the selected model produces a skilful forecast. A model is skilful if it produces prediction results that perform better than a naïve model. In this thesis a naïve model simply uses the last observation $(t - 1)$ as the prediction for next observation t. The predictions for the naïve model were obtained by applying the pandas (a python library for data transformation) lag shift operator function. A lag shift of one was used. Table 5.3 depicts a sample of the predictions of the naïve model on the test set.

Table 5.3: Example naïve approach results.

Actual	**Predictions**
218	247
318	218
247	318
90	247

5.2.2 Univariate ARIMA Formulation

A univariate time series consists of a set of observations on a single variable y, in the univariate experiment the variable is social unrest. If there are T observations, they may be denoted as $y_t, t = 1, 2, \dots, T$.

In the ARIMA models, the parameters p, d and q need be defined properly. This thesis implements the auto.arima function in R from the "forecast" library. The auto.arima function conducts a grid search over possible models and selects the best model order (p, d, q).

Using auto.arima, the optimal p, d and q parameters were calculated for each time series and are shown in Table 5.4.

Table 5.4: Univariate ARIMA parameters.

Ex	METHOD	Model
2	Univariate ARIMA_2	ARIMA (5,1,1)
3	Transformed Univariate ARIMA_3	ARIMA (1,1,1) $\lambda = 0.247$

The selected model for experiment 2 was ARIMA(5,1,1) with the untransformed time series and for experiment 3 ARIMA $(1, 1, 1)$ with the Box-Cox transformed time series. The link to the GitHub repo containing the data and code to reproduce the analysis has been provided in appendix D.

5.2.3 Univariate LSTM Formulation

Keras Deep learning library was used to build the LSTM models. The initial hyper-parameters were chosen based on Smith et al. (2017) initial configuration and improved on using the grid search approach. Table 5.5 shows the hyper-parameters used. Further, the univariate model only consisted of one LSTM layer, one feed-forward layer, and an output layer.

Table 5.5: Univariate LSTM hyper-parameters.

Hyper-parameters	**Value**
Activation function	Relu
Optimizer	ADAM
Learning rate	0.001
Dropout rate	None
Batch size	100
Hidden units size	512
Epoch	100
Loss	MSE
Look-back window	28
Time step	1

5.2.4 Multivariate ARIMA Formulation

Similar to the univariate approach, auto.arima was used to obtain the optimal model parameters $p.d$ and q. The optimal p, d and q parameters were calculated for each multivariate time series and shown in Table 5.6.

Table 5.6: Multivariate ARIMA parameters.

Ex	**Method**	**Model**
5	Multivariate ARIMA_5	ARIMA $(0,1,3)$
6	Multivariate ARIMA_6 transformed	ARIMA$(2,1,1)_7$ $\lambda = 0.822$
7	ARIMA_7 all variables	ARIMA(5,1,1)
8	ARIMA_8 transformed all variables	ARIMA(2,1,2) $\lambda = 0.822$

5.2.5 Multivariate LSTM Formulation

In total, two Multivariate LSTM models were trained. Like the univariate LSTM, the initial hyper-parameters were chosen based on Smith et al. (2017) initial configuration and offered an improvement on using a grid search approach. The hyper-parameters that produced the lowest error rates over the test set were selected. The final multivariate LSTM model hyper parameters are shown in Table 5.7 below.

Table 5.7: Multivariate LSTM parameters.

Hyper-parameters	**Value**
Activation function	Relu
Optimizer	ADAM
Learning rate	0.0001
Dropout rate	0.2
Batch size	100
Hidden unit sizes	512
Epoch	500
Loss	MSE
Look-back window	Full data set
Time step	1

In order to overcome overfitting in the multivariate LSTM models, a dropout of fraction 0.2 was added to the LSTM layer. The models only consisted of one LSTM layer and an output layer.

5.3 Experimental Results Univariate Models

The best model is chosen based on the minimum RMSE, MAE, and SMAPE. In Figure 5.1 and Table 5.7, the RMSE, MAE, and SMAPE for each of the univariate models are described.

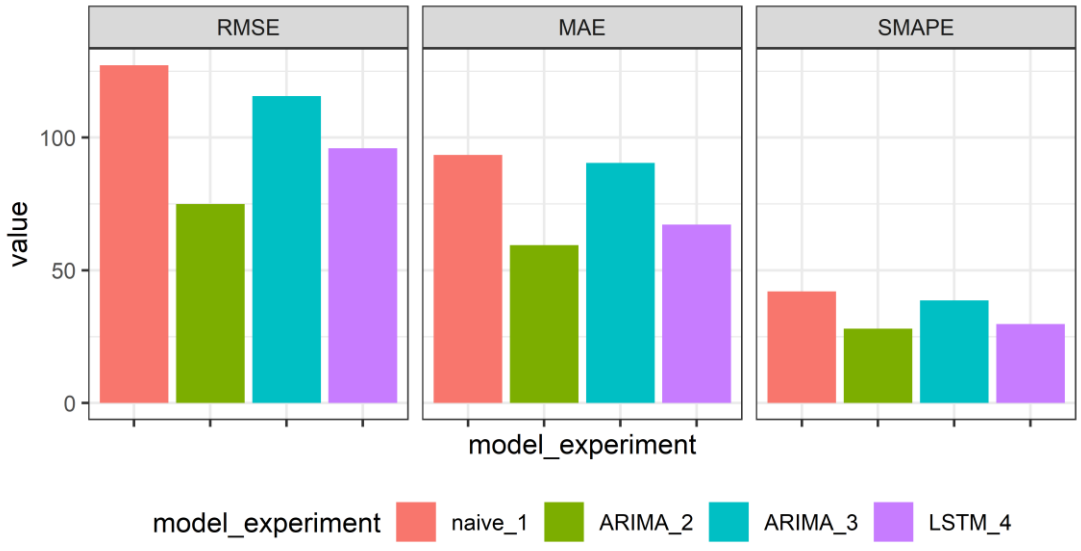

Figure 5.1: Univariate experiment results.

Table 5.8: Univariate model results (RMSE MAE and SMAPE).

Model experiment	RMSE	MAE	SMAPE (%)
naïve_1	127.247	93.416	41.982
ARIMA_2	74.943	59.466	27.994
ARIMA_3	115.633	90.468	38.631
LSTM_4	95.919	67.245	29.755

Figure 5.1 and Table 5.8 show that the univariate ARIMA (experiment two) outperforms the naïve model and LSTM across all metrics. These results suggest that LSTMs are not always better than baseline methods. However, the LSTM model does perform better than the naïve method across all metrics. LSTM also performs better than the transformed ARIMA time series (experiment three). The results also highlight that applying Box-Cox transformation did not improve the model performance.

5.4 Experimental Results Multivariate Models

This thesis hypothesised that including additional variables to explain variation in the time series would improve forecast performance. Figure 5.2 and Table 5.9 are the results of the multivariate experiments. Recall that ARIMA_5, ARIMA_6, and LSTM_9 use only the standard variables of: AvgTone, ActionGeo_Lat, ActionGeo_Long, NumMentions, NumArticles, NumSources, GoldsteinScale. ARIMA_7 and LSTM_10 incorporate additional variables, the actors, and other events.

The results reveal that LSTMs are better than ARIMA at handling the complexities associated with a multivariate time series. Both LSTM models outperform ARIMA. The two ARIMA models (ARIMA_5 and ARIMA_7) without Box-Cox transform do not perform better than the naïve approach. The transformed ARIMA models perform better than the naïve approach, and there is only a slight difference when additional variables are included. There is a noticeable improvement in the performance of LSTM when additional variables are included across all performance metrics.

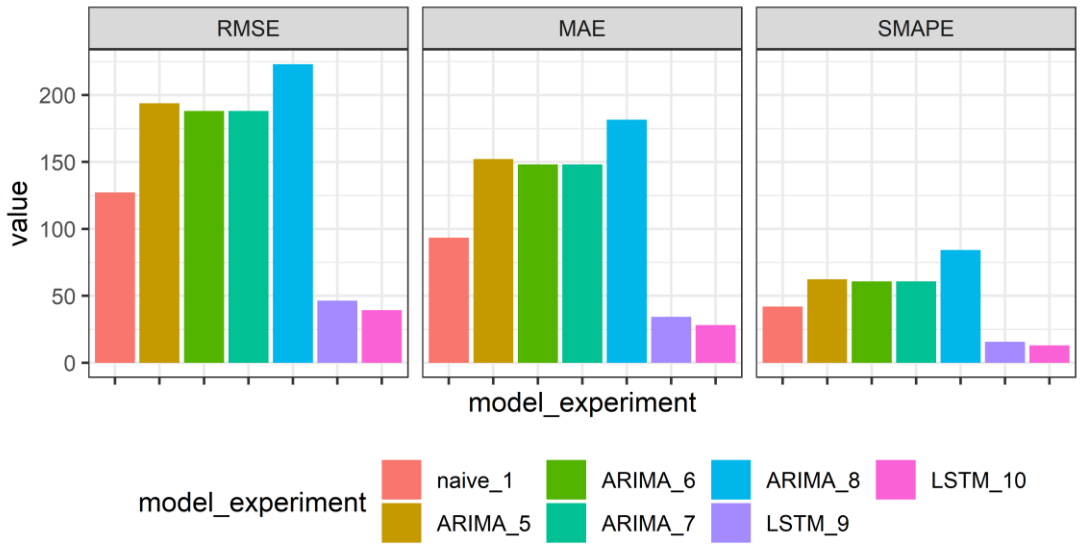

Figure 5.2: Multivariate experiment results.

Table 5.9: Multivariate experiment results.

Model experiment	**RMSE**	**MAE**	**SMAPE (%)**
naïve_1	127.247	93.416	41.982
ARIMA_5	193.831	152.258	62.317
ARIMA_6	85.060	69.949	31.691
ARIMA_7	188.183	148.066	60.817
ARIMA_8	86.601	69.747	31.556
LSTM_9	46.490	34.278	15.618
LSTM_10	39.279	28.261	12.926

5.5 Prediction Results

To get a better understanding of the quality of the predictions for each experiment, the predictions are presented and compared to the true values in the plots below.

5.5.1 Univariate predictions

Figures 5.3-5.5 are the predictions of the univariate ARIMA and univariate LSTM models. The plots on the left are the time series of the predictions plotted together with the true values. The plots on the right are the actual values plotted against the predictions. The predictions of the best performing univariate model ARIMA_2, show a stronger

relationship with the actual values (Figure 5.3 right). The LSTM predictions are slightly more scattered around the regression line. The transformed ARIMA series (ARIMA_3) does not register the peaks of the original time series well in comparison to ARIMA_2. Transforming the time series assumed a constant mean and a normally distributed time series, which is not the case. The normality assumption is restrictive and simplifies the true nature of the time series by quieting the random effects.

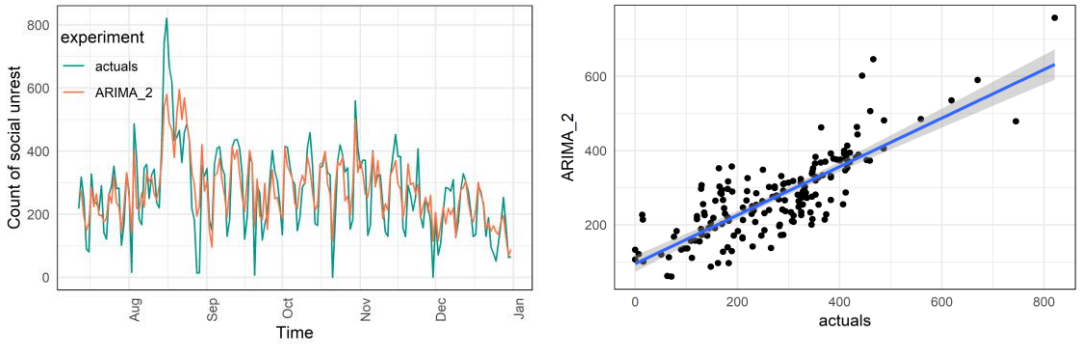

Figure 5.3: Social unrest predictions ARIMA_2 (univariate model without Box-Cox transformation).

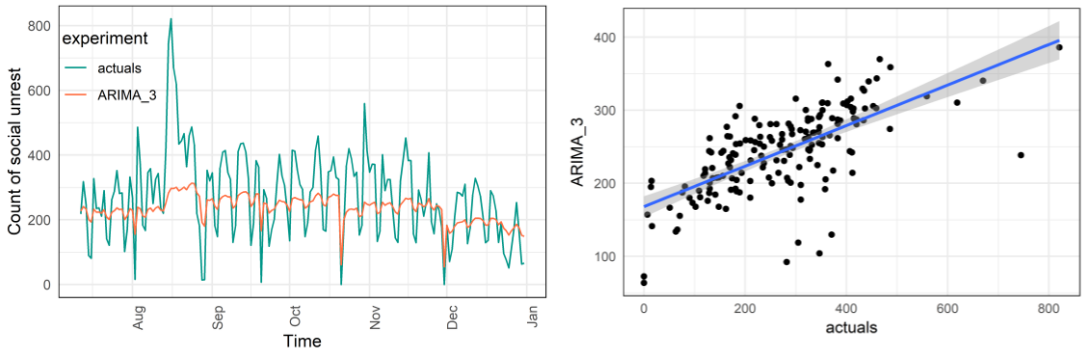

Figure 5.4: Social unrest predictions ARIMA_3 (univariate model with Box-Cox transformation).

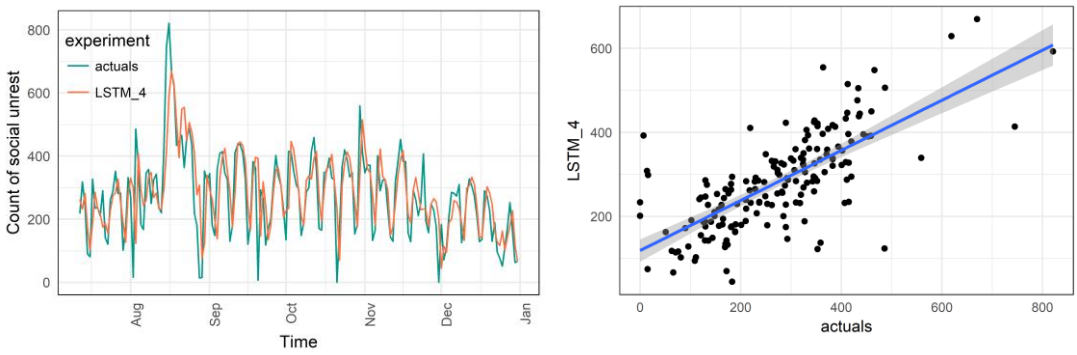

Figure 5.5: Social unrest predictions LSTM_4 (univariate LSTM model).

5.5.2 Multivariate predictions

Figures 5.6-5.11 describe the predictions for the multivariate experiments. The plots reveal that multivariate ARIMA models without Box-Cox transformation (ARIMA_5 and ARIMA_7) do not follow the trend of the original time series. The predictions appear to be more random (Figure 5.6 and 5.8 right). ARIMA_6 and ARIMA_8 follow the trend of the time series even much better than the univariate model. However, the transformed time series underestimates the peaks and dips of the original time series. LSTM_9 and LSTM_10 show a similar performance with LSTM_10 being slightly better than LSTM_9 at detecting the peaks.

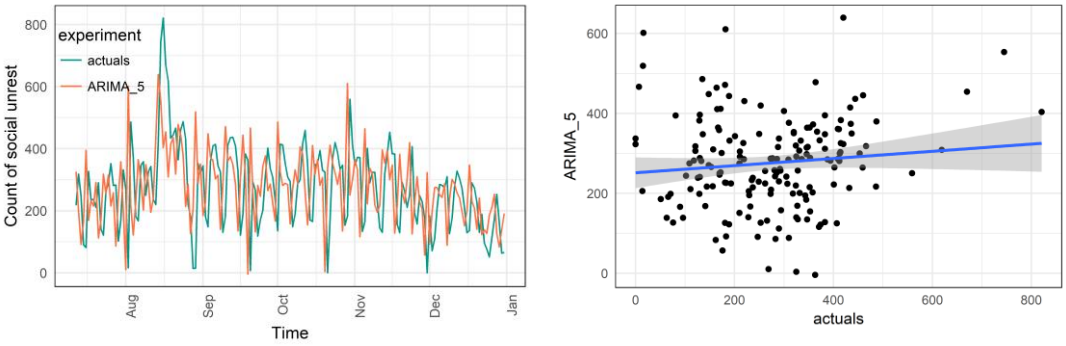

Figure 5.6: Social unrest predictions ARIMA_5 (multivariate model without Box-Cox transformation).

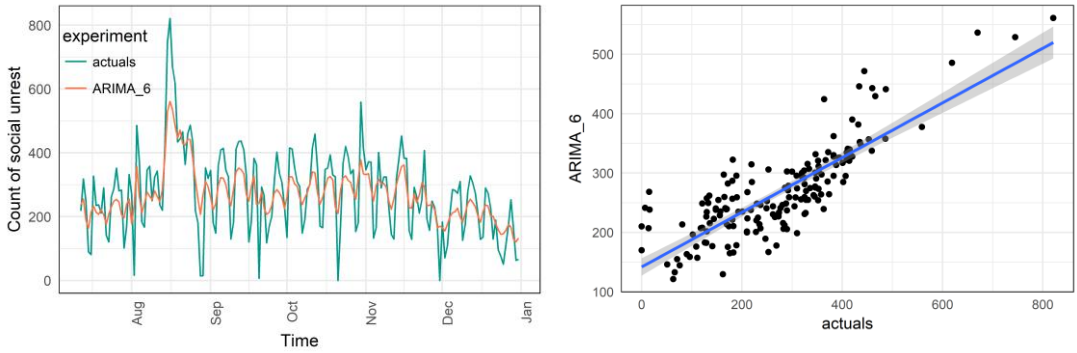

Figure 5.7: Social unrest predictions ARIMA_6 (multivariate model with Box-Cox transformation).

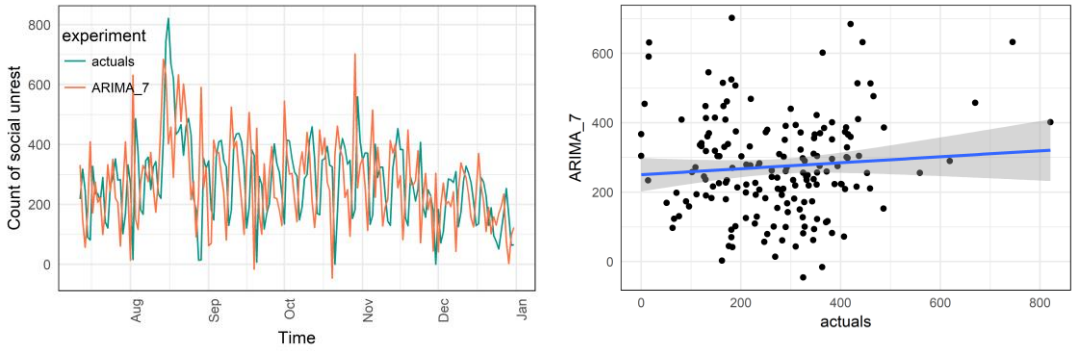

Figure 5.8: Social unrest predictions ARIMA_7 (multivariate model with additional variables and without Box-Cox transformation).

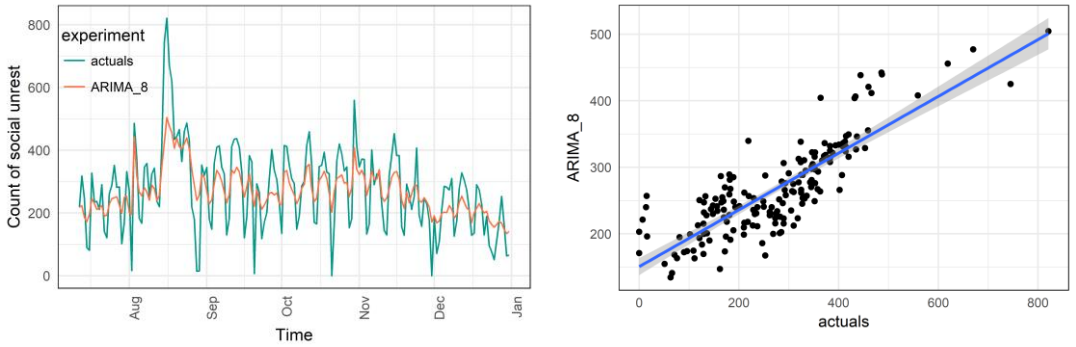

Figure 5.9: Social unrest predictions ARIMA_8 (multivariate model with additional variables and with Box-Cox transformation).

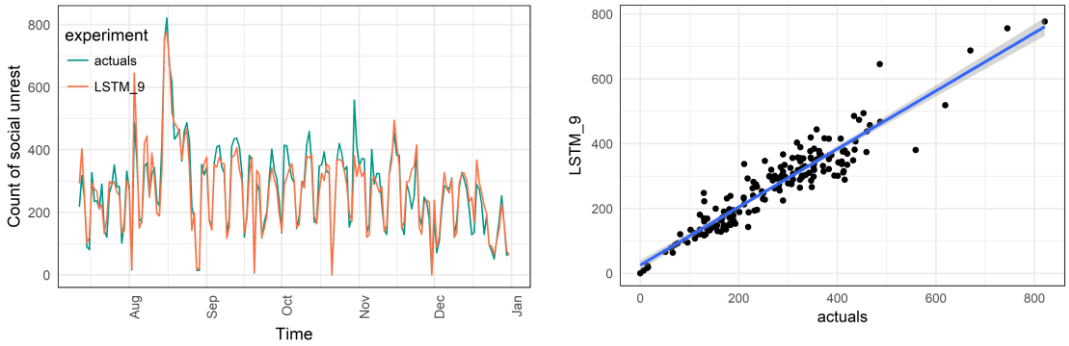

Figure 5.10: Social unrest predictions LSTM_9 (multivariate model with standard).

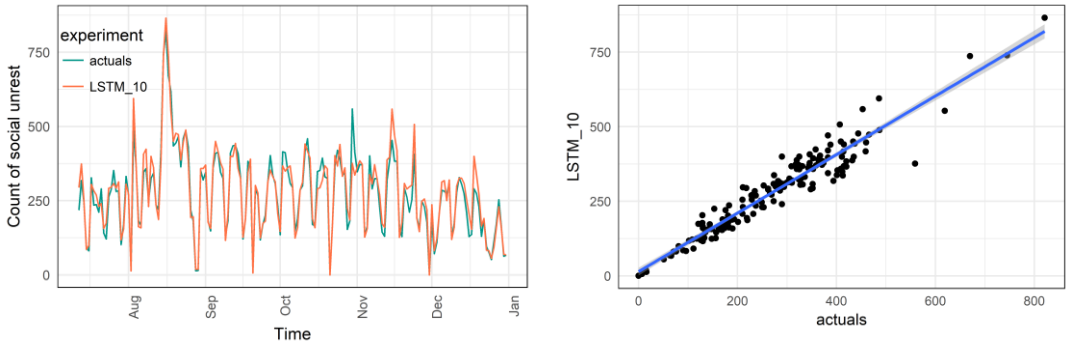

Figure 5.11: Social unrest predictions LSTM_10 (multivariate model with additional variables).

In conclusion, the experiments demonstrated that a skilful forecast of social unrest can be achieved with both ARIMA and LSTM. The results of the multivariate analysis are more consistent with the literature and highlight the power of LSTM for multivariate time series forecasting over ARIMA. LSTMs significantly outperformed ARIMA for more complex multivariate time series in all forecast metrics. Further, univariate ARIMA was more suited for simple one-step univariate time series forecasting. In the multivariate experiments, Box-Cox transformation was necessary for ARIMA to produce a skilful forecast. Further, incorporating additional features improved model performance for the LSTM method, though only marginally. Incorporating these additional features had near to no impact on the Box-Cox transformed ARIMA model and deteriorated the performance of the ARIMA model with no transformation.

To verify the LSTM implementation process described in this chapter, a replication of Smith et al. (2017)'s study is described in Appendix E.

Chapter 6

Discussion

6.1 Conclusion

The primary purpose of this thesis was to predict the volume of social unrest events in South Africa using GDELT data. Therefore, the effectiveness of LSTM neural networks for social unrest time series forecasting was investigated. Ten experiments were conducted, and the performance of the LSTM models were compared to the baseline ARIMA models and a naïve method. The best found performing model across all experiments was the multivariate LSTM, with all additional features included.

The ten models were evaluated on the following performance measures: RMSE, MAE, and SMAPE. The best performing model across all experiments was a one-step multivariate LSTM with all explanatory variables included. The best performing baseline method was a one-step univariate ARIMA model where no transformations were applied to the time series. The performance difference in the evaluation matrices are shown in Table 6.1 and are in favour of the LSTM method:

Table 6.1: Best performing ARIMA vs best performing LSTM across all experiments.

RMSE	**MAE**	**SMAPE**
47.59%	52.48%	53.82%

The significant improvements in the performance measures reveal that multivariate LSTMs were the more suited approach and significantly

better than univariate and multivariate ARIMA at learning complex time series relationships for the research problem.

Table 6.2 shows the performance difference when additional variables (actor and other events) were included in the time series for the LSTM experiments (LSTM_9 and LSTM_10). The results suggest that the inclusion of actor and other event variables improve forecast performance for the LSTM approach.

Table 6.2: Performance improvement by including other variables.

RMSE	**MAE**	**SMAPE**
15.51%	18.64%	17.60%

The multivariate ARIMA models where no Box-Cox transformations were applied were ineffective at registering the trend of the original time series. These experiments were outperformed by the naïve method. Box-Cox transformation was necessary to reduce complexity of the multivariate ARIMA models. The Box-Cox transformed ARIMA model outperformed the naïve approach. Therefore, ARIMA models did not do well with added complexity.

LSTM's models can be used to predict the volume of social unrest events in South Africa. Anticipating social unrest activities before they occur has benefits to both the public and government. The ability to forecast violent strikes can provide the government with the ability to formulate more effective mitigation plans, mobilize resources, coordinate responses and prioritize citizen grievances (Qiao, 2017). The public can be made aware beforehand on the areas to avoid. Being able to anticipate these events can assist decision makers in making more effective public policies.

6.2 Future Work

The experiments conducted in this thesis showed that a multivariate LSTM model produced much better results than a univariate model. However, the feature selection stage was a black-box approach, and all features were assumed relevant and included in the model training step. There is no clear understanding as to what features were the most important to the prediction problem. It will be worthwhile to explore model performance using automated feature selection processes that can suggest the most important features. One such method is a hybrid approach of a convolutional neural network (CNN) and LSTM.

CNNs are well known for efficient learning of the feature space. This capability will be highly favourable to a problem of this nature where long sequences are used to predict the next time step. Another automated method is the use of LSTM auto-encoders as a dimension reduction technique to reduce long sequences. Auto-encoders can produce non-linear representations of the data.

Given that social media data remains popular in the domain of social unrest forecasting, the study may also benefit from the inclusion of other data such as network measures based on interactions on social media.

References

Adhikari, R. and Agrawal, R.K. (2013). An Introductory Study on Time Series Modeling and Forecasting. Journal of Examples 2 (1): 21–23.

Asur, S. and Huberman, B. (2010). *Predicting the Future with Social Media*. Proceedings - 2010 IEEE/WIC/ACM International Conference on Web Intelligence, [online] WI 2010. 1. 10.1109/WI-IAT.2010.63. Available at: https://www.researchgate.net/publication/45909086_Predicting_the_Future_with_Social_Media [Accessed 8 Apr. 2017].

Bi, J., Feng, T. and Yuan, H. (2018). *Real-time and short-term anomaly detection for GWAC light curves*. Computers in Industry Volume 97, May 2018, Pages 76-84, [online]. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0166361517303287?via%3Dihub [Accessed 12 Nov. 2017].

Box, G. E. P. and Cox, D. R. (1964). *An analysis of transformations*. Journal of the Royal Statistical Society, Series B, 26, 211-252.

Chatfield, C. (2003). *The Analysis of Time series: An introduction*. Chapman and Hall 6th Edition.

Chollet, F. and Allaire, J.J. (2015). *Keras*. Github, [online]. Available at: https://github.com/fchollet/keras [Accessed 02 Nov. 2017].

Compton, R., Lee, C., Xu, J., Artieda-Moncada, L., Lu, T., Silva, L. and Macy, M. (2014). *Using Publicly Visible Social Media to Build Detailed Forecasts of Civil Unrest*. Security Informatics, [online] 3. 4. 10.1186/s13388-014-0004-6. Available at: https://www.researchgate.net/publication/274491240_Using_publicly_visible_social_media_to_build_detailed_forecasts_of_civil_unrest [Accessed 10 Apr. 2017].

Damle, C. (2005). *Flood Forecasting Using Time Series Data Mining*. University of South Florida, [online]. Available at: https://scholarcommons.usf.edu/cgi/viewcontent.cgi?article=3843&context=etd [Accessed 10 Apr. 2017].

Fischer, T. and Krauss, C. (2018). *Deep learning with long short-term memory networks for financial market predictions*. European Journal of Operational Research Volume 270, Issue 2, 16 October 2018, Pages 654-669, [online]. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0377221717310652?via%3Dihub [Accessed 18 Nov. 2018].

Ford, T.E., Colwell, R.R., Rose, J.B., Morse, S.S., Rogers, D.J. and Yates, T.L. (2009). *Using Satellite Images of Environmental Changes to Predict Infectious Disease Outbreaks.* Emerging Infectious Diseases, [online] 15 (9): 1341–6. Available at: https://www.ncbi.nlm.nih.gov/pubmed/19788799 [Accessed 10 Apr. 2017].

Froude, M.J and Petley, D. (2018). *Global fatal landslide occurrence from 2004 to 2016*. Department of Geography, University of Sheffield, Sheffield, S10 2TN, UK, [online]. Available at: https://nhess.copernicus.org/articles/18/2161/2018/ [Accessed 18 Apr. 2018].

Hammond, J. and Weidmann, N.B. (2014). *Using Machine-Coded Event Data for the Micro-Level Study of Political Violence*. Research & Politics, [online] 1–8. Available at: https://journals.sagepub.com/doi/full/10.1177/2053168014539924 [Accessed 6 May 2017].

He, K., Zhang, X., Ren, S. and Sun, J. (2015). *Delving Deep into Rectifiers: Surpassing Human-Level Performance on Imagenet*

Classification. [online] arXiv:1502.01852. Available at: https://arxiv.org/abs/1502.01852 [Accessed 6 May 2017].

Hyndman, R.J. (2017). *forecast: Forecasting functions for time series and linear models. R package version 8.2,* [online]. Available at: http://pkg.robjhyndman.com/forecast [Accessed 06 Nov. 2017].

Hyndman, R.J. and Khandakar, Y. (2008). *Automatic time series forecasting: the forecast package for R*. Journal of Statistical Software, 26(3): 1-22, [online]. Available at http://www.jstatsoft.org/article/view/v027i03 [Accessed 10 Dec. 2017].

Inzaugarat, E. (2018). *Understanding Neural Networks: What, How and Why?*. Towards Data Science, [online]. Available at: https://towardsdatascience.com/understanding-neural-networks-what-how-and-why-18ec703ebd31

Jain, A.K., Mao, J. and Mohiuddin, K. (1996). *Artificial Neural Networks: A Tutorial*. IEEE Computational Science & Engineering, 29 (3).

Jason, B. (2017). Machine Learning Mastery: White Noise Time Series with Python. [online]. Available at: https://White Noise Time Series with Python.com/white-noise-time-series-python [Accessed 10 Apr. 2018].

Jason, B. (2018). Deep Learning for Time Series Forecasting Predict the Future with MLPs, CNNs and LSTMs in Python. (2018). Edition: v1.2.

Jason, B. (2019). Machine Learning Mastery: A Gentle Introduction to the Rectified Linear Unit (ReLU). [online]. Available at: https://machinelearningmastery.com/rectified-linear-activation-

function-for-deep-learning-neural-networks/ [Accessed 10 May. 2019].

Korkmaz, G., Cadena, J., Kuhlman, C.J., Marathe, A., Vullikanti, A. and Ramakrishnan, N. (2015). *Combining Heterogeneous Data Sources for Civil Unrest Forecasting*. In Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM '15), [online] Paris, France, 258–65. Available at: http://people.cs.vt.edu/naren/papers/multiple_data_sources_cu_asonam15.pdf [Accessed 16 Jun. 2017].

(2013). *GDELT – Data Format Codebook V 1.03*. [online]. Available at: http://data.gdeltproject.org/documentation/GDELT-Data_Format_Codebook.pdf [Accessed 2 Jan. 2017].

Leetaru, K. and Schrodt, P.A. (2013). *The: Global Data on Events, Language, and Tone, 1979-2012*. International Studies Association Annual Conference, [online] San Francisco, CA. Available at: http://data.gdeltproject.org/documentation/ISA.2013.GDELT.pdf [Accessed 20 Aug. 2017].

Lipton, Z.C., Kale, D.C., Elkan, C. and Wetzel, R. (2017). *Learning to Diagnose with Lstm Recurrent Neural Networks*. Conference Paper at ICLR, [online] v7. Available at https://arxiv.org/abs/1511.03677 [Accessed 6 May. 2018].

McKinney, W. (2010). *Data structures for statistical computing in python*. Proceedings of the 9th Python in Science Conference, Volume 445, 2010, pages 56 – 61.

Nami, Z., Misman, O., Erbil, A. and May, G.S. (1997). Semi-empirical neural network modeling of metal-organic chemical vapor deposition," in IEEE Transactions on Semiconductor Manufacturing, vol. 10, no. 2, pp. 288-294, May 1997, doi: 10.1109/66.572084.

Panigrahi, S. and Behera, Dr. H. (2017). *A hybrid ETS-ANN model for time series forecasting*. Eng. Appl. of AI. 66. 49-59. 10.1016/j.engappai.2017.07.007.

Qiao, F., Li, P., Zhang, X., Ding, Z., Cheng, J. and Wang, H. (2017). *Predicting Social Unrest Events with Hidden Markov Models Using GDELT*. Discrete Dynamics in Nature and Society 2017, [online] (8180272): 13 pages. Available at: https://www.hindawi.com/journals/ddns/2017/8180272/ [Accessed 6 May 2017].

Revelsa, S., Kumara, S.A.P. and Ben-Assulib, O. (2017). *Predicting obesity rate and obesity-related healthcare costs using data analytics*. Health Policy and Technology, Volume 6, Issue 2, Pages 198-207.

Rudorfer, G. (1997). A Very Short Introduction to Artificial Neural Networks. [online]. Available at: https://rudorfer.homedns.org/apl94/node3.html [Accessed 20 Jan. 2018].

Schrodt, P.A. (2012). *CAMEO Conflict and Mediation Event Observations Event and Actor Codebook*. Pennsylvania State University, [online] Available at: https://www.gdeltproject.org/data/documentation/CAMEO.Manual.1.1b3.pdf [Accessed 2 Jan. 2017].

Smith, E., Smith, J., Legg, P. and Francis, S. (2017). *Predicting the Occurrence of World News Events Using Recurrent Neural Networks and Auto-Regressive Moving Average Models*. Advances in Intelligent Systems and Computing, [online], 650. 191-202. 10.1007/978-3-319-66939-7_16. Available at: https://www.researchgate.net/publication/319491803_Predicting_the_Occurrence_of_World_News_Events_Using_Recurrent_Neural_Netwo

rks_and_Auto-Regressive_Moving_Average_Models/citation/download [Accessed 2 Nov. 2017].

Singh, S. and Pal, R. (2018). *Characterizing and Detecting Social Outrage on Twitter: Patel Reservation in Gujarat*. In: Panda B., Sharma S., Roy N. (eds) Data Science and Analytics. REDSET 2017. Communications in Computer and Information Science, vol 799. Springer, Singapore.

The pandas development team. (2020). *Pandas, version latest*. Zenodo, [online]. Available at: https://doi.org/10.5281/zenodo.3509134 [Accessed 8 July. 2017].

Tom, M. (1997). Machine Learning, McGraw Hill. pp.97-123.

Tumasjan, A., Sandner, T.O. Sprenger, P.G. and Welpe, I.M. (2010). *Predicting Elections with Twitter: What 140 Characters Reveal About Political Sentiment.* Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media, [online]. Available at: https://www.aaai.org/ocs/index.php/ICWSM/ICWSM10/paper/viewFile/1441/1852 [Accessed 16 June. 2017].

Wang, B., Zhang, Duo, Zhang, Duanhao., Brantingham, P.J. and Bertozzi, A.L. (2017). *Deep Learning for Real Time Crime Forecasting*. UCLA, [online], *arXiv:1707.03340*. Available at: https://arxiv.org/pdf/1707.03340.pdf [Accessed 8 July. 2017].

Wickham, H., Francois, R., Henry, L. and Müller, K. (2017). *dplyr: A Grammar of Data Manipulation*. R package version 0.7.4, [online]. Available at: https://CRAN.R-project.org/package=dplyr [Accessed 19 Aug. 2018].

Xin, G. and Byeong-Ho, K. (2018). *PRICAI 2018: Trends in Artificial Intelligence.* 15th Pacific Rim International Conference on Artificial

Intelligence, [ebook] Nanjing, China, August 28–31, 2018, Proceedings, Part II.

Yonamie, J.E. (2013). *Predicting Future Levels of Violence in Afghanistan Districts Using Gdelt. Inpress*, [online]. Available at: http://data.gdeltproject.org/documentation/Predicting-Future-Levels-of-Violence-in-Afghanistan-Districts-using-GDELT.pdf [Accessed 8 Jul. 2017].

Zang, X., Fuehres, H. and Gloor, P.A. (2011). *Predicting Stock Market Indicators Through Twitter 'I Hope It Is Not as Bad as I Fear'*. Procedia - Social and Behavioral Sciences, [online]. Available at: http://www.ickn.org/documents/COINs2010_Twitter4.pdf [Accessed 19 Aug. 2018].

Appendix A

Literature Survey

Appendix A are the summary statistics of the literature surveyed in the domain of social unrest.

Figure A1 illustrates the sequence of steps followed to collect the publications. Semi-automated means were used to gather relevant publications in the domain of social unrest events forecasting.

1. Data Collection: The first step required establishing a connection to Microsoft Azure's Bing Custom Search API and mining the search data in R.

2. Key word filtering: A series of data pre-processing and keyword extraction that insures that the publications left contain the relevant themes that are within the scope of the literature review.

3. Feature identification and classifications of articles: This is a thorough review of each relevant publication in which features such as year the paper was written, country and/or region of analysis, machine learning approaches applied, data source e.g. Twitter, Facebook, Tumblr etc were identified. Each paper was then classified into groups based on data source and machine learning techniques and provided a review within each category.

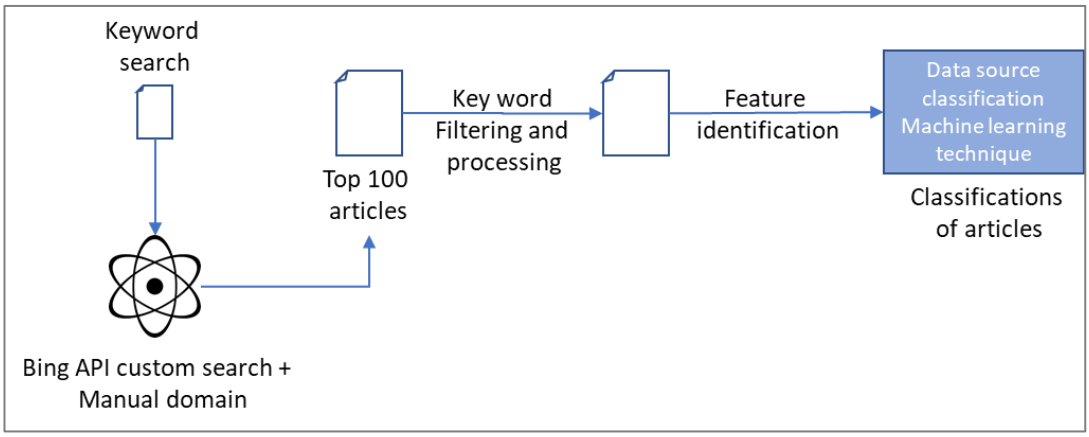

Figure A 1: Research framework for literature review.

The Bing Custom Search API allows one to customize search engine results to specific website domains. In the analysis conducted for this paper, the API was customized to survey and retrieve search data from: Research gate, arxiv and Social Science Research Network (SSRN). Using the keywords: civil unrest, protest, material conflict, social unrest, event forecasting, early warning and extremism detection, the top 100 search results from the selected domains were queried. Table A1 below is a summary of the returned search results. The table reveals most publications are found on research gate webpages (550) followed by arxiv (442) and lastly, SSRN (419).

Table A 1: Research survey results.

Domain	**Total Articles**	**protest**	**unrest**	**conflict**	**total**
Research gate	550	24	38	43	105
Arxiv	442	9	14	3	21
SSRN	419	3	0	20	22

The next step of the process was to apply keyword filters to the returned results. The Bing custom search API returns the title of the

publication and the first two lines of the abstract. Using this, automated keyword filters were applied to only obtain publications with the words *predict*, *forecast, protest, unrest* or *conflict*. Of the 550 publications found on research gate, 152 contained the themes forecast and predict. Further keyword filtering of the 152, 24 were themed around protest, 38 unrest, and 43 conflict. This gave a total of 105 articles to review further. The other webpages, arxiv and SSRN produced much less relevant content in terms of the themes of interest. Therefore, this process of applying the keyword filters (step 3 in Figure 3) automatically narrowed down analysis to 105 articles on research gate, 21 on arxiv and 22 on ssrn.

The next step in keyword filtering involved a manual verification of each publication returned. The manual verification step involved accessing the full text of each paper and reading the abstracts, data and study design sections. The analysis revealed 15 relevant publications from research gate and only two from arxiv and SSRN (Table A2). To supplement these findings, the search was extended to google scholar results as well as citations found in other related publications. This step was a manual process and produced 20 relevant publications from Google scholar.

Table A 2: Relevant publications for meta-analysis.

Domain	**Relevant Publications**
Research gate	15
arxiv	2
ssrn	2
Google scholar and other	20
Total unique	34

Table A2 reveals that a total of 34 unique papers in the area of social unrest event forecasting and event detection were surveyed. Figure A2

below reveals that the maximum number of papers were published in 2014 and 2016. Figure A2 also reveals that this forecasting problem has only recently gained the attention of researchers given that research only surfaced between the year 2010 and 2019. The 34 publications identified proceed to the next step of the research framework, feature identification through meta-analysis and classification of articles.

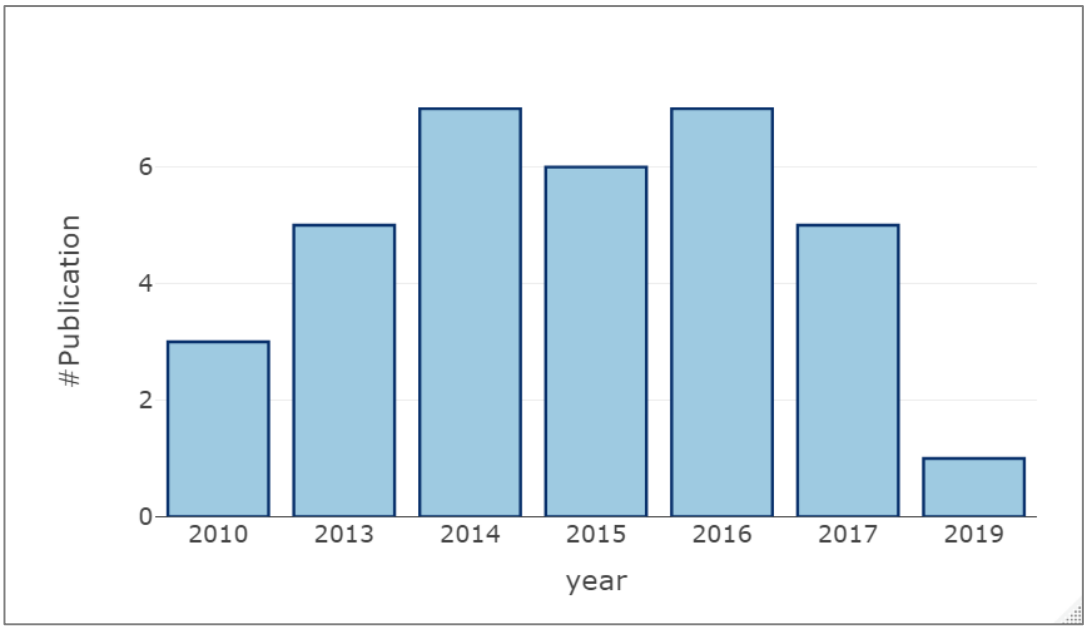

Figure A 2: Number of Publications and articles over a time period relevant to social unrest.

The y axis on Figure A3 below signify the count of each publication where the colours represent the data source distribution and the x axis, the year the paper was published. Figure A3 reveals that social media is the most widely used data source in the literature of social unrest prediction; more specifically twitter.

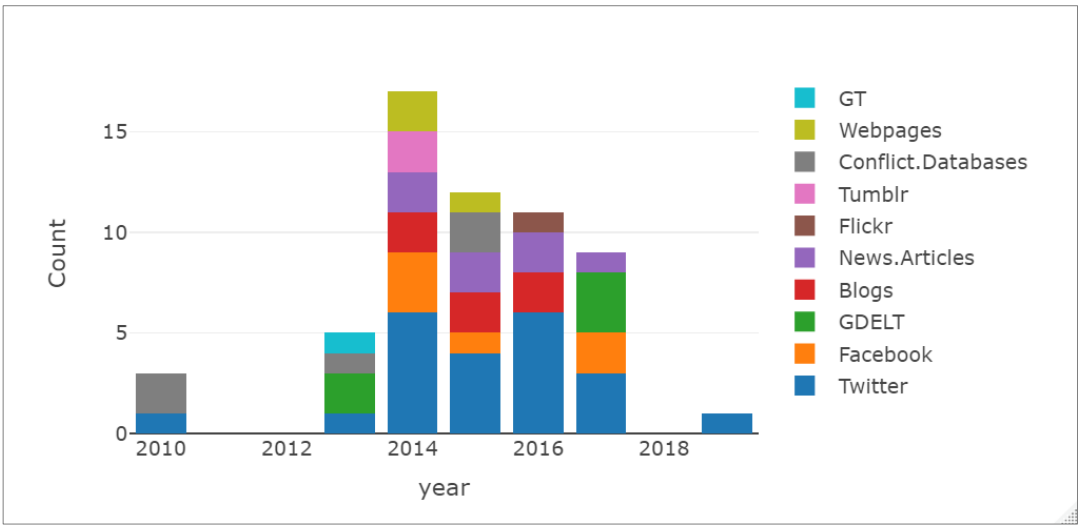

Figure A 3: Publication categorised based on data source.

Figure A4 provides grouped summary statistics of the machine learning techniques used in the literature of social unrest event prediction. Keyword Filtering, Clustering, Logistic Regression and SVM are the most commonly used techniques.

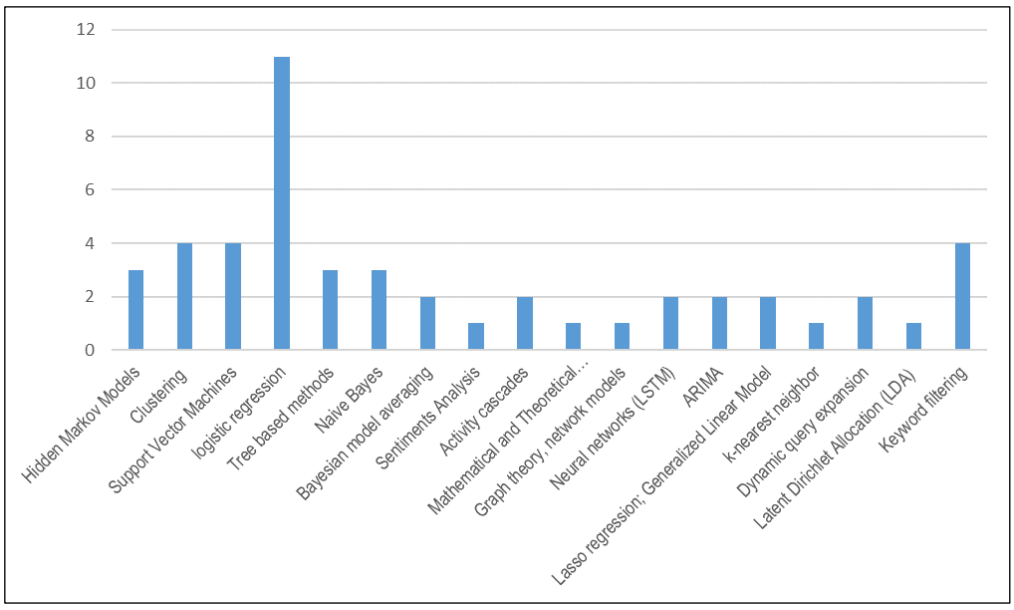

Figure A 4: Machine learning techniques summary statistics.

Appendix B

Correlation Tables

Table B 1: Actor type correlations.

Actor Type Variable One	Actor Type Variable Two	Correlation
Police_forces	Government	0.682966
Government	Police_forces	0.682966
Police_forces	Media	0.669388
Media	Police_forces	0.669388
Police_forces	Civilian	0.657399
Civilian	Police_forces	0.657399
Police_forces	Business	0.628046
Business	Police_forces	0.628046
Media	Judiciary	0.627289
Judiciary	Media	0.627289
Media	Government	0.621215
Government	Media	0.621215
Government	Business	0.609913
Business	Government	0.609913
Government	Civilian	0.608023
Civilian	Government	0.608023
Police_forces	Judiciary	0.601258
Judiciary	Police_forces	0.601258

Table B 2: Variable correlations with social unrest.

Variables	Correlation
ActionGeo_Long	0.93691662
Verbal_Conflict	0.92722238
NumMentions	0.92101424

NumArticles	0.92061166
NumSources	0.91798038
Material_Cooperation	0.91070954
Verbal_Cooperation	0.89870348
Police_forces	0.86108347
Government	0.82100897
Media	0.79134046
Judiciary	0.76952822
Civilian	0.74531538
Business	0.73374786
Criminal	0.70848966
Material_Conflict	0.66311984
Political_Opposition	0.65147653
Education	0.64079174
Military	0.5812476
Labor	0.52124181
Legislature	0.51805242
Unaligned_Armed_Forces	0.48999897
Health	0.4744745
Elites	0.44310676
Rebels	0.36237886
Multinational_Corporation	0.33061306
InterGovernmental_Organization	0.32563715
InternationalTransnational_Generic	0.30672335
State_Intelligence	0.30415702
Agriculture	0.2709496
NonGovernmental_Organization	0.20042082
Human_Rights	0.18555862
Refugees	0.16115197
Separatist_Rebels	0.15782316
International_Militarized_Group	0.15522128
Radical	0.14640637
Environmental	0.11514224

Unidentified_State_Actor	0.09237401
Settler	0.08173184
Insurgents	0.07448847
GoldsteinScale	0.03536913
ActionGeo_Lat	-0.93286846

Appendix C

Feed-forward Neural Network from Scratch

Alternatively, the code can be access via the link below:

https://github.com/samvdm/Neural-Networks/tree/master/BackpropPythonExample

In [20]:
```
def initialze_network(n_input, n_hidden, n_output):
    network = []
    hidden_layer = {'weights': [[np.random.rand(n_input,n_hidden)] , [0]]}
    network.append(hidden_layer)
    output_layer = {'weights': [[np.random.rand(n_hidden,n_output)] , [random.random()]]}
    network.append(output_layer)
    return network
#forward pass
def forward_pass(weights, inputs, bias):
    a = np.dot(weights, inputs) + bias
    return a
def out_gradient(target, output, inputs):
    gradient = (dloss(target, output))*inputs
    return gradient
def bias_gradient(target, output):
    gradient = dloss(target, output)
    return gradient
#one hidden layer back prop
def hid_gradient(target, output, inputs, a, weight):
    k = np.sum((-(target - output))*weight) #the first column of a matrix
    g = a*(1- a)*inputs
    return g*k

#Loss function
def lossfunction(target,output):
    loss = (1/2)*(target - output)**2
    return loss
```

In [21]:
```
def train_network(network, inputs, target, l, n_epoch):
    weights = [neuron['weights'] for neuron in network]
    for epoch in range(n_epoch):
        myloss = []
        h_gradients = []
        for i in range(len(inputs)):
            a_hidden = forward_pass(weights[0][0][0], inputs[i], weights[0][1])
            outputs = forward_pass(weights[1][0][0].T, a_hidden, weights[1][1])
            loss = lossfunction(target[i], outputs)
            t_loss = np.sum(loss)
            myloss.append(t_loss)
            grad = [out_gradient(target[i], outputs, item) for item in a_hidden]
            #print("this is output gradient {} for iteration {}".format(grad, i))
            gradients = np.vstack(grad).T     #output gradients
            bias_grad = bias_gradient(target[i], outputs)
            w = np.array(weights[1][0]).T
            for item in w:
                grad = hid_gradient(target[i], outputs, inputs[i], a_hidden[0], item)
                h_gradients.append(grad)
                mygrad = np.vstack(h_gradients).T
            weights[0][0] = weights[0][0] - l*mygrad
            weights[1][0] = weights[1][0] - l*gradients
            weights[1][1] = weights[1][1] - l*bias_grad
            final_loss = np.sum(myloss)
            #print('>epoch={}, error={}'.format(n_epoch, final_loss))
            return final_loss
```

Appendix D

R and python scripts

Link to Github repo for the thesis:

https://github.com/samvdm/MastersThesis

Appendix E

Predicting Material Conflict Afghanistan

To verify the LSTM implementation process described in Chapter 5, this section aims to reproduce Smith et al. (2017)'s study. The focus is to only reproduce Smith's static LSTM results as the proposed approach in this thesis is based on a static LSTM model. A static LSTM model is one in which the predictions on the test set of the data are

made on fixed parameters. The alternative to this is a “real-time” approach in which new information is presented and incorporated into the trained model and the model parameters are updated.

The data for this analysis was collected using the same process as the South African study. That is; using Google BigQuery in which all actor codes with string “AFG” Afghanistan were extracted from the database. Data was taken from the period 2001-02-01 and 2012-05-01. The aim of Smith’s study was to predict material conflict events, therefore the data was filtered based on the Quad class “material conflict”. Figure E.1 shows the count of material conflict events from 2001 to 2012.

Note in Figure E.1, there is a slight difference in the volume of the data collected in this thesis; however, the overall pattern is the same as with that in Smith’s analysis. The Afghanistan material conflict event also shows a skewed distribution. The volume of material conflict events reported is larger than that of South Africa.

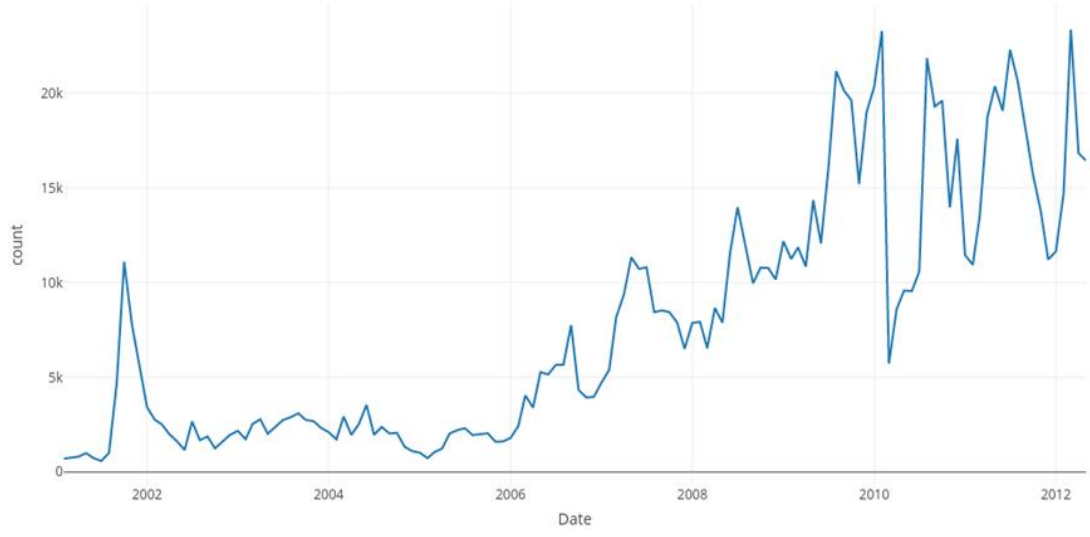

Figure E.1: Afghanistan count of material conflict events 2001 to 2012.

LSTM Forecast Results

Using Smith's source code, the model training process was replicated, and the results are discussed in this section.

Two experiments were implemented. Experiment one reproduces the results of Smith's static LSTM model in which a many to many LSTM architecture was used. Experiment two implements a many to one architecture in order to further validate that this architecture produces better forecast.

Smith took nine numerical features and aggregated them to the daily level by using an appropriate aggregation function:

1. Count for global event ID.
2. Mean for event code (expanded to fill zeros to the right), tone, Goldstein scale, latitude, and longitude.
3. Sum for mentions, sources, and articles.

Each variable was normalised using min-max to fall between -1 and 1. For the first experiment, the LSTM configuration was: two-layer stateful LSTM, 512 nodes, 50% dropout, Adam optimiser (1e-4 learning rate), fully connected layer for outputs and MSE loss function. The batch size was effectively one per time step trained at one epoch. For the second experiment, the LSTM configuration was: a single layer stateful LSTM with 128 neurons with a 50% dropout, one batch input trained at one epoch with Adam optimiser, a learning rate of 1e-4, and MSE loss function

Table E.1 presents the performance metrics results of the two experiments as well as for the naïve model. Here, the results of experiment one is within the same range as Smith's study. The results were successfully replicated validating the LSTM implementation of this thesis.

In order to get as close as possible to Smith's study, the experiments were replicated 30 times. Neural networks initialise random weights at the start of a training process, therefore Smith's results cannot be reproduced 100%.

In Table E.1, the LSTM is better than the naïve approach in both experiments. The many to one architecture produce much lower error rates than the many to many architecture.

Table E.1: Afghanistan material conflict forecast matrices.

Experiment	METHOD	RMSE	MAE	MAPE
1	LSTM baseline	3831.63	3216.03	21.01%
2	LSTM Single	1744.79	1415.85	9.9%
3	Naïve	4181.14	2911.18	24.51%

Figure E.2 depicts the predictions plotted against the actual time series. The predictions show similar LSTM behaviour to Smith's study.

Figure E.2: Afghanistan LSTM material conflict predictions vs actuals.

The findings from the experiments conducted on the Afghanistan data set appear to be consistent with that in Smith et al. (2017).