FORECASTING RETWEET COUNT DURING ELECTIONS USING GRAPH CONVOLUTION NEURAL NETWORKS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Raghavendran Vijayan

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

August 2018

Purdue University

Indianapolis, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. George Mohler, Chair

Department of Computer and Information Science

Dr. Mohammad Al Hasan

Department of Computer and Information Science

Dr. Snehasis Mukhopadhyay

Department of Computer and Information Science

Approved by:

Dr. Shiaofen Fang

Head of the Graduate Program

I am dedicating my thesis work to my parents, my sister and her extended family, teachers, and friends for all of their support.

ACKNOWLEDGMENTS

First of all, I would like to thank my advisor Dr. George Mohler for all his continuous guidance, encouragement and support in my thesis work. The thing that I admire about him the most is, he never gets annoyed, nor hesitates to answer my queries even if it's very basic, or absurd. I have always felt that I am interacting with my friend during our meetings. I can't thank you enough for creating such a great environment during the past eight months. I can say with certainty that this thesis would not be possible removing you and your efforts from the equation. I have learned a lot from you and this work about approaching a problem without fear, and being open to challenges. Such is the pleasure working with you. I am also grateful to you for offering me the teaching assistantship role for the course, "Introduction to Data Science".

Next, I would like to thank Dr. Mohammad Al Hasan and Dr. Snehasis Mukhopadhyay for your generosity in accepting my invite to serve on my thesis committee. Being a beginner in the machine learning community, I feel fortunate to be a student in the wonderful lectures of Data Mining and Intelligent Systems taught by you. They have taught me the essentials, and helped me in my research work and job search to a great extent.

A great thanks to Dr. Rajeev Raje for being a great well wisher of mine and believing in my abilities. I have learned and enjoyed a lot working under you during my teaching assistantship term for the course, "Distributed Computing". I would like to thank my other professors Dr. Yuni Xia, Dr. Mihran Tuceryan, Dr. Xukai Zou, Dr. James Hill, and Dr. Arjan Duressi for teaching me the different arenas of computer science including Databases, Big Data, Cloud Computing, Image Processing, Design of Algorithms, Software Engineering. I appreciate your time for being always open

to help me with my queries. I am thankful to Nicole, Joan, and Lori for being very kind and prompt with all administration and logistics related queries.

Last but not least, I would like to thank my father Vijayan, my mother Usharani, my sister Veena, and my brother in law Vivek for encouraging my decision to pursue higher studies abroad. I am forever grateful to my parents for all the sacrifices they have made for the well bringing of me and my sister. They have given me freedom to chase my dreams and set themselves as an example of being very patriotic, respecting everyone, staying calm and dedicated on the work, and treating the failures and success the same way. Being away from my parents for most part of my teenage days, my sister put my needs first before her own and helping me in every way. Her constant support and cheers have pushed me to succeed further. Such is the fortune of having a great sister. Staying thousands of miles away from home, I am thankful forever to my brother in law Vivek for taking care of my parents, keeping them comfortable during my absence. Even today my parents remind me of the phrase, "Amma, Appa, Guru, Kadavul", which means Mother, Father, Teacher, God in English. True to the line, my teachers starting from high school until today have been instrumental in shaping me to what I am today. The great piece of advice I have learned from them is being a lifelong student without losing curiosity in whatever I do. I take this report as an opportunity to thank my sister in law's family and my cousin staying in the United States, who have created a positive environment for me so I don't get homesick. Spending time with their kids, Ritvi and Nirosh, is the most fun I have had in the past two years. Lastly, it's time to thank my friends who have made a great influence on my life. Your encouragements, beliefs, limitless support have been the catalysts in my personal and professional development. Bearing all my insane jokes, you have always wished to see a smile on my face. I have no words to thank your kindness and support. A special thanks to you, Russell Sherman for helping me improve the quality of the report.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

ABBREVIATIONS

SEISMIC	self-exciting model for information cascades
SVM	support vector machine
SNAP	social network analysis platform
JSON	javascript object notation
UTC	coordinated universal time
ANC	african national congress
DA	democratic alliance
EFF	economic freedom fighters
GCN	graph convolution neural network
ReLU	rectified linear units
LR	linear regression
FF	feed forward neural network

ABSTRACT

Vijayan, Raghavendran. M.S., Purdue University, August 2018. Forecasting Retweet Count During Elections Using Graph Convolution Neural Networks. Major Professor: George Mohler.

A retweet refers to sharing a tweet posted by another user on Twitter and is a primary way information spreads on the Twitter network. Political parties use Twitter extensively as a part of their campaign to promote their presence, announce their propaganda, and at times debating with opponents. In this work we consider the problem of early prediction of the final retweet count using information from the network during the first several minutes after a post is made. Such predictions are useful for ranking and promoting posts and also can be used in combination with fake news detection. From a machine learning perspective, the task can be viewed as a regression problem. We introduce a novel graph convolution neural network for forecasting retweet count that combines network level features through a graph convolution layer as well as tweet level features at a higher dense layer in the network. We first will provide an overview of the graph convolution network architecture and then perform several experiments on Twitter data collected during presidential elections in South Africa, Kenya, and Nigeria. We show that the model outperforms baseline models including a feed-forward neural network and the popular point process based model SEISMIC.

1. INTRODUCTION

1.1 Background

There is no fixed definition for communication. It is a process of sending and receiving information among people, and it requires a medium to spread the information. The world has witnessed the development of science and technology and it's application in spreading information to larger groups of audiences. Thanks to the evolution of smart phones and the Internet, the competition today is how fast can the information be spread. Although it's a debatable topic, it's imperative that effective usage of social media has a great impact on our day to day lives. With the advent of social media applications like Twitter, Facebook, and others, people create and connect to new friends, spread information about what they are doing, and what they are witnessing to the rest of the world.

The popularity of any social media is in spreading the information among the connected members as fast as possible. An e-learning management company simplilearn.com [1] says more than 70 percent of people who use social media, take this medium to pass valuable information, and demonstrate their involvement in what's happening in the world. It may be supporting or fighting for a cause, discussions related to environmental changes, politics, or the news. To support the statement, a study by the Pew Research Center [2] reported two-thirds of Americans get their news updates on social media. There are many features of social media that influence the information cascade. One of them is the profile of the user. Many celebrities, politicians, and social activists have brought important news into light and have supported, debated, criticized the same. In cases, being an influencer has altered the fate of the news due to their huge fan following. There are other features specific to social media companies, including groups subscribed with people of similar interest, push

notifications, live chats, user mentions, or hashtags to ensure everyone stays updated on current topics.

1.2 Impact of Social Media During Elections

One of the three datasets that we have used in our research is the Twitter data collected during the 2014 presidential elections in South Africa. From reprobate.co.za [3], we understood that the major political parties in South Africa, i.e. African National Congress, Democratic Alliance, Economic Freedom Fighters, and Agang South Africa, have effectively used Twitter and Facebook for their campaign. It's also observed from [3], in only the eight months prior to the elections, these parties have grown their followers count over 100 percent.

The power of social media during elections has also been witnessed in the U.S. presidential elections that concluded by 2016. According to Wikipedia [4], the nominees from both the Republic and Democratic parties, Donald Trump and Hilary Clinton have effectively used this medium for their campaign. It's been observed that the former one posts 11 tweets per day and the latter one posts 5 shares per day, on average from their official Twitter handles. They have debated over several things including immigration, health care, environment, and education. On the 9th of April 2018, Facebook CEO Mark Zuckerberg announced [5] that his company is working on initiatives that helps in understanding how social media is impacting elections and results. These use cases are enough evidence to say social media has its own significance in determining the outcome of the elections.

From the stats, we can observe that content shared by, and about, the political parties and its representatives provide great value. Their news feeds and posts have an influence on target audience's voting decisions.

1.3 Problem Statement

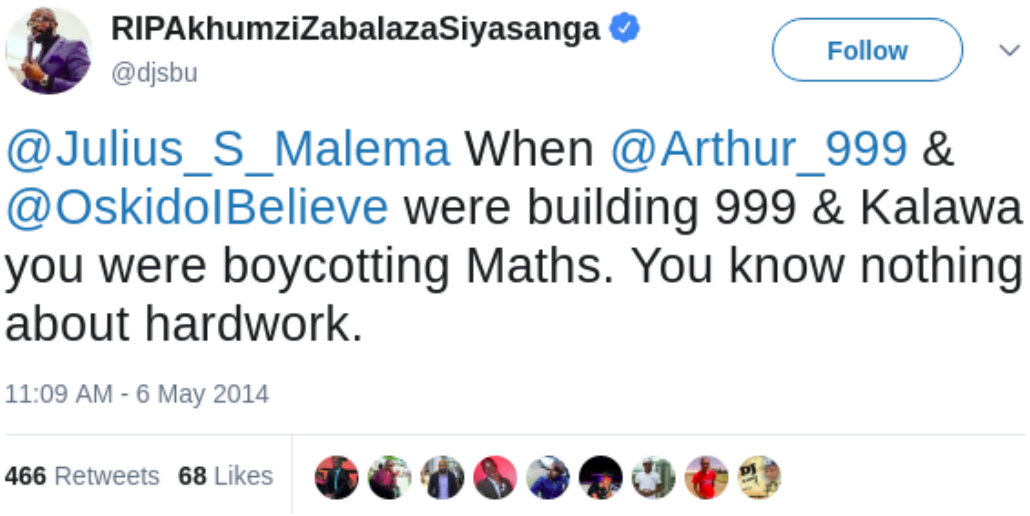

Fig. 1.1.

A sample tweet from South Africa Elections

There are 466 retweets for the tweet sequence contained in Figure 1.1. Each point in the Figure 1.2 indicates a retweet. It's posted by a verified user who is a radio jockey and philanthropist in South Africa. It has reached a 23 retweet count(marked in red color) in the first ten minutes and additional 22 retweets(marked in blue color) in the second ten minutes. Our models take into account the features corresponding to these 45 retweets, and the actors who retweeted them, and tries to estimate the final number of retweets marked in cyan color.

1.4 Contribution to this Thesis

The advancements in computing resources and the state of the art machine learning have attracted researchers in social media analytics as well. There are several studies in the past that forecasting the final number of reshares for a given post. Some researchers altered the problem statement and designed it as a classification problem. They predicted if a post was going to be a viral or not. Predicting a scalar

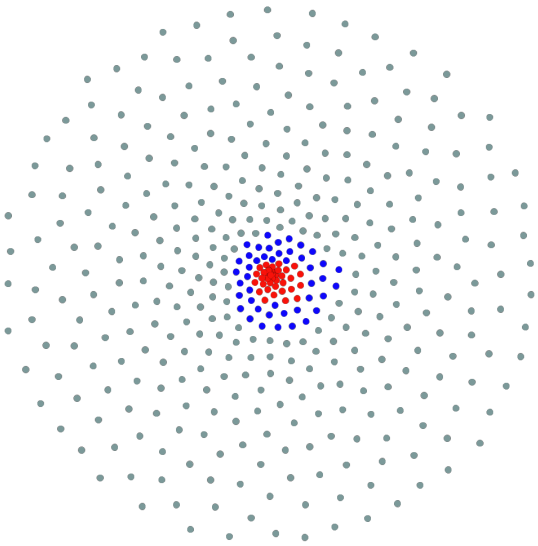

Fig. 1.2.
Information Cascade in the first twenty minutes

value is an interesting and challenging task. Most of the above mentioned studies have widely used traditional machine learning models to solve the problem. However, using deep neural network models to solve the same problem is one area that hasn't been explored as much. In this thesis, I have taken twitter data related to elections and proposed a solution to predict a final number of reshares possible for a given tweet sequence using different machine learning methods including conventional and neural network models.

1.5 Organization

In this thesis, three machine learning models are developed to predict the retweet count for a given tweet sequence. The rest of the thesis is organized as follows. Chapter 2 explains our takeaways from the existing studies similar to our problem. We have discussed the assumptions and challenges they have in constructing their models. Chapter 3 explains how tweets are represented in the dataset and the different datasets that are used in the research. Chapter 4 presents a comprehensive study

on the benchmarked model SEISMIC. Also, it covers the results that are obtained by testing the model on our datasets. Chapter 5 discusses the feature engineering, design and implementation of our three machine learning models, and the metrics to evaluate our models in detail. Chapter 6 analyses the performance of all of the constructed models on our datasets with necessary visualizations. It compares the accuracy and error scores of the different models. It contains more details about how the performance is altered by modifying layers of the neural network. Also, it contains results of other additional experiments conducted. Chapters 7 and 8 present concludes this thesis with a summary of our work and discussions on the possible future work with our research.

2. RELATED WORK

People have conducted experiments on predicting popularity of content in different social media applications including Twitter. In this chapter, we will cover how the existing studies approach solving the problem, and how their innovation keeps us motivated and challenging in our work.

Danah Boyd et al. [6] provides more details about the different conventions of retweets and how it aids in information diffusion. Sean Gransee et al. [7] assigns scores to words in the tweet and focuses on the history of retweet cascade to predict retweet count. Eytan Bakshy et al. [8] helps us understand the role of actors in spreading the information to as many users as possible, whom will be accepting it. Ethem Can et al. [9] extracts predictive features from the images attached to the tweet in addition to tweet content related and actor related features. Then, they applied linear regression, SVM, and random forest models to find retweet count. Roja Bandari et al. [10] forecasts the popularity of the news articles on social media before they are published with the help of content, context, and network properties. Balint Daroczy et al. [11] observes the relationship between the popularity of the author and the success of a particular tweet among all of his/her tweets using temporal evaluation framework. It's important to understand how one influences the other as we have both tweet related and actor related features in our models. Andreas Kanavos et al. [12] analyzes the tweet content, along with the behavior of the users to find if any of the six basic human emotions are expressed or not. Andrey Kupavskii et al. [13] differs from others by predicting retweet count for a fixed time period using the information cascade and PageRank on the retweet graph, which is a directed graph built upon users involved in the cascade.

As it's difficult to predict exact number of retweets or popularity, for a given post, researchers have defined categories to represent the expected number of retweets.

Gang Liu et al. [14] is a two phase model that combines both classification and regression problems. In the classification stage, the tweet sequences are classified into one of the seven categories related to popularity and the regression is performed on each categories. Liangjie Hong et al. [15] and Karthik Subbian et al. [16] model the problem statement as a classification problem and see if the tweet is going to be viral by taking into account temporal, content features, and network information related to the given tweet. Also, [15] predicts the volume of retweets that a given tweet may get and categorize time into four groups. Using the Bayesian network classifier, William Webberley et al. [17] classifies the tweet sequence into four groups depending on how interesting it is. They calculate interestingness using Amazon Mechanical Turk Service, that involves human intelligence.

There are variety of approaches examined by researchers to observe the information cascade and the deciding factors influencing the users to retweet. Zicong Zhou et al. [18] finds how information spreads across a larger number of users with using a friends and followers network. In addition to retweet count prediction, Andrey Kupavskii et al. [19] uses an online prediction model to find the number of users who are able to see the tweet. Given the tweet and the history of the cascade to a user, Zi Yang et al. [20] constructs factor graph model that observes the user's retweeting behaviors and decide if he/she will retweet after viewing it. Similarly Huan-Kai Peng et al. [21] proposed a solution to understand factors motivating users to retweet. It comes with a drawback that it will be expensive for huge datasets. Peng Cui et al. [22] finds the users who should share the given news so that the information will reach larger groups of audiences. Zhunchen Luo et al. [23] predicts the users who are going to retweet for a given tweet using learning-to-rank framework.

Some studies build statistical models and formula to solve the same problem instead of constructing expensive feature engineering models. The research conducted by Tauhid Zaman et al. [24] is similar to the SEISMIC model covered in the next chapter. Rather than predicting the final retweet count, their model predicts the time path that the retweet sequence may follow. They have used a Bayesian approach to

solve the problem. They take into account the time the retweet event happened, and the followers count of the actor who retweeted. [25] proposed a theoretical graph model using survival theory in which they observe how information cascades through hidden or unobserved paths.

From Jing Zhang et al. [26], it's interesting to find the likelihood of a news getting cascaded increases or decreases strongly depending on the structure of the neighbors, i.e. friends in the network. Qi Zhang et al. [27] proposed an attention based neural network model that considers the similarity of tweets and classifies whether the given tweet is positive or negative. Considering similarity forms the core of the Graph Convolution Neural network model that we have implemented in one of our models.

After reading these research papers, we have a better understanding of the different types of features that can be extracted that contributes to the success of the machine learning model. The experiments section helped us understand how to set up the optimal working environment. Many researches have used conventional machine learning regression models like Linear Regression, SVM, Random Forest, and Bayesian Approach to solve the problem. In the recent years, the amount of computational power has increased very much. It has taken Deep Learning into one step further by solving many problems [28–33] from various arena including Image Processing, Computer Vision, Natural Language Processing, and Autonomous Driving. In various use cases, Deep Learning has performed better than existing traditional models. It created curiosity in us to see whether we can apply deep learning in our application domain.

In the upcoming chapter, we will discuss in detail about the working of the benchmarked model SEISMIC. Also, we have discussed the performance of the model after testing it on our datasets.

3. DESCRIPTION OF THE DATASET

I owe a great thanks to Dr George Mohler for providing me three datasets that we used to test our models. They contain tweet sequences published during the presidential elections in Nigeria, Kenya, and South Africa.

Table 3.1.

Original Representation of Tweet

Key	**Type**	**Description**
id	String	Unique identifier to represent tweet or retweet sequence
actor	JSON	Details about the person who tweeted/retweeted that post
verb	String	A categorical variable. “post” denotes original tweets ”share” denotes retweets
generator	JSON	To identify the device that generated the post
provider	JSON	To identify the website that created the post
inReplyTo	JSON	URL to the original tweet (for retweets)
location	JSON	Geographical details of the place where the post was created
geo	JSON	Coordinates of the place where the post was created
twitter entities	JSON	List of hashtags, mentions, URL given in the posted
twitter extended entities	JSON	List of media items present in the given post
link	String	Permanent URL to the post
body	String	A 140 character or less Tweet content
object type	String	A string “activity”
object	JSON	Original tweet for retweets. Summary for original posts
posted time	String	the time the tweet was posted

The data is in JSON format. Each tweet sequence in the dataset contains values for all keys listed in Table 3.1 In general, a dataset may be multiple original tweets, multiple retweets. Table 3.2 shows the number of tweets and retweets present originally in the datasets. The rightmost column shows the count of unique tweet sequences present in the dataset.

Table 3.2.

Tweet and Retweet Sequences in Different Datasets

Dataset	**Number of Tweets**	**Number of Retweets**	**Number of Unique Tweet Sequences**
Kenya Elections	627102	508740	140521
South Africa Elections	91933	89483	22572
Nigeria Elections	792855	1958900	211691

In the coming chapter, we will discuss in detail about the working of our benchmarked model SEISMIC and it's performance.

4. SEISMIC MODEL

4.1 Introduction

SEISMIC (Self-Exciting Model for Information Cascades) [34] is a project developed by the researchers at Stanford University in 2015. In this model, they have extended the idea of self exciting process(also referred to as Hawkes Process), that is used to model earthquakes. The Social Network Analysis Platform(SNAP) library helps us in implementing the SEISMIC model in R. This model forecasts the information cascade as a self exciting point process, rather than having machine learning models. Hence, the outcome of the model, i.e. tweet popularity, can be computed with the formulas implemented in the project. Their functionalities are implemented as inbuilt functions within snap library in R language.

4.2 Working of SEISMIC

The points in this self-exciting point process represent the time and the author corresponding to that event, in this case a retweet. The self exciting keyword denotes that, one event in the point process has an influence on the other event. It means the reshare event recorded at a time in the past may pave the way to another reshare event in the future. One of the assumptions they have in implementation is that the followers of everyone who is resharing the tweet is disjoint.

Given the history of the cascade, R_t observed within time t, the model initially computes the effective (N_e^t) and normal cumulative popularity (N_t) of the tweet until time t. The popularity here represents the degree of reshares. The effective cumulative score considers the delay s that the user is taking to reshare the post from the time it started appearing in his/her newsfeed. It's referred to as memory kernel $\phi(s)$. The

normal cumulative score is the aggregated sum of the followers (n) of all the resharing actors observed until time t. The effective score is the aggregated sum of the followers count multiplied with the integral of memory kernel function over the time t. The complete algorithm is given in 1.

Algorithm 1 SEISMIC Algorithm to compute normal and effective cumulative popularity

1: **for** $i = 0, ..., R_t$ **do**
2: $\quad N_t \mathrel{+}= n_i$
3: $\quad N_t^e \mathrel{+}= n_i \int_{t_i}^{t} \phi(s - t_s)\ \mathrm{ds}$
4: **end for**

Keeping constants c, power law decay parameter θ, and s_0 at 6.27×10^{-4}, 0.242, and 5, the memory kernel $\phi(s)$ can be computed as given in 4.1.

$$\phi(s) = \begin{cases} c & \text{if } 0 < s \leq s_0 \\ c(s/s_0)^{-(1+\theta)} & \text{if } s > s_0 \end{cases} \tag{4.1}$$

Once these values are calculated, SEISMIC measures the infectiousness of the post at the time t, using the triangular weighting kernel $K_t(s)$ given in 4.2. It's chosen over other kernels for ease of computations.

$$K_t(s) = max\ \{1 - \frac{2s}{t},\ 0\},\ s > 0 \tag{4.2}$$

For all the events observed until time t, the model computes the difference in the time the tweet was reshared and the set time window, and then feeds the difference value into the weighted kernel function 4.2 that we discussed earlier. It keeps the cumulative score $\bar{R}_t$ aside and proceeds with the same set of operations, considering the memory kernel, as well, as explained in 2. The post infectiousness at the time t, p_t is obtained by dividing the effective cumulative degree of reshares $\bar{R}_t$ by the computed value $\bar{N}_t^e$.

Algorithm 2 SEISMIC Algorithm to compute infectiousness of the tweet

1: $\bar{R}_t = 0$, $\bar{N}_t^e = 0$
2: **for** $i = 0, ..., R_t$ **do**
3: $\quad \bar{R}_t \mathrel{+}= K_t(t - t_i)$
4: **end for**
5: **for** $i = 0, ..., R_t$ **do**
6: $\quad \bar{N}_t^e \mathrel{+}= n_i \int_{t_i}^{t} K_t(t - s)\phi(s - t_i)$ ds
7: **end for**
8: $p_t = \bar{R}_t / \bar{N}_t^e$

Once the infectiousness and the popularity of the tweet is identified with the help of 1 and 2, the final retweet count, $\hat{R}_\infty(t)$, for the given post at time infinity, is calculated using 4.3. The α_t and γ_t are scaling constants that are used to minimize the median absolute percentage error. n_* denotes the mean of the degree i.e. followers count, of the actors involved in information cascade until time, t.

$$\hat{R}_\infty(t) = R_t + \alpha_t p_t (N_t - N_t^e)/(1 - \gamma_t p_t n_*) \tag{4.3}$$

4.3 SEISMIC Results

Using the library 'seismic' available on CRAN, we tested the SEISMIC algorithm on our datasets. The description of the dataset and metrics to evaluate a model are explained in the coming chapter. We have considered the tweets that have at least 10 reshares and observed cascade in the first 20 minutes after the original tweet gets published. For a tweet sequence, the input to the model contains multiple lines, where each line represent the node that reshared them in the first twenty minutes. There are two components present in each line of the input, which are the relative time difference(in seconds) between posted time of retweet and original tweet, and the followers count of each node sharing. The model always requires the first line of

the input to hold information of the node that is the author of the tweet sequence and the time difference to be sorted in ascending order. Hence, the first line will always contain 0 and the number of followers for the author. The rest of the lines will contain information about the nodes that reshare them.

Table 4.1.

Performance Evaluation on SEISMIC Model

Dataset	**Mean Absolute Error**	**R2 Value**
Kenya Elections	9.31	0.79
South Africa Elections	8.27	0.74
Nigeria Elections	16.14	0.92

To understand how the model predicts, we tested on the same test dataset that we prepared for our models. Table 4.1 explains the performance of SEISMIC on our datasets. Figure 4.1 shows the comparison between SEISMIC predicted value and original retweet count for top ten popular tweets in Nigerian Presidential Elections dataset.

In this chapter, we have covered the baseline model SEISMIC, which is recognized very well within social media analytics researches. The coming chapter explains the design and implementation of three algorithms that were tested in our research work, and the model evaluation metrics.

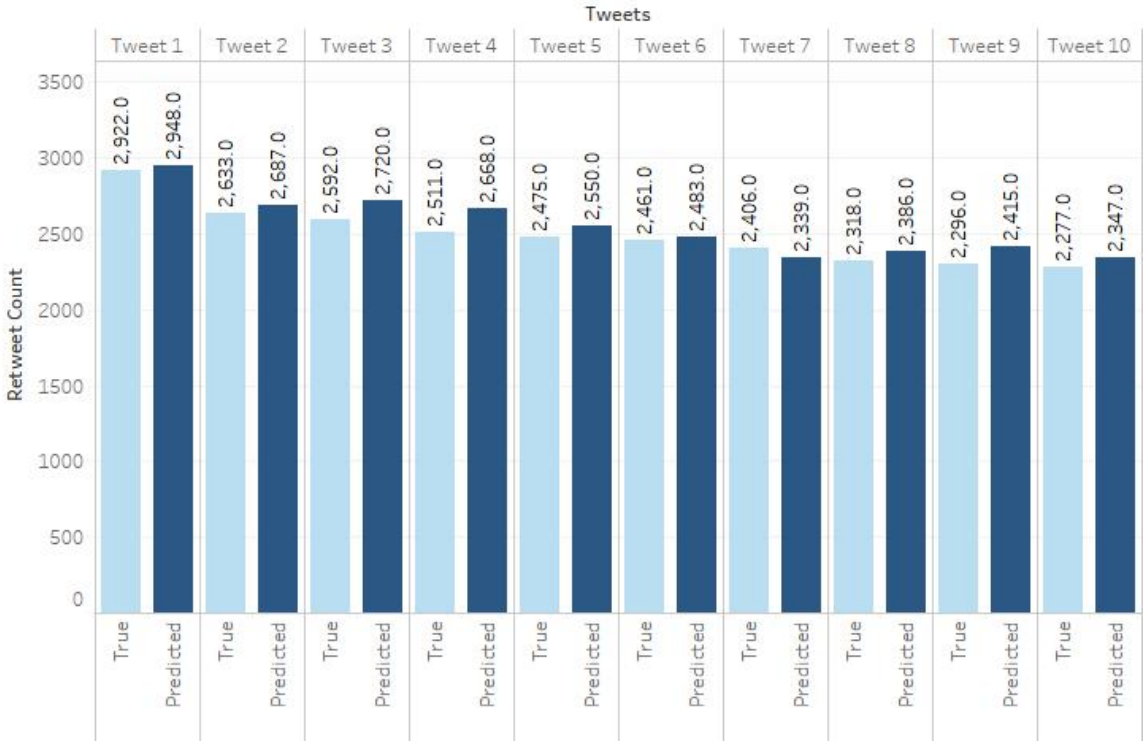

Fig. 4.1.

Top ten trending tweets in Nigerian Presidential Elections - Comparison between original retweet count and SEISMIC predicted values

5. DESIGN AND IMPLEMENTATION

5.1 Introduction

In this section, we describe the data acquisition, feature engineering, and the different machine learning models tried and tested in our research work to predict retweet count. As we are trying to predict the number of retweets possible for a given sequence, which is a scalar value, we can model this problem statement as a regression problem. We should identify necessary information from the raw dataset that is critical to the solution.

The rest of the chapter is organized as follows. First, we will see the background of the data that we are using for our research. Then, the overall architecture of the model is explained in detail, followed by a discussion on how we performed the feature engineering task. The next section explains the different models tried in this work, followed by a discussion on the evaluation metrics considered. In the coming chapter, we will go into detail on the experimental results of these algorithms.

5.2 Overall Architecture of The Model

To solve this problem, we have designed feature-based regression models in which we extract necessary features from the raw data and predict the possible number of retweets for that tweet. In a regression problem, there are two types of variables that are crucial to the model, dependent and independent variables. The dependent variable is referred to as a target variable. We find the correlation between independent features and the target variable, which is the output. We are aiming to predict the output of the model with the least possible loss value while being as close to the original value as possible. This is our research work's primary goal.

The following Figure 5.1 describes the overall architecture of the proposed models. After extracting the features and normalizing the dataset, the training and test dataset is constructed. We have considered all the ways to minimize the loss value and boost the prediction accuracy while constructing all three models. The features and target list in the training set that was built earlier are used to train the model. The trained model is used to predict the retweet count for sequences in the test dataset. We then assess our models performance by some evaluation metrics.

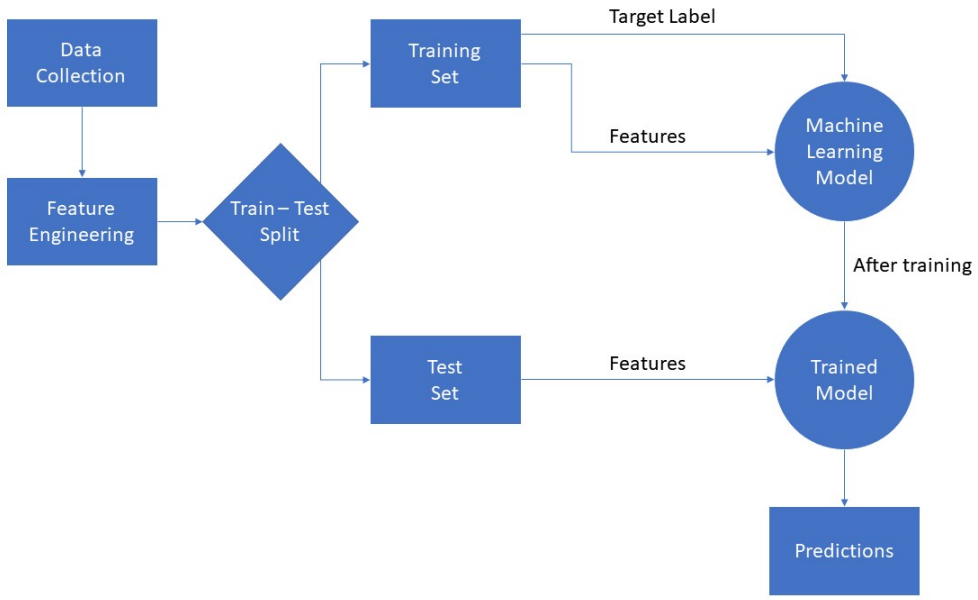

Fig. 5.1.

Overall Architecture of the model

Now, it's time to identify the type of regression model that suits our problem well, and the critical features that are going to be extracted from the original dataset.

5.3 Feature Engineering

For any predictive analytics, the first step is to extract the useful information from the dataset. This is referred to as the features. Also, we can say features

are the representation of the original dataset. They indeed have a large impact on the output and contribute to the success of a machine learning algorithm to a great extent. Hence, better feature engineering enhances the accuracy and effectiveness of the model. In this section, we are going to discuss the different types of features extracted. Also, we will cover the detailed description of every feature used in the computation.

Table 5.1 lists the three types of features that are identified to be informative in building the features set. They are the features of the tweet, the features of the author who is posting or retweeting tweets, and the sentiments embedded in the tweet content. All the identified features except the feature that identifies the party supported in the tweet are continuous variables. Most of the features are independent of each other. For instance, the mentions in the tweet are independent of the hashtags listed. But, the number of tweets in the first ten minutes has some influence on the number of tweets in the second ten minutes.

The brief introduction to features in all the categories, and a discussion on how some of the identified critical features are going to influence target will be covered in the next section.

5.4 Tweet Content Related Features

The body of the tweet is the atomic building block of everything in Twitter. Hence, it's essential to have some features that are extracted from the original tweet content. The goal of the project is to forecast retweet count based on how it's cascading in the initial stages. We have set a time window of twenty minutes and captured some of the information. In other words, observing the trends in the early stages i.e. twenty minutes, our models predict output in the future.

1. Number of mentions

   This is one of the critical features in our features set. In the original post, the author might have mentioned some members relevant to that tweet. Also, the

Table 5.1.

Categories of features extracted from the data

Tweet Content related features	**User related features**	**Sentiment related features**
1. Number of mentions 2. Number of unique words 3. Number of hashtags 4. Number of media items 5. Number of tweets in the first ten minutes 6. Number of tweets in the second ten minutes	1. Number of followers 2. Number of friends 3. Is verified	1. Polarity of the tweet 2. Respective party supported in the tweet

people who are retweeting this post may tag somebody in the post. By tagging them, the post starts reaching out to others. As this method helps in spreading the post, we have considered the number of the unique mentions present in the original and retweets in the first twenty minutes after the original tweet got published.

2. Number of Unique Words

   Twitter recently announced the doubling of the 140-character limit for a tweet to 280 characters. But, the dataset that we have contains maximum of 140 characters per tweet as they were collected before this new announcement. We have kept the number of words that are contained in the tweet as a feature.

3. Number of Media Items

   In some cases, having image(s) or video(s) will influence the reshare of the original tweet. Hence, we have kept the number of media items contained in the tweet.

4. Number of tweets in the first ten minutes

   The greater the number of shares in the first ten minutes indicates the post is going to be viral. In this feature, we are capturing the number of retweets in the first ten minutes.

5. Number of tweets in the second ten minutes

   Like the previous one, this feature indicates the number of retweets in the second ten minutes after the original tweet got published.

5.5 User Related Features

Enhancing the spreading of the news means making the tweet appear in news feeds of larger groups of audiences. Followers and friends play an important role in the information cascade.

1. Number of Followers

   This is a value that tells how many followers the author of the tweet has. So, the more follower an author has, the wider the exposure of tweets for him/her.

2. Number of Friends

   This value tells the number of people that the author follows. Being a friend of another Twitter user helps in information cascade of the tweets authored by his/her friend.

3. Is Verified

   This is a binary variable indicating whether the author of the post is a celebrity or not. Twitter allows any user to get his/her account verified via a number of steps. In general, the posts coming from a verified profile are not fake. Hence, we have included this feature as one of our user related features.

5.6 Sentiment Related Features

As the dataset contains tweets that are published/reshared during elections, it makes more sense to add features related to a political scenario. We have used sentiment analytics in understanding them. We have kept two features that expresses insights about the emotions expressed in the tweet and have an impact on the dependent variable.

1. Polarity of the tweet

   The polarity score helps us in approximating the orientation that the author kept in mind while writing the tweet. The polarity score is going to be one among the following possible values. If the sentiment expressed in the tweet is classified to be positive, the polarity score is assumed to be 1, which is the maximum value. If the tweet expresses unhappy or negativity, its classified as -1. For the rest of the tweet sentences, which doesnt carry any emotions, it gets 0 score.

2. Respective party supported in the tweet

   It's one of the interesting features we have constructed. We have first identified the top keywords belonging to top political parties in the respective countries, using frequent words, mentions, and hashtags. Then, we have done keyword matching on the tweet content to identify which political party the tweet belongs to. If the polarity score of the sentence is negative, i.e. the tweet is unhappy, we assume the tweet belongs to the opponents.

The following Table 5.2 shows the top ten trending hashtags, mentions and frequent words contained in the tweets belonging to South Africa Elections. The characters written inside the brackets denotes their most relevant political party. Unknown means the party cannot be found using the given keyword.

Table 5.2.
List of trending hashtags, user mentions and words in the South African presidential elections

Popular Hashtags	**Popular Mentions**	**Frequent Words**
1. ayisafani(ANC)	1. helenzille(DA)	1. da(DA)
2. siyanqoba(Unknown)	2. lindimazibuko(DA)	2. anc(ANC)
3. ivoteda(DA)	3. julius sello malema(EFF)	3. helenzille(DA)
4. nkandla(ANC)	4. mmusi maimane(DA)	4. zuma(ANC)
5. zuma(ANC)	5. myanc_(ANC)	5. maimaneam(DA)
6. iecmustanswer(Unknown)	6. agangsa(Agang)	6. malema(EFF)
7. togetherforchange(DA)	7. jacob g. zuma(ANC)	7. ayisafani(ANC)
8. wecanwin(ANC)	8. iecsouthafrica(Unknown)	8. amp(Unknown)
9. voteda(DA)	9. mamphela ramphele(Agang)	9. elections2014(Unknown)
10. 20yrsdemoc(Unknown)	10. whyivoteanc(ANC)	10. sabc(Unknown)

The following Figure 5.2 is an example tweet gathered during South Africa Elections. The way the party is supported in the tweet is identified is as follows. Running the polarity test on the body of the tweet, it's polarity score is found to be positive.

There are keywords like anc, and whyivoteanc present in the tweet. As the polarity score is positive, and the keywords assume the party to be ANC, we classify that the tweet belongs to an author who supports ANC.

Fig. 5.2.

A tweet posted during South Africa Presidential Elections

5.7 Linear Regression Model

Linear regression is one of the commonly used types of regression. The term linear in the model describes the proportionality between the dependent and independent features. If there is a change in the independent variables, it will affect dependent value as well. We have used 70-30 train-test split approach. This means, the top 70 percent data from the shuffled original dataset constitutes the training set whereas the rest forms the test set.

The model fits the independent features, described in the previous section, into a line against the target variable, i.e. retweet count. Once the model is fit with the use of training data, we feed into the model the unseen independent features and wait for its predicted result. The found output is evaluated against the original value to determine how well the model is performing.

Since we have multiple features, we can call this approach multiple linear regression problem, which is a variant of linear regression. The model can be expressed mathematically in the form.

$$y = \beta_0 + \beta_1 X_1 + \beta_1 X_2 + ... + \beta_1 X_n \tag{5.1}$$

The output variable y in the equation 5.1 is the dependent variable, i.e. retweet count. The variables starting with X, i.e. X_1, X_2, until X_n, denotes the n independent features discussed in the previous section.

The whole process of forming the training and test dataset is as follows. As the dataset contains n number of tweets sequences, we have taken into account the tweet sequences that have at least a 10 reshare count. Our features set contains both static and dynamically changing features as well. For those identified tweet sequences, we have found the static features, like the followers count, friends count, polarity score. Simultaneously, we have also calculated dynamic features like mentions count, hashtags count, and number of tweets in first and second ten minutes with what is observed in the twenty minutes time window after the original post. The objective is, using the early computed features to try to predict whats going to happen in the future.

Once the feature matrix for all tweet sequences are identified and stored in a single dataframe, we shuffle the rows and assign the top 70 percent of rows to the training set and the rest forms the test dataset. Later, we train the model using the data from training set and predict the results on the test dataset. In the last section of this chapter, we will discuss how are we evaluating the models performance.

5.8 Feed Forward Neural Network Model

In this section, we will discuss the concept of feed forward neural networks, the layers present in it, the way the network is trained, and the optimization functions. A feed forward neural network model often called multilayer perceptrons, contains

several layers of neurons connected to another set of neurons in the subsequent layer or to the neuron(s) in the last output layer. A multilayer perceptron model contains at least three layers of neurons, i.e. input layer, hidden layer(s), output layer. In our regression problem, we have used non-linear activation function on outputs from input layers, and hidden layers (except the last hidden layer). The term feed-forward in the name of the model explains the communication from input layer to the output layer takes place in a direction, and there are no ways to pass feedback about the output of the model into itself. Usually, the nodes in a layer will be fully connected with all the nodes in the next layer. Figure 5.3 depicts an example of a feed-forward neural network model.

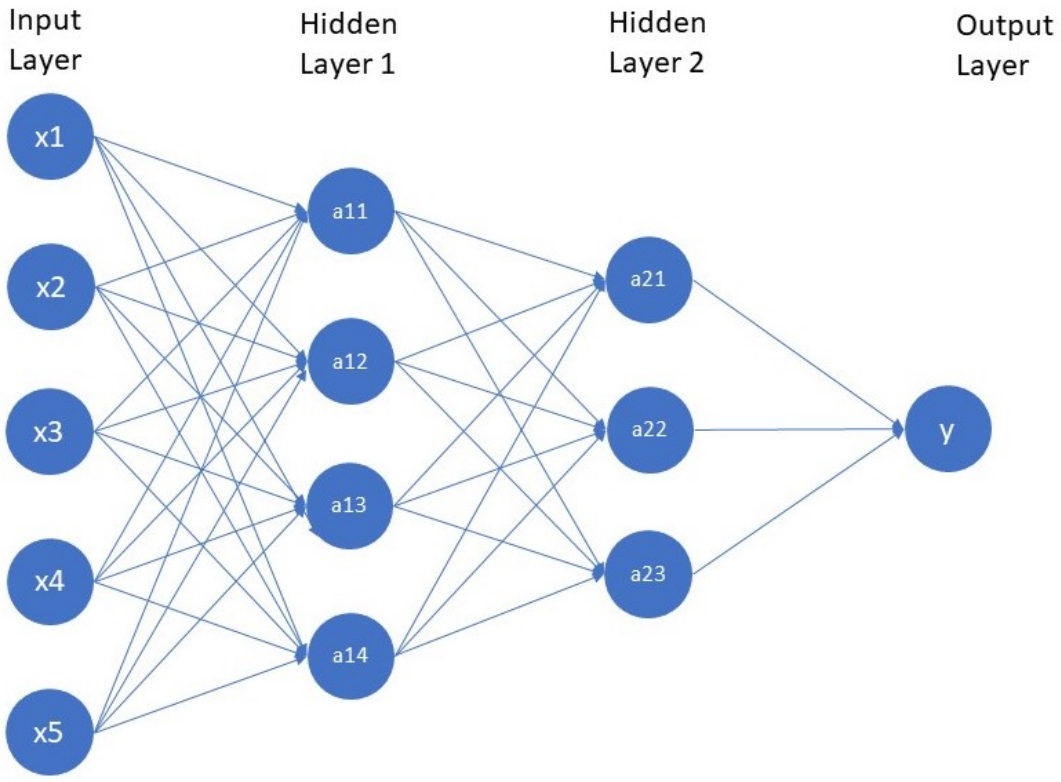

Fig. 5.3.

A Feed Forward Neural Network Model

The left most layer with node values $(x1, x2, x3, x4, x5)$ is called input layer. The right most layer that contains only one node(y) is called output layer. The layers in the middle are referred to as hidden layers. The values that are contained in the hidden layers, indicate an activation function has been applied on top of the output received from the previous computation. It's assumed that there are weights present on all the edges connecting the nodes in the graph.

The first step in computing value for a hidden layer node is by summing the product of weights incoming to the node with the node value at the previous layer. Then, bias value is added to the summed value. The next and final step is to apply the activation function on the computed value. Equations 5.2 and 5.3 formulate the process involved in computing the value $a11$ and is the same for all the hidden layer nodes. The $w_{11}, w_{12}, w_{13}, w_{14}$ represent the weight of the edges incoming towards node $a11$.

$$net_{11} = (w_{11} \cdot x_1) + (w_{12} \cdot x_2) + (w_{13} \cdot x_3) + (w_{14} \cdot x_4) + b \tag{5.2}$$

$$a_{11} = \text{ReLU}(net_{11}) \tag{5.3}$$

We can generalize the computation on nodes in the neural network as given in equation 5.4. The σ denotes the non-linear activation function that's applied on the computed value.

$$f(H^{(l)}) = \sigma(H^{(l)}W) \tag{5.4}$$

Once we get the output values with random weights and biases initialized, it will be compared against the original value to compute the loss score. Then, the model will use optimizer functions to minimize the loss, which is referred to as backpropagation. As we have initialized the random weights and biases, these set of values wont accurately represent the neural network. Our ultimate goal is to train the model to predict the original value of that particular instance, which means reducing the error margin as low as possible. To achieve this, we use chain rule of differentiation. It assists in updating the weights that are used currently in the network. With the below listed formula, we can understand how it works. Thus, the weight corresponding to

the connection between the output layer and the neuron at the top of the last hidden layer is adjusted.

Assume the weight contained on the edge starting from the last hidden layer's top neuron, a_{21}, and the output variable, y is w_{31}. Equations 5.5 and 5.6 will help understanding how the value of w_{31} gets changed after computing the y value for that iteration.

$$E_{total} = \frac{1}{2}(target - y)^2 \tag{5.5}$$

$$\frac{\partial E_{total}}{\partial w_{31}} = \frac{\partial E_{total}}{\partial a_{21}} \cdot \frac{\partial a_{21}}{\partial net_{21}} \cdot \frac{\partial net_{31}}{\partial w_{31}} \tag{5.6}$$

In our model, we have 12 features identified for a dataset that are going to be propagated between neurons. We have created a feed forward neural network model having two hidden layers, with 9 and 6 neurons in it. On each iteration, the model learns from the error difference between the original and predicted values and optimizes the cost function using Adam Optimizer. The role of activation functions, loss functions, and optimizer functions is discussed in the later section. We construct the training and test dataset using the same approach practiced in the previous model.

5.9 Graph Convolution Neural Network

For our research, we tried the concept implemented by the researcher, Thomas Kipf, of University of Amsterdam. The idea of graph convolution neural network is relatively new to social media analytics. We have extended his idea [35] of applying GCN to classify research papers based on the association among the works cited in it, to solve our problem taking into account the similarity between user behaviors in retweeting. To our best knowledge, we are the first team to extend this idea to this sort of problem. In this section, we will cover the concept of GCN, the difference of approach between GCN and the feed forward neural network, and the environmental setup for executing the program.

5.9.1 Difference between GCN and Feed Forward Neural Network

Usually, social networks are represented in the form of graphs. Mining information from graphs is ubiquitous. Observing features at every individual node will not give much information about the connectivity of that node associated with the rest of the network. This means, representing features for every node present in the graph wont reconstruct the graph. The GCN captures information about the connectivity of the nodes as well, in the form of an adjacency matrix, which is a representative description of the graph. With respect to our problem, we can view individual tweet sequences as nodes. The idea is, knowing the similarity between the actors will help in cascading the news among the group.

The major difference between the two neural network models is that the feed forward network operates on feature level, whereas the GCN operates on node level. In a feed forward neural network, every node in a layer represents a feature of an instance in the dataset. Their values gets updated on calculations related to weights, biases, and feature values of the previous layer. We have discussed the computations in the previous section. The important thing is that the model isn't capturing the connectivity between tweet sequences, as they are fed into the model for training separately. By representing the actors contributing to the tweet sequence as individual nodes, we can form connectivity among the nodes. That's the way the GCN model differentiates itself from the former one. Combining everything, the author describes the graph convolution neural network in the equation 5.7

$$f(H^{(l)}, A) = \sigma(AH^{(l)}W^{(l)}) \tag{5.7}$$

5.9.2 An Example to demonstrate Adjacency Matrix construction

In general, when two users reshare the same post, they are more likely to have similar interests. This is the approach followed in framing the connectivity graph between authors of tweet sequences. Using the toy dataset given in Table 5.3, we can understand how we are computing the adjacency matrix.

Table 5.3.

Toy dataset

Retweet Id	**Retweet Actor Id**	**Original Tweet Id**	**Original Tweet Author Id**
R1	A	T1	D
R2	B	T1	D
R3	C	T1	D
R4	A	T2	X
R5	D	T2	X
R6	B	T3	Y
R7	C	T3	Y
R8	D	T3	Y

In this dataset, there are seven retweet sequences(R1 to R8) present. There are three original tweet sequences(T1 to T3) that get shared among this dataset. There are six actors involved in the dataset. They are A, B, C, D, X, Y. Table 5.4 shows the adjacency matrix built on the actors involved in the dataset, which indicates the connectivity among them.

Table 5.4.

Adjacency Matrix for the toy dataset

	A	**B**	**C**	**D**	**X**	**Y**
A	0	1	1	1	0	0
B	1	0	2	1	0	0
C	1	2	0	1	0	0
D	1	1	1	0	0	0
X	0	0	0	0	0	0
Y	0	0	0	0	0	0

Thus the adjacency matrix is constructed. The value 2 between B and C indicates B and C have shared two tweets among their network. We have considered other options to build the adjacency matrix with the help of mentions. The coming chapter will help in understanding what's missing in these alternatives.

5.9.3 Architecture of our GCN Model

For this work, we have used a deep learning model of three hidden layers (two GCN layers and an fully connected layer). The section 5.9.2 explains the steps in building adjacency matrix A, which remains constant for all tweet instances. The features matrix, X for the GCN layer is going to be the contributors matrix. The contributors matrix contains three features named contributors, active contributors, and mention contributors. All three of them are binary matrices. The ones in the active contributors matrix denotes that the actor had retweeted the given tweet. The ones in the mention contributors matrix denotes that the actor has been referred to in the mentions. We initialize the weights, W randomly and keep on updating it on each iteration. After passing through two GCN layers, the output is concatenated with tweet related features before feeding it into the fully connected layer. To speed up the computations, we have considered only the tweet related features from the list given in 5.1 except the UTC time difference. The actor related features and the tweet related features gets changed according to the tweet sequence.

Given a graph with actors as nodes, the connectivity among the nodes, the adjacency matrix for a tweet sequence, the values of GCN layer neurons are computed using the equation 5.7. Once the GCN layer computations are done, we concatenate the output with the tweet related features corresponding to the particular tweet and feed into fully connected layer. The values of neurons at the fully connected layer is computed using the equation 5.4. The overall architecture of the GCN model is visualized in the figure 5.4. The rationale behind keeping actor information in the graph convolution layer is that X corresponding to every tweet sequence is an indicator of

the part supported in the tweet. When multiplying A with X, it approximates the users belonging to the same party, who may be exposed to that tweet sequence. It's a way of increasing the outreach of the tweet sequence beyond the immediate followers and friends.

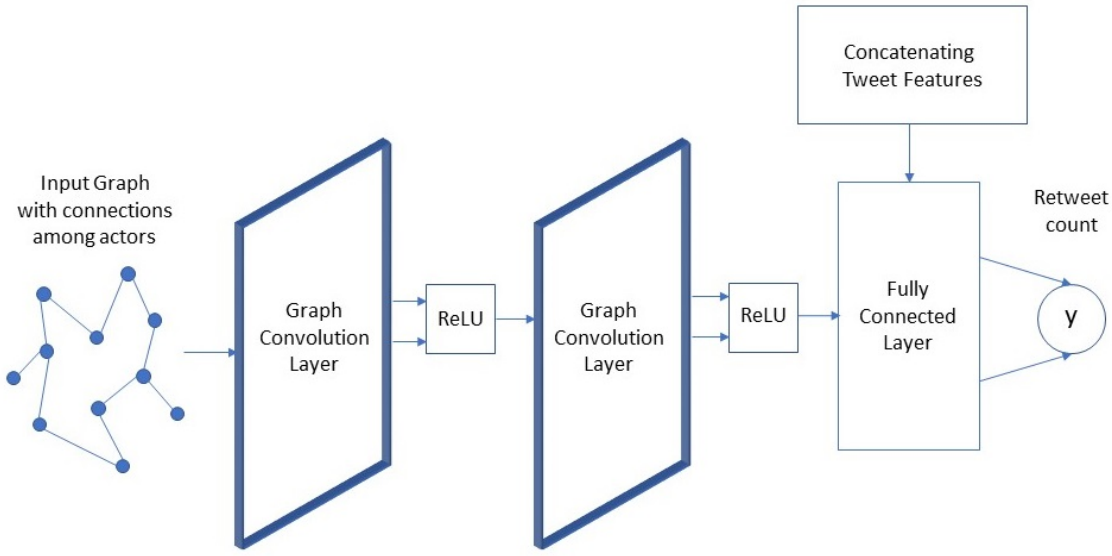

Fig. 5.4.
Architecture of the Graph Convolution Neural network Model

5.10 Example Calculations

The following example will summarize everything that we discussed regarding the graph convolution neural networks with calculations. Assume the graph given in the figure 5.5 is a network of people subscribed to an advertising company and there are two advertisements. The goal is to predict a target advertisement for each node out of the two, taking into account both the properties of the nodes and the network. So, we can call it as a two class classification problem. Assume every node contains three features, x_1, x_2 and x_3, which forms the feature matrix X. The adjacency matrix, A represents the connectivity among nodes. The presence of an edge linking two nodes

is marked as 1 in A between them. A sample feature matrix and the adjacency matrix corresponding to the graph 5.5 is listed in tables 5.5 and 5.6. We implemented a deep learning model of two hidden graph convolution layers, similar to explained in figure 5.4 except the fully connected layer.

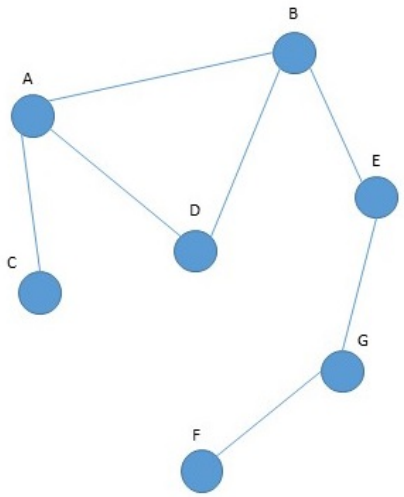

Fig. 5.5.
Input graph of actors subscribed to an advertisement channel

In the layer 1, let's convolve the graph of actors having three features into four. First, We randomly initialize the weight and bias of dimensions 3*4 and 1*4. Later, we multiply X with W and add bias, followed by multiplying with A. After adding the bias to the output, we apply activation function, ReLU. The output of layer 1 is given in the table 5.7. In the second Graph Convolution layer, we convolve the output of the previous layer into two dimensional features which represents two classes, using softmax activation function. Their values denote the probability of the person belonging to that particular class. For each instance, the final class label is the class that holds the maximum score among them. Prediction results are given in the figure 5.8.

Table 5.5.

Features Matrix - X

	x_1	x_2	x_3
A	2	4	5
B	4	1	5
C	3	4	10
D	6	3	4
E	7	8	8
F	5	2	1
G	6	7	8

Table 5.6.

Adjacency Matrix - A

	A	**B**	C	D	E	F	G
A	0	1	1	1	0	0	0
B	1	0	0	1	1	0	0
C	1	0	0	0	0	0	0
D	1	1	0	0	0	0	0
E	0	1	0	0	0	0	1
F	0	0	0	0	0	0	1
G	0	0	0	0	1	1	0

Table 5.7.

Output of Graph Convolution layer 1

	f_1	f_2	f_3	f_4
A	0.405	0.084	0.359	0.054
B	0.412	0	0.355	0
C	0.098	0.001	0.097	0.027
D	0.208	0.035	0.18	0.037
E	0.293	0.032	0.256	0.008
F	0.184	0	0.164	0
G	0.260	0	0.205	0

Table 5.8.

Output of Graph Convolution layer 2

	class 1	**class 2**	**Class label**
A	0.48	0.052	**1**
B	0.4993	0.5006	**1**
C	0.4996	0.5003	**1**
D	0.4995	0.5004	**1**
E	0.501	0.4008	**0**
F	0.500	0.499	**0**
G	0.5002	0.4997	**0**

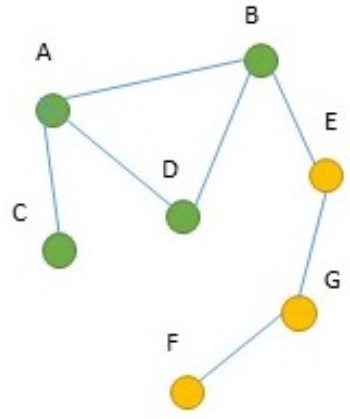

Fig. 5.6.

Final Predictions on toy dataset using Graph Convolution Neural Networks

To summarize, the graph 5.6 is obtained as the result of applying the equation 5.8 on the input graph 5.5.

$$f(X, A) = softmax(A\ ReLU(A\ X\ W^{[1]})\ W^{[2]}) \tag{5.8}$$

After classification using Graph Convolution Neural Networks, the output appears as given in the figure 5.6. The nodes marked with green represents some portion of actors who are going to receive same advertisement. Similarly, the people marked in yellow will receive another set of advertisement.

5.11 Loss Functions, Activation Functions, and Cost Functions

We have earlier discussed how the backpropogation helps in the adjustment of weights of previous neurons. The difference in the predicted and target value is referred to as loss. In both of our models, we have used mean squared error as the loss function.

Mean-squared error is defined as the average of the square of the difference between original and predicted values. The formula to compute Mean Squared Error is given in equation 5.9.

$$Mean\ Squared\ Error = \frac{1}{n}\sum_{i=1}^{n}(original_i - predicted_i)^2 \tag{5.9}$$

As discussed earlier, our goal is to minimize the error in the prediction and maximize the accuracy of the model. The optimization function is achieving this goal. In our models, we have used the Adam optimizer. Unlike the traditional stochastic gradient descend algorithm, the Adam optimizer maintains adaptive learning rate, where the current update relies on previous update to some extent. It helps to keep the momentum going forward throughout the training.

An activation function is applied to any value computed in the hidden layers. The role of the activation function is to convert an input signal into an output signal with added non-linearity. If we dont add activation functions, the model behaves nearly the same as the linear regression. There are different types of activation functions available. However, we have chosen Rectified Linear Unit activation function, commonly called ReLU to solve our problem. Sometimes choosing the simple method yields the best results. From the equation of the ReLU given in 5.10, we can easily infer that passing through this function will give output x if x is positive. If not, it will take zero.

$$ReLU(x) = maximum(0, x) \tag{5.10}$$

5.12 Performance Evaluation

5.12.1 Mean Absolute Error

The mean absolute error given in 5.11 is defined as the average of the absolute difference between predicted and original values. The less the mean absolute error value the better the performance of the model.

$$Mean\ Absolute\ Error = \frac{1}{n}\sum_{i=1}^{n} |(original_i - predicted_i)| \tag{5.11}$$

5.12.2 R2 Score

The R2 score also referred to as coefficient of determination is a statistical measure of how close the predicted values are mapped to the regression line and is explained in 5.12. The values could be negative if the prediction results are arbitrarily poor. The greater the r2 score indicates the predicted values are close to the regression line.

$$r^2\ score = 1 - \frac{\sum_{i=1}^{n}(original_i - predicted_i)^2}{\sum_{i=1}^{n}(original_i - mean\ of\ original\ scores)^2} \tag{5.12}$$

6. ANALYSIS OF RESULTS

6.1 Introduction

We have discussed the three types of models that have been developed in this work, so far. In this section, we will see how well each model performs, the portions where it could be improved, and a discussion on how each model performs against the baseline model, SEISMIC discussed earlier. We have used visualizations for the ease of understanding.

6.2 Dataset Pruning

We can find from 3.2 that there are 140,521 individual tweet sequences present in the Kenya Elections dataset. Among them, there are 130,942 tweets containing reshares less than 10, which is around 93 percent. It will be expensive and unnecessary if we apply machine learning to predict retweet counts for tweets that won't go viral. Hence, we have taken into account only the tweet sequences that have reshare count above 10. Figure 6.1 provides an overview of the number of tweets that have retweet count lesser than 10 in South African elections. As an input to all the models, we will consider only the tweet sequences after pruning. Table 6.1 shows the number of tweet sequences before and after pruning for all the three datasets.

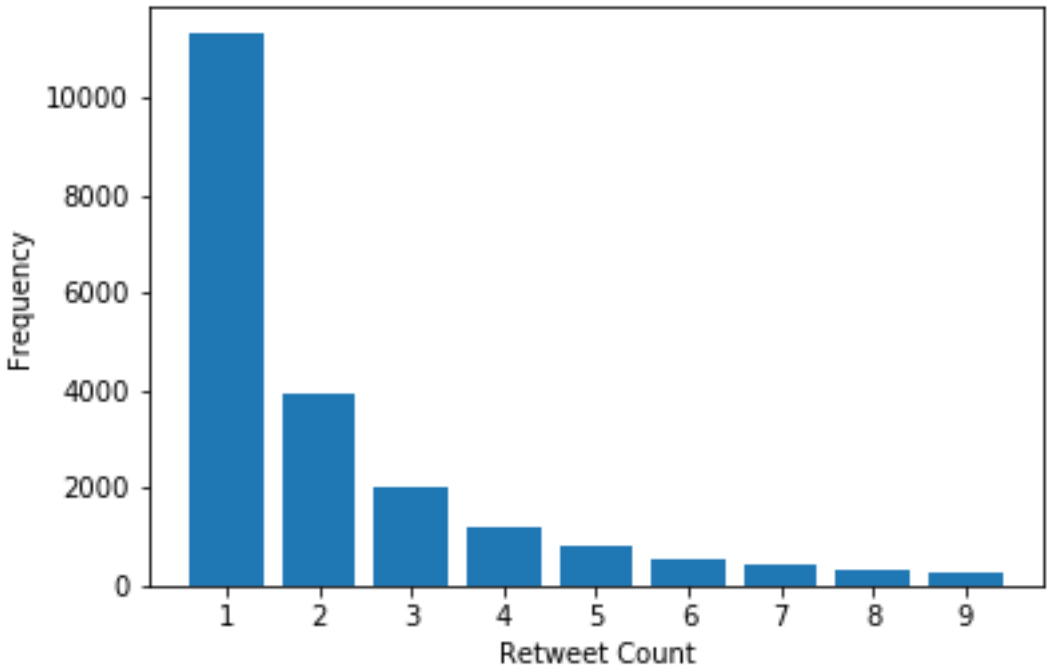

Fig. 6.1.

Frequency of tweets in South Africa Elections dataset having retweets less than 10

Table 6.1.

Results of Data Pruning

Dataset	**Number of tweet sequences before pruning**	**Number of tweet sequences after pruning**
Kenya Elections	140521	9579
South Africa Elections	22572	1677
Nigeria Elections	211691	31672

6.3 Performance of Linear Regression Model

We know that there are three types of features that we have extracted from the original dataset and filtered out tweet sequences that have reshare count less than 10. There are many features that we obtain directly from the dataset. Examples include: followers count, friends count, and number of mentions. However, the sentiment related features need use of external packages and some political knowledge on the parties, and candidates in the respective countries. To say a tweet is belonging to a political party, we have done some keyword matching.

First, we identified the top 60 most frequent user mentions, hashtags and the words in the tweet. The results for popular hashtags and mentions looked reasonable and convincing. But, the most popular words contained many stop words, which don't carry any meaning. We used the nltk package to remove stopwords from the tweet content and identified the trending words. The figure shows the top ten trending hashtags, mentions, and words in the dataset. We made use of the Wikipedia, twitter trends, interesting stats, and identified the major political parties from all the three countries. For instance, there are four major political parties in South Africa, African National Congress, the Democratic Alliance, the Economic Freedom Fighters, Agang. It appears the last three parties have different reasons to be an opponent of the African National Congress. The African National Congress is the ruling party, whereas, the Democratic Alliance is the primary opponent. The media company research says, the DA was more active on social media during the elections than anyone. On average, the party or party spokesperson from the opponent party tweeted 28 tweets a day, whereas the ruling party tweeted only five times a day. Similarly, Nigeria has two predominant political parties, which are Peoples Democratic Party and the All Progressives Congress. We have conducted similar study to identify the keywords belonging to each political party. When we find the polarity of the sentence to be negative, we assume the tweet is belonging to opponent party.

Figure 6.2 is a tweet posted by a twitter user Farieda Khan, regarding the scandal that took place at Nkandla, the birth place of the president of South Africa, his excellency Jacob Zuma. There are keywords like Nkandla, and ANC that makes our keyword matching algorithm (without considering polarity factor) to say, the tweets belongs to the African National Congress. But the opinion expressed in the tweet shows anger, and sadness which means the polarity of the sentence is negative. So, we classify the tweet into the opponent of the ANC, which is DA.

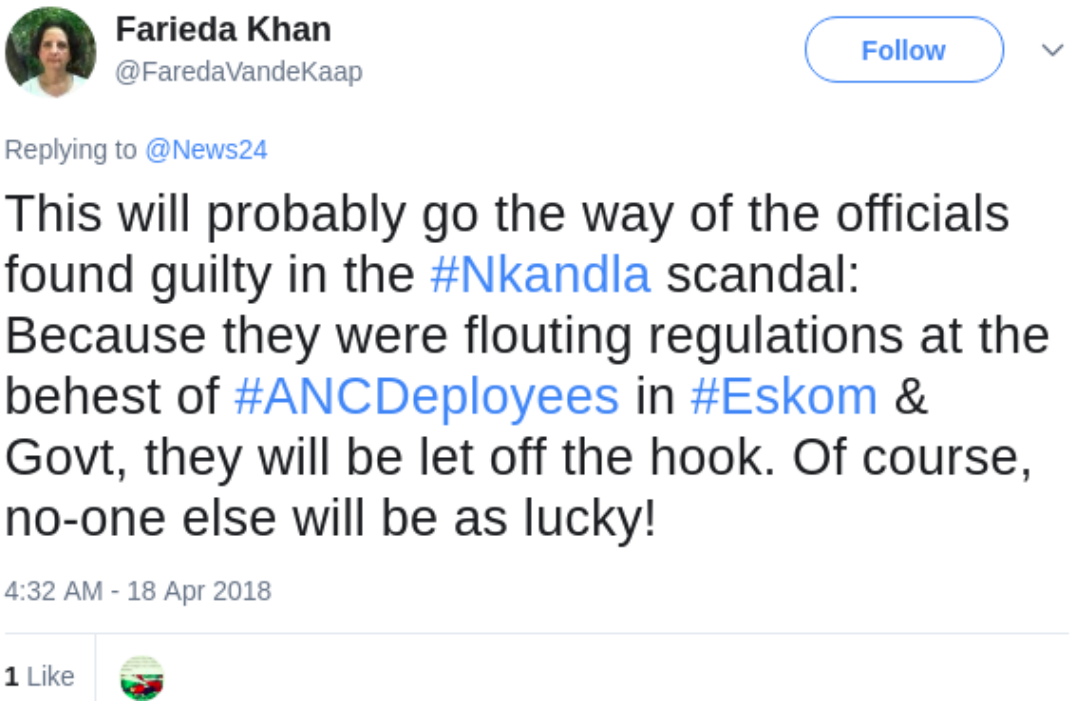

Fig. 6.2.

A tweet containing information regarding to Nkandla scandal

For tweet given in Figure 1.1, the feature matrix is constructed as represented in Figure 6.3. Once the feature matrix for all the necessary tweet sequences is found, we shuffle and construct training and test set using 70-30 approach as discussed earlier. The following Table 6.2 shows the number of tweet sequences in every dataset, and the dimensions of training and test dataset.

```
{
"followers_count": 339650,
"friends_count": 336,
"is_verified": "TRUE",
"media_count": 0,
"posted_time": 1399400000,
"mentions_count": 180,
"word_count": 19,
"hashtag_count": 0,
"first_10": 23,
"second_10": 22,
"emotion": 0,
"party": 2
}
```


Fig. 6.3.

Feature Representation for the tweet contained in Figure 1.1

Table 6.2.

Dimensions of training and test dataset

Dataset	**Dimension of entire dataset after pruning**	**Dimension of Training set**	**Dimension of Test set**
Kenya Elections	9416 * 13	6592 * 13	2824 * 13
South Africa Elections	1568 * 13	1098 * 13	470 * 13
Nigeria Elections	29947 * 13	20963 * 13	8984 * 13

Using the scikit learn packages available in the python language, we implemented linear regression and experimentally measured the performance of the model. The results after testing the model on our features matrix is listed in Table 6.3 .

Table 6.4 shows one of the interesting studies that we conducted to observe the set of features that influences the performance of the model. Looking at the contributions of tweet related features alone, it's very clear that it plays crucial role in predicting the

Table 6.3.

Performance Evaluation on Linear Regression Model

Dataset	**Mean Absolute Error**	**R2 Value**
Kenya Elections	7.97	0.86
South Africa Elections	11.85	0.59
Nigeria Elections	18.82	0.92

retweet count. Also, the addition of actor related information increases the value of the model. The improvement can be seen with the raise in the R2 value. As expected, the actor related features and the sentiment related features alone can't predict the retweet count. Also, the addition and removal of sentiment related features from permutations didn't alter the performance significantly.

Figure 6.4 shows the correlation between the true and predicted values after running our model on South Africa Elections dataset. As we see that more number of points are accumulated between 0 and 200, we can observe that there are many tweet sequences that don't have retweet count in bigger numbers.

Among the datasets, the regression line looks the best with the Nigeria dataset. As the dataset is huge, the model got enough data to train and learn parameters from them. Hence is the accuracy in prediction. Figure 6.5 shows the scattered plot drawn between these values on the Nigeria dataset.

The reason the linear regression model shows superior results than the SEISMIC model is that the features identified are mostly independent of each other, which satisfies the basic assumption of Linear Regression.

Figure 6.6 shows the comparison between Linear Regression predicted value and original retweet count for top ten popular tweets on Kenyan Presidential Elections dataset.

Table 6.4.

Different types of features and their influence on performance of the Linear Regression Model

Features	**Dataset**	**Mean Absolute Error**	**R2 Value**
Tweet Related Features	Kenya Elections	7.87	0.86
	South Africa Elections	12.02	0.57
	Nigeria Elections	18.7	0.92
Actor Related Features	Kenya Elections	17.06	0.005
	South Africa Elections	14.42	0.10
	Nigeria Elections	57.87	0.006
Sentiment Related Features	Kenya Elections	17.39	-0.007
	South Africa Elections	15.96	-0.02
	Nigeria Elections	57.76	-0.012
Tweet Related Features + Actor Related Features	Kenya Elections	7.9	0.86
	South Africa Elections	11.88	0.6
	Nigeria Elections	18.77	0.92
Tweet Related Features + Sentiment Related Features	Kenya Elections	7.94	0.86
	South Africa Elections	12.09	0.56
	Nigeria Elections	18.73	0.92
Actor Related Features + Sentiment Related Features	Kenya Elections	17.12	0.007
	South Africa Elections	14.5	0.09
	Nigeria Elections	57.91	0.007
All three (Tweet, Actor,Sentiment) related Features	Kenya Elections	7.97	0.86
	South Africa Elections	11.85	0.59
	Nigeria Elections	18.82	0.92

6.4 Performance of Feed Forward Neural Network Model

Construction of multi layer perceptron or feed forward neural network is easier using the tensorflow package available in python. Similar to the linear regression

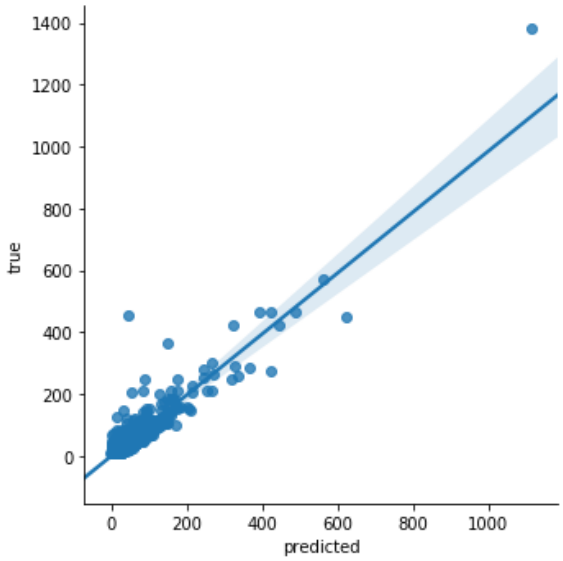

Fig. 6.4.

Scatter plot drawn between original and Linear Regression predicted values on South Africa Elections Dataset

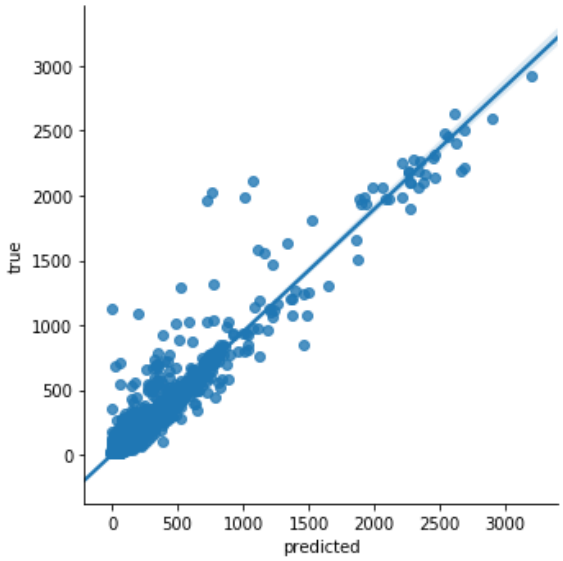

Fig. 6.5.

Scatter plot drawn between original and Linear Regression predicted values on Nigeria Elections Dataset

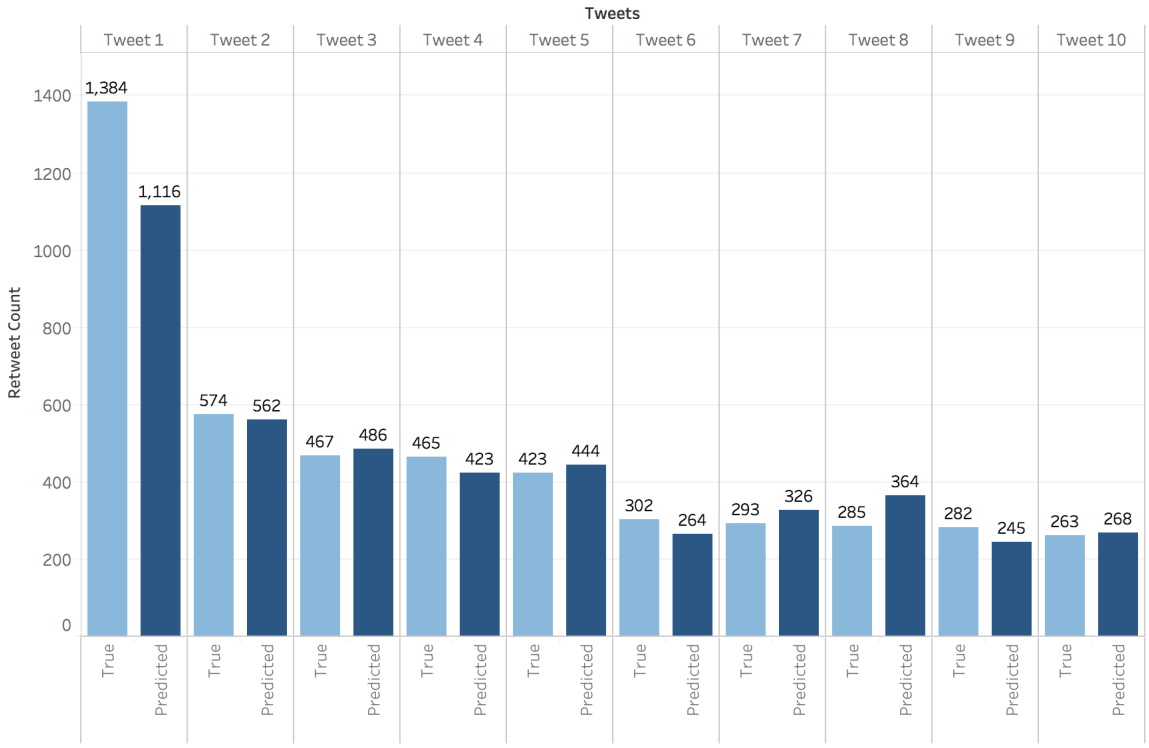

Fig. 6.6.

Top ten trending tweets in Kenyan Presidential Elections - Comparison between original retweet count and Linear Regression predicted values

model, the neural network yielded reasonable results. The batch size for the model is set at 50. The number of epochs denotes the number of times we train and predict the model using the constructed network. In our model, we have set the epochs count as 500. The optimizer function that we used in our problem is Adam optimizer with learning rate of 0.001. it takes care of reducing the loss value between the original and predicted values for the trained instances. This helps in obtaining the optimal weights for the edges in the network.The mean absolute error, r2 score after testing the model with test data is listed in Table 6.5.

Figure 6.7 shows the comparison between feed forward neural network predicted value and original retweet count for top ten popular tweets in South African Presidential Elections dataset.

Table 6.5.

Performance Evaluation on Feed Forward Neural Network Model

Dataset	Mean Absolute Error	R2 Value
Kenya Elections	7.75	0.86
South Africa Elections	14.26	0.32
Nigeria Elections	17.92	0.92

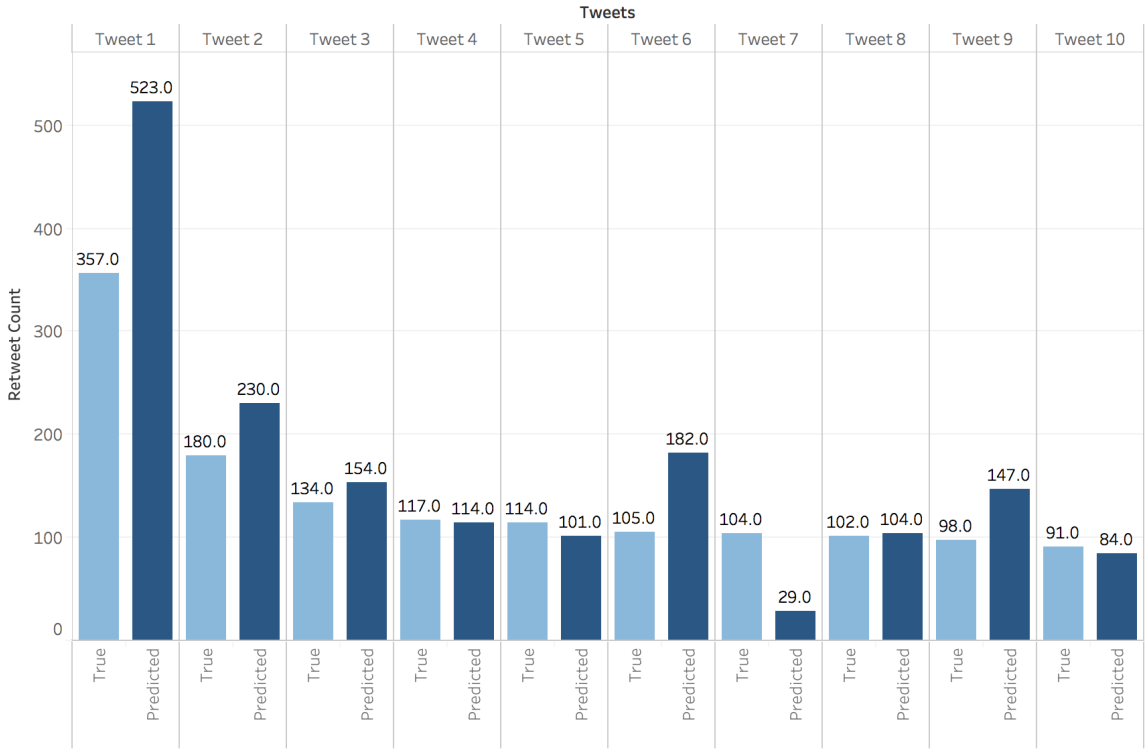

Fig. 6.7.

Top ten trending tweets in South African Presidential Elections - Comparison between original retweet count and Feed Forward Neural Network predicted values

6.5 Performance of Graph Convolution Neural Network Model

In the previous two models, we have extracted features for individual tweet sequences and both the models train themselves on iterations internally. This means

the first tweet sequence in the dataset calculates hidden layer values using the randomly initialized weights and biases, and the backpropogation helps in adjusting the weight for that instance. Using the modified weight and bias, the next instance in the training set predicts values, and in turn, it updates them depending on the loss incurred. So, the point is the tweet instances never consider any connectivity between the tweets, author of the tweets, or people who are resharing them.

The adjacency matrix explains the connectivity among the actors who are involved in the tweet, be it the author of that tweet, the person who reshared, or the person who is mentioned in the original tweet and retweets. Earlier, we discussed an example of how we are calculating the adjacency matrix. However, this is different from that approach that we tried earlier. Initially, we considered only the mentions present in the tweet and formed a binary adjacency matrix. If the cell value of i^{th} row and j^{th} column in the adjacency matrix is one, it means the actor indexed in the i^{th} row has mentioned the actor indexed in the j^{th} row or vice versa, in any of the tweet sequences. The ones in the adjacency matrix denotes that they have mentioned each other at least for a tweet. As we have huge set of actors, framing the binary matrix contained lot of zeros, which affected the calculation of hidden layer values. Then we proposed the alternate solution discussed in the previous chapter. If the cell value of the i^{th} row and j^{th} column in the adjacency matrix constructed using the new approach is n, it means the actor indexed in the i^{th} row and the actor indexed in the j^{th} row have retweeted the same n tweets. It remains the same for the entire calculations.

The results after testing on our datasets is shown in Table 6.6. It shows considerable improvement over all the other models developed for this research work and the baseline model SEISMIC.

Figure 6.8 shows the comparison between Graph Convolution neural network predicted value and original retweet count for top ten popular tweets in South African Presidential Elections dataset.

Table 6.6.

Performance Evaluation on Graph Convolution Neural Network Model

Dataset	Mean Absolute Error	R2 Value
Kenya Elections	7.34	0.72
South Africa Elections	10.18	0.73
Nigeria Elections	16.5	0.89

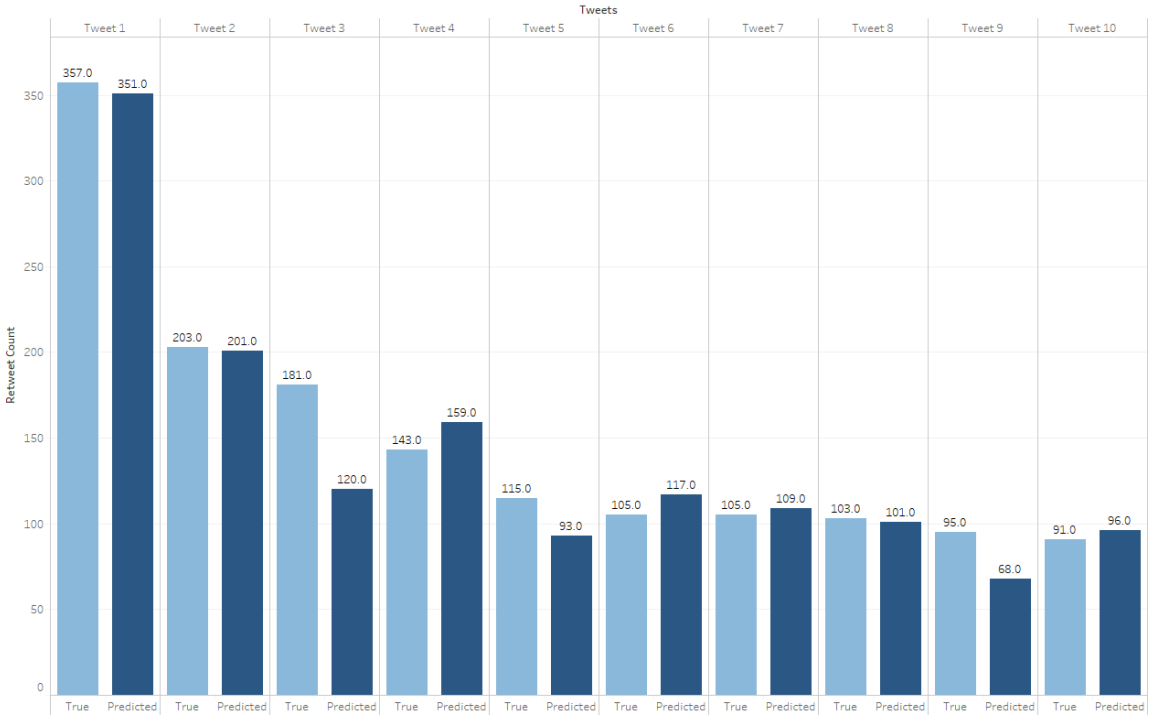

Fig. 6.8.

Top ten trending tweets in South African Presidential Elections - Comparison between original retweet count and Graph Convolution Neural Network predicted values

6.6 Importance of different layers in the GCN model

For GCN model, We take into account the features related to tweet and the adjacency matrix built upon the similarity of user's retweeting behaviors. We have con-

ducted experiments to observe what are the features that influence the performance of the GCN model. Table 6.7 provides information about how the performance of the GCN model gets changed with having only tweet features, actor features, and combination of both, which is the regular model. By keeping both the features, the model performs better than keeping only one. Similar to what we observed from the results of 6.4, having tweet related features reduces error and improves correlation.

Table 6.7.

Different types of features and their influence on performance of the GCN Model

Dataset	**Only Actor features**	**Only Tweet features**	**Both Tweet and Actor features**
Kenya Elections	12.47	11.43	7.34
South Africa Elections	13.20	14.26	10.18

6.7 Performance Summary of All the Models

Until now, we have seen the performance of every model individually and analyzed their effectiveness based on the reported Mean Absolute Error and R2 score values. Comparing the results of each models will make the research complete and help us in understanding how well is our GCN model performing against others. Table 6.8 compares the mean absolute error observed from different models when tested on our datasets. We are able to reproduce similar results computed with the benchmarked model SEISMIC and even better scores for Kenyan elections. It's interesting to see that our GCN model predicts results with 23.7% reduction in error than the benchmarked SEISMIC model on Kenyan elections. Table 6.9 shows, the conventional models fit results to the regression line with around 50 percent for South African Elections. Our GCN model fits values with around 73 percent, which is around 10% better than the next model. Although models - Linear Regression, and Feed Forward Neural Network correlates results very well for Kenyan and Nigerian Elections, the

performance of GCN doesn't look too bad. Figure 6.9 shows the top ten trending tweets in Kenyan Presidential Elections and compares the values predicted by these four models.

Table 6.8.

Comparison of Mean Absolute Error reported by All the Models

Model	**Kenya Elections**	**South Africa Elections**	**Nigeria Elections**
SEISMIC	9.31	**8.27**	**16.14**
Linear Regression	7.97	11.85	18.82
Feed Forward Neural Network	7.75	14.26	17.92
Graph Convolution Neural Network	**7.34**	10.18	16.5

Table 6.9.

Comparison of R2 score reported by All the Models

Model	**Kenya Elections**	**South Africa Elections**	**Nigeria Elections**
SEISMIC	0.66	0.66	0.83
Linear Regression	**0.86**	0.59	**0.92**
Feed Forward Neural Network	**0.86**	0.32	**0.92**
Graph Convolution Neural Network	0.72	**0.73**	0.89

6.8 Other Experiments

In addition to the above mentioned experiments, we have performed some similar and critical tests, that are supplements to check performance of the models and alternatives to actor related features.

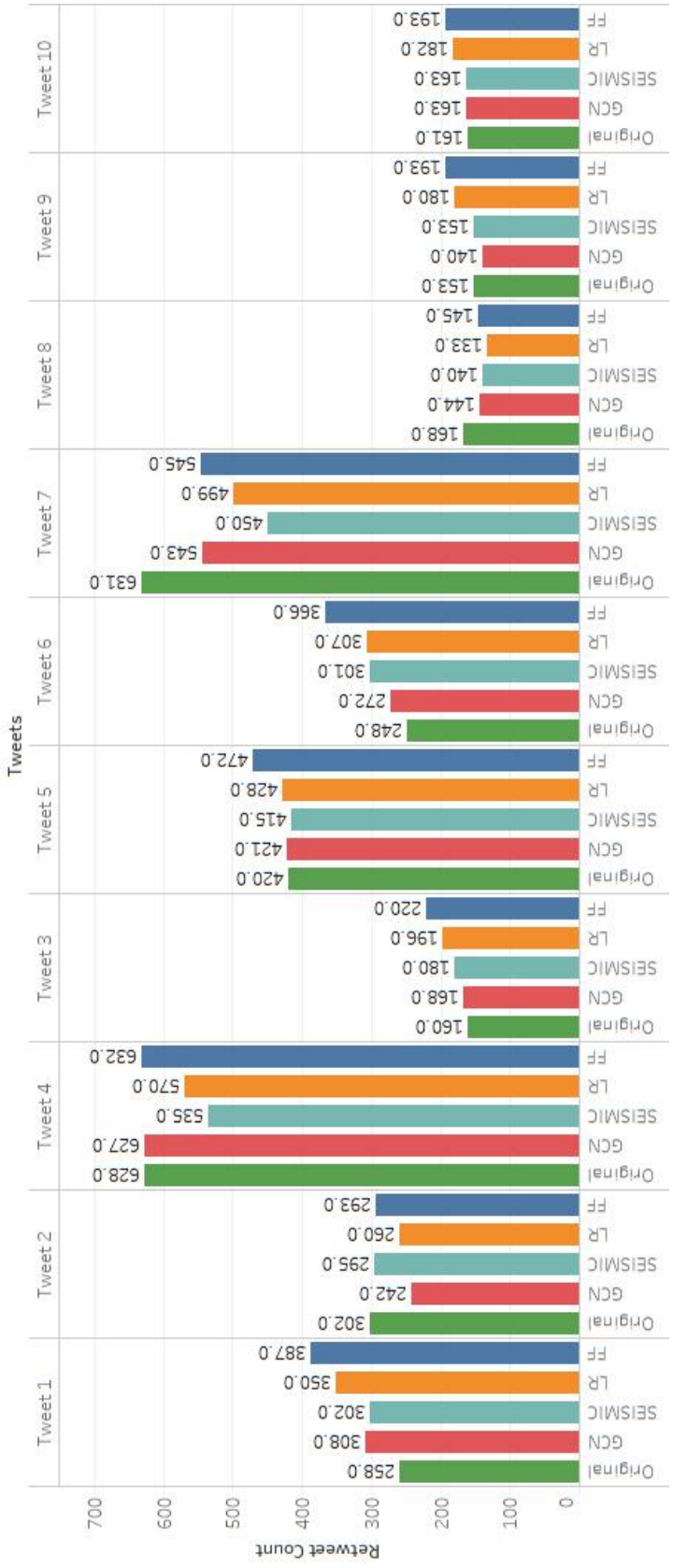

Fig. 6.9. Top ten trending tweets in Kenyan Presidential Elections - Comparison between original retweet count and the predicted values from all the models

6.8.1 Problem statement as classification problem

To check how well the models predict the top 10 percent popular tweets, we classified them into popular and non-popular tweets. To achieve the same, we used threshold to identify their class. The threshold value is the 90th percentile of the predicted values. The viral tweets are represented as 1. The other one is represented as 0. Precision score identifies how many of the the predicted popular(TP) tweets are originally popular(TP, FP). The formula to calculate precision score is given in the equation 6.1.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{6.1}$$

The summary of the results are given in table 6.10.

Table 6.10.

Precision scores for the popular tweets reported by All the Models

Model	**Kenya Elections**	**South Africa Elections**	**Nigeria Elections**
SEISMIC	0.66	0.66	**0.83**
Linear Regression	0.7	0.53	0.8
Feed Forward Neural Network	0.46	0.48	0.54
Graph Convolution Neural Network	**0.77**	**0.72**	0.79

A good statistical model like SEISMIC captures the relation between attributes in detail and build formulas even before starting experiments. Thanks to the well constructed formulas, the results appear positive. From section 6.7, we can see how good is GCN model with regression results. Now we can say it also perform well in identifying popular tweets. It's interesting to see that our GCN model identifies popular tweets 15.38% and 6.71% precise than the SEISMIC model on Kenyan and South African presidential elections dataset.

6.8.2 Spearman's rank correlation

Spearman's coefficient of correlation, ranging between -1 and 1, measures the relationship between two sets of continuous variables that are related to each other. The score of -1 indicates the most negative correlation. The score of 1 indicates the most perfect and positive correlation. We use the same to understand the correlation between predicted and original retweet count values. Table 6.11 shows the Spearman's coefficient scores computed by different models when tested on our datasets. It appears like all the models except feed forward neural network computes correlation scores more alike.

Table 6.11.

Spearman's rank correlation scores reported by All the Models

Model	**Kenya Elections**	**South Africa Elections**	**Nigeria Elections**
SEISMIC	0.77	0.61	0.81
Linear Regression	0.78	0.6	0.77
Feed Forward Neural Network	0.36	0.3	0.38
Graph Convolution Neural Network	0.71	0.53	0.74

6.8.3 Feed forward neural network with spectral embedding of actors adjacency matrix

In our feed forward neural network model, we have kept tweet related, actor related and sentiment related features. They are extracted explicitly from the dataset. Still, it could not capture any connectivity information among actors or tweet sequences. Hence is the reason we moved to Graph Convolution Neural Networks. A suggestion given by the thesis committee allows us to embed connectivity among the actors within feed forward network. First, we construct the adjacency matrix with the same approach discussed in section 5.9.2. It captures relation between actors involved

in dataset. Given adjacency matrix of actors, doing spectral embedding on A allows us to represent information in reduced dimensions. The spectral embedding method internally computes graph Laplacian on the adjacency matrix and interprets the results in reduced size. For every tweet sequence, we concatenate tweet related features along with the information of the author available in the embedded space. We used the same architecture followed in 6.4 and tested in our datasets. The summary of the results are given in table 6.12.

Table 6.12.

Performance Evaluation on Feed Forward Neural Network Model with spectral embedding of actors adjacency matrix

Dataset	**Mean Absolute Error**	**R2 Value**
Kenya Elections	16.12	0.09
South Africa Elections	10.82	0.38
Nigeria Elections	45.89	0.22

It's an experiment to see if the embedding could alter the performance of feed forward neural network model. We observe that there is a way to keep connectivity information in the form of spectral embedding. There may or may not have ways to enhance the performance results given in 6.12. As this experiment is conducted in the last moment, we keep it as an open area to experiment.

7. SUMMARY

In this work, we have proposed machine learning models that extract three types of features from tweet sequences and predict the final number of retweets. Our primary goal was to predict the number based on what we observe from the initial twenty minutes.

We performed experiments on three different datasets that are collected during the Kenyan, South African, and Nigerian presidential elections. For the Linear Regression model, we extracted seven tweet content related features, three user related features and two sentiment related features. The core of the features is the tweet content related features. It refers to the features that are extracted from the behavior of the tweets in the first twenty minutes. As the person posting the tweet may influence people's opinions, we gave enough importance in extracting some information related to the author. Further, we applied simple Natural Language Processing to identify the sentiment expressed in the tweet, and the party possibly supported in the tweet. Once the feature engineering is done, we applied Linear Regression and reported the Mean Absolute Error, R2 Score and Pearson Coefficient scores. We also conducted observations on what features maximized the performance of the model and found tweet related features to be the answer.

Later we discussed how we use the constructed features, and predict results, with the feed forward neural network. We have also covered the architecture of the neural network in detail and the performance of the model, by testing it on our datasets.

Then, we covered the differences in architectures of the previous model and the graph convolution neural network. We also studied the challenges in constructing the optimal adjacency matrix for our regression problem. We have used tweet related features at the feed forward layer, and the actor related information at the GCN layer. Similarly we have shown experimental results obtained with the GCN Model. For

each of the models, We have identified trending tweets and plotted the original count versus the count predicted by that model. We have also witnessed how well the GCN Model is performing when compared to other built models, and the benchmarked SEISMIC model. Later, we altered our problem statement as classification problem to see how much clear these models rank the popular tweets.

The main challenge in a deep learning regression problem lies in identifying the optimal number of hidden layers, choosing appropriate loss functions, activation functions, and learning rate. In addition to the above mentioned challenges, the GCN Model has overweight of adjacency matrix multiplication. Training on a reasonable number of epochs contributes to the accuracy of the model. Hence, it consumes a larger amount of time in training the model. There is not much sense in reducing the adjacency matrix size in order to decrease running time of the model. We can say performance of the model comes with a trade off in the running time of the system. To conclude, GCN model outperforms traditional machine learning models in predicting the retweet count with greater accuracy.

8. RECOMMENDATIONS FOR FUTURE WORK

In this chapter, we will cover some ideas that we assume would be a potential enhancement to what we have done in this thesis. The following list explains the three primary improvements to the current system.

1. Although it's found that tweet related features carry more importance, it would be interesting to extract even more complex features and see how the models behave. Implementation of advanced Sentiment Analytics to understand the tweet content and author would help us possibly create an improvised adjacency matrix.

2. We haven't analyzed much about the limitations of our model. One area that clearly needs to be enhanced is the time required to train the GCN Model.

3. The models are currently tested on gigabyte sized datasets. As the growth of social media is exponential, it will be more appropriate to migrate our programming environment to an in-memory computing framework like Spark. Researches on the Internet have demonstrated implementing deep learning with Spark and TensorFlow.

4. After constructing a graph with nodes representing retweet events in the first twenty minutes for a tweet sequence, we can apply self exciting point process on the nodes to predict the cascade of the nodes over the period of time. It in turn gives the retweet count. Similar ideas have been successfully applied in studies [36] and [37] and with the help of Expectation-Maximization approach and Recurrent Neural Networks.

REFERENCES

REFERENCES

[1] "What is the real impact of social media?" accessed: 2018-05-06. [Online]. Available: https://www.simplilearn.com/real-impact-social-media-article

[2] "News use across social media platforms 2016," accessed: 2018-05-06. [Online]. Available: http://www.journalism.org/2016/05/26/news-use-across-social-media-platforms-2016/

[3] "South africa election 2014: the social media wars," accessed: 2018-05-06. [Online]. Available: http://reprobate.co.za/south-africa-election-2014-the-social-media-wars/

[4] Wikipedia contributors, "Social media in the united states presidential election, 2016 — Wikipedia, the free encyclopedia," https://en.wikipedia.org/w/index.php?title=Social_media_in_the_United_States_presidential_election,_2016&oldid=837839522, 2018, [Online; accessed 7-May-2018].

[5] "Facebook's new initiatives related to elections," accessed: 2018-05-06. [Online]. Available: https://newsroom.fb.com/news/2018/04/new-elections-initiative/

[6] D. Boyd, S. Golder, and G. Lotan, "Tweet, tweet, retweet: Conversational aspects of retweeting on twitter," in *Proceedings of the 2010 43rd Hawaii International Conference on System Sciences*, ser. HICSS '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 1–10. [Online]. Available: http://dx.doi.org/10.1109/HICSS.2010.412

[7] "Twitter retweet prediction," accessed: 2018-05-06. [Online]. Available: https://sites.google.com/site/learningtweetvalue/home

[8] E. Bakshy, B. Karrer, and L. A. Adamic, "Social influence and the diffusion of user-created content," in *Proceedings of the 10th ACM Conference on Electronic Commerce*, ser. EC '09. New York, NY, USA: ACM, 2009, pp. 325–334. [Online]. Available: http://doi.acm.org/10.1145/1566374.1566421

[9] E. F. Can, H. Oktay, and R. Manmatha, "Predicting retweet count using visual cues," in *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, ser. CIKM '13. New York, NY, USA: ACM, 2013, pp. 1481–1484. [Online]. Available: http://doi.acm.org/10.1145/2505515.2507824

[10] R. Bandari, S. Asur, and B. A. Huberman, "The pulse of news in social media: Forecasting popularity," *CoRR*, vol. abs/1202.0332, 2012. [Online]. Available: http://arxiv.org/abs/1202.0332

[11] B. Daróczy, R. Pálovics, V. Wieszner, R. Farkas, and A. A. Benczúr, "Predicting user-specific temporal retweet count based on network and content information." in *INRA@ RecSys*, 2015, pp. 6–13.

[12] A. Kanavos, I. Perikos, P. Vikatos, I. Hatzilygeroudis, C. Makris, and A. Tsakalidis, "Modeling retweet diffusion using emotional content," in *IFIP International Conference on Artificial Intelligence Applications and Innovations*. Springer, 2014, pp. 101–110.

[13] A. Kupavskii, L. Ostroumova, A. Umnov, S. Usachev, P. Serdyukov, G. Gusev, and A. Kustarev, "Prediction of retweet cascade size over time," in *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2012, pp. 2335–2338.

[14] G. Liu, C. Shi, Q. Chen, B. Wu, and J. Qi, "A two-phase model for retweet number prediction," in *International Conference on Web-Age Information Management*. Springer, 2014, pp. 781–792.

[15] L. Hong, O. Dan, and B. D. Davison, "Predicting popular messages in twitter," in *Proceedings of the 20th International Conference Companion on World Wide Web*, ser. WWW '11. New York, NY, USA: ACM, 2011, pp. 57–58. [Online]. Available: http://doi.acm.org/10.1145/1963192.1963222

[16] K. Subbian, B. A. Prakash, and L. Adamic, "Detecting large reshare cascades in social networks," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 597–605.

[17] W. M. Webberley, S. M. Allen, and R. M. Whitaker, "Retweeting beyond expectation: Inferring interestingness in twitter," *Computer Communications*, vol. 73, pp. 229–235, 2016.

[18] Z. Zhou, R. Bandari, J. Kong, H. Qian, and V. Roychowdhury, "Information resonance on twitter: Watching iran," in *Proceedings of the First Workshop on Social Media Analytics*, ser. SOMA '10. New York, NY, USA: ACM, 2010, pp. 123–131. [Online]. Available: http://doi.acm.org/10.1145/1964858.1964875

[19] A. Kupavskii, A. Umnov, G. Gusev, and P. Serdyukov, "Predicting the audience size of a tweet," 2013. [Online]. Available: https://www.aaai.org/ocs/index.php/ICWSM/ICWSM13/paper/view/6077

[20] Z. Yang, J. Guo, K. Cai, J. Tang, J. Li, L. Zhang, and Z. Su, "Understanding retweeting behaviors in social networks," in *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, ser. CIKM '10. New York, NY, USA: ACM, 2010, pp. 1633–1636. [Online]. Available: http://doi.acm.org/10.1145/1871437.1871691

[21] H.-K. Peng, J. Zhu, D. Piao, R. Yan, and Y. Zhang, "Retweet modeling using conditional random fields," in *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*. IEEE, 2011, pp. 336–343.

[22] P. Cui, F. Wang, S. Liu, M. Ou, S. Yang, and L. Sun, "Who should share what?: Item-level social influence prediction for users and posts ranking," in *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '11. New York, NY, USA: ACM, 2011, pp. 185–194. [Online]. Available: http://doi.acm.org/10.1145/2009916.2009945

[23] Z. Luo, M. Osborne, J. Tang, and T. Wang, "Who will retweet me?: finding retweeters in twitter," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2013, pp. 869–872.

[24] T. Zaman, E. B. Fox, and E. T. Bradlow, "A bayesian approach for predicting the popularity of tweets," *CoRR*, vol. abs/1304.6777, 2013. [Online]. Available: http://arxiv.org/abs/1304.6777

[25] M. Gomez-Rodriguez, J. Leskovec, and B. Schölkopf, "Modeling information propagation with survival theory," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ser. ICML'13. JMLR.org, 2013, pp. III–666–III–674. [Online]. Available: http://dl.acm.org/citation.cfm?id=3042817.3043011

[26] J. Zhang, J. Tang, J. Li, Y. Liu, and C. Xing, "Who influenced you? predicting retweet via social influence locality," *ACM Trans. Knowl. Discov. Data*, vol. 9, no. 3, pp. 25:1–25:26, Apr. 2015. [Online]. Available: http://doi.acm.org/10.1145/2700398

[27] Q. Zhang, Y. Gong, J. Wu, H. Huang, and X. Huang, "Retweet prediction with attention-based deep neural network," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, ser. CIKM '16. New York, NY, USA: ACM, 2016, pp. 75–84. [Online]. Available: http://doi.acm.org/10.1145/2983323.2983809

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[29] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3128–3137.

[30] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, June 2014, pp. 1701–1708.

[31] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, Nov 2012.

[32] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue *et al.*, "An empirical evaluation of deep learning on highway driving," *arXiv preprint arXiv:1504.01716*, 2015.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[34] Q. Zhao, M. A. Erdogdu, H. Y. He, A. Rajaraman, and J. Leskovec, "SEISMIC: A self-exciting point process model for predicting tweet popularity," *CoRR*, vol. abs/1506.02594, 2015. [Online]. Available: http://arxiv.org/abs/1506.02594

[35] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[36] E. W. Fox, M. B. Short, F. P. Schoenberg, K. D. Coronges, and A. L. Bertozzi, "Modeling e-mail networks and inferring leadership using self-exciting point processes," *Journal of the American Statistical Association*, vol. 111, no. 514, pp. 564–584, 2016. [Online]. Available: https://doi.org/10.1080/01621459.2015.1135802

[37] S. Xiao, J. Yan, X. Yang, H. Zha, and S. M. Chu, "Modeling the intensity function of point process via recurrent neural networks." in *AAAI*, 2017, pp. 1597–1603.

APPENDIX

A. TOOLS AND TECHNOLOGIES

In this section, We will cover the programming languages and important packages used to implement our problem statement.

1. Python

   A survey released by IEEE Spectrum ranked Python as the top programming language of the year in 2017. Python is a high level object oriented programming language that's simple and easy to use. It comes with many inbuilt libraries that are required to solve problems in Machine Learning, Artificial Intelligence, Image Processing, and Natural Language Processing. We have used Python in all of our three models.

2. R

   R is a language that provides a wide variety of statistical functions, and data manipulation support. Our benchmarked model, SEISMIC, is written in this language. In our research, we have used R to see how the SEISMIC model performs on our datasets.

3. Scikit-learn

   Scikit-learn is a package that contains functions efficient for data analytics and machine learning. It is the heart to most of the regression, supervised and unsupervised learning algorithms. Also, it provides different metrics to evaluate the model. We have used this package to build one of our models and evaluate performance on all the models.

4. TensorFlow

   TensorFlow is a library built especially for high performance computations, especially for deep neural network construction. It supports execution on both CPU and GPU. For each piece of the code written, it forms a computation graph and

then executes it. It has advanced features like saving the model and reloading it later. We have used this package to create two models for our research.

5. Pandas

   Pandas is a data structure within Python which is very simple to use. Pandas DataFrame is a 2D data structure that stores data in structured tabular format with rows and columns labeled. It supports all sort of aggregation, and filtering similar to spreadsheets. In all our models, we have used pandas DataFrame while doing the feature engineering part.