




Universitetet  
i Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

## MASTER'S THESIS

Study programme/specialisation:  Computer Science	Spring / Autumn semester, 2020  Open/ <del>Confidential</del>
Author: Henrik Lessø Mjaaland	 ..... (signature of author)
Programme coordinator: Vinay Jayarama Setty  Supervisor(s): Vinay Jayarama Setty	
Title of master's thesis:  Detecting Fake News and Rumors in Twitter Using Deep Neural Networks	
Credits: 30 ECTS	
Keywords: Deep Learning Neural Networks Fake News Detection Natural Language Processing Sequential Classification Non-Sequential Numeric Classification	Number of pages: 71  Stavanger, 15 June 2020





Faculty of Science and Technology  
Department of Electrical Engineering and Computer Science

# Detecting Fake News and Rumors in Twitter Using Deep Neural Networks

Master's Thesis in Computer Science  
by

Henrik Lessø Mjaaland

Internal Supervisors

Vinay Jayarama Setty

Reviewers

Vinay Jayarama Setty

Avishek Anand

June 7, 2020





*“If you can’t fly,  
then run, if you can’t run,  
then walk,  
if you can’t walk,  
then crawl,  
but whatever you do,  
you have to keep moving forward.”*

Martin Luther King Jr.



# *Abstract*

The scope of this thesis is to detect fake news by classifying them as either real or fake based on article content, metadata, tweets and retweets of news articles from the Politifact data set using graph neural networks.

Fake news generally spread more rapid than real news. This is most likely because fake news are usually more novel or dramatic and contain more superlatives than real news. Tweets of fake news articles, thus, tend to spread exponentially. Fake tweets also tend to have more rumor cascade hops than real news, meaning tweets of fake news are retweeted more than real news. Tweets of real news articles on the other hand, tend to have a more constant and slow spread, and does not reach as many people as fake news overall.

There are generally two characteristics that are used for detecting fake news: article content and rumor path propagation. Most existing works have presented models based solely on one of these characteristics, which has its advantages (i.e. reduced training time), but is also reflected by poor performance results.

This thesis proposes a hybrid model that takes a combination of article content, rumor path propagation in the form of a temporal pattern, and metadata as input using bidirectional LSTM with the Keras Sequential model. Article content is word-embedded using pre-trained GloVe vectors. The metadata, which is continuous, is normalized and discretized. The rumor path propagation time series is computed using dates from metadata related to tweets and retweets.

Some other deep learning and machine learning models are also implemented and tested for comparison. Experimental results demonstrated that the proposed model performs significantly better than all of these models.





# *Acknowledgements*

This thesis is the culmination of my master's degree in Computer Science and the result of my fascination for computers and social media. I took an interest in computers at a young age, and what first sparked my interest in computers is how small they make the world. At an age of eight, I was already chatting with people from all around the world. As social media emerged, the world was made even smaller. Information was made more accessible to the public as social media can be used to share information, but they can also be used to share fake news. Recently, I have taken an interest in deep neural networks. What interests me about deep neural networks is that they can perform very well when the data size is large enough, they can combine features without being explicitly told to do so, thus, simplifying the process of feature engineering, and they can be used to solve complex problems such as speech recognition, image classification, and natural language processing which can be used for identifying fake news as is done in this thesis. I would like to express my deepest gratitude to Vinay Jayarama Setty for introducing me to deep neural networks and giving me the necessary tools to write this thesis, and for guiding me throughout the semester by giving valuable feedback. I would also like to thank my friends and family for motivating and supporting me.

# Contents

<b>Abstract</b>	<b>vi</b>
<b>Acknowledgements</b>	<b>viii</b>
<b>Abbreviations</b>	<b>xi</b>
<b>Symbols</b>	<b>xiii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	2
1.2 Problem Definition . . . . .	3
1.3 Challenges . . . . .	4
1.4 Contributions . . . . .	4
1.5 Outline . . . . .	5
<b>2 Background</b>	<b>7</b>
2.1 Neural Networks . . . . .	7
2.1.1 Feed-Forward (Forward-Propagation) . . . . .	8
2.1.2 Back-Propagation . . . . .	8
2.1.3 Gradient-Descent . . . . .	9
2.2 Neural Networks for Natural Language Processing . . . . .	9
2.2.1 Term Frequency-Inverse Document Frequency (TF-IDF) . . . . .	9
2.2.2 Word Embedding . . . . .	9
2.2.3 Recurrent Neural Networks . . . . .	10
2.2.4 Gated Recurrent Unit (GRU) . . . . .	10
2.2.5 Long Short-Term Memory (LSTM) . . . . .	11
2.2.6 Vanishing Loss Gradient . . . . .	13
2.2.7 Bidirectional Long Short-Term Memory (BiLSTM) . . . . .	13
2.3 Related Work . . . . .	14
2.3.1 Machine Learning-based Works . . . . .	14
2.3.2 Deep Learning-Based Works . . . . .	15
2.4 Existing Approaches . . . . .	16
2.5 Analysis . . . . .	17

<b>3</b>	<b>Solution Approach</b>	<b>19</b>
3.1	Proposed Model . . . . .	19
3.2	Preprocessing . . . . .	19
3.2.1	Natural Language Processing (NLP) . . . . .	20
3.2.2	Continuous Attributes . . . . .	20
3.2.3	Rumor Propagation . . . . .	21
3.3	Layers . . . . .	22
<b>4</b>	<b>Experimental Setup and Data Set</b>	<b>25</b>
4.1	Technologies . . . . .	25
4.1.1	Natural Language ToolKit (NLTK) . . . . .	25
4.1.2	Wordcloud . . . . .	25
4.1.3	Keras . . . . .	26
4.1.4	Regular Expressions (RE) . . . . .	26
4.1.5	Datetime . . . . .	26
4.2	Dataset . . . . .	26
4.2.1	Synthetic Minority Over-sampling Technique (SMOTE) . . . . .	27
4.2.2	Undersampling . . . . .	27
4.2.3	Gridsearch . . . . .	27
4.3	Experimental Results . . . . .	28
4.3.1	Cumulative Distribution Function (CDF) . . . . .	28
4.3.2	Users . . . . .	28
4.3.3	TFIDF . . . . .	28
4.3.4	Rumour Propagation . . . . .	31
4.4	Statistical Measures . . . . .	31
4.5	Experimental Setup . . . . .	34
4.5.1	Support Vector Machine (SVM) . . . . .	35
4.5.2	Naive Bayes (NB) . . . . .	35
4.5.3	Convolutional Neural Networks (CNN) . . . . .	36
4.5.4	Long Short-Term Memory (LSTM) . . . . .	36
4.5.5	Optimization Techniques . . . . .	36
<b>5</b>	<b>Discussion</b>	<b>39</b>
5.1	Baseline Comparison . . . . .	39
5.2	Implications of Findings . . . . .	43
<b>6</b>	<b>Conclusion and Future Directions</b>	<b>45</b>
6.1	Future Directions . . . . .	45
6.2	Conclusion . . . . .	46
	<b>List of Figures</b>	<b>46</b>
	<b>List of Tables</b>	<b>49</b>
	<b>Bibliography</b>	<b>51</b>

# Abbreviations

<b>NN</b>	<b>N</b> eural <b>N</b> etworks
<b>DNN</b>	<b>D</b> eep <b>N</b> eural
<b>RNN</b>	<b>R</b> ecurrent <b>N</b> eural <b>N</b> etworks
<b>GRU</b>	<b>G</b> ated <b>R</b> ecurrent <b>U</b> nit
<b>LSTM</b>	<b>L</b> ong <b>S</b> hort <b>T</b> erm <b>M</b> emory
<b>BiLSTM</b>	<b>B</b> idirectional <b>L</b> ong <b>S</b> hort <b>T</b> erm <b>M</b> emory
<b>CNN</b>	<b>C</b> onvolutional <b>N</b> eural <b>N</b> etworks
<b>TF</b>	<b>T</b> erm <b>F</b> requency
<b>IDF</b>	<b>I</b> nverse <b>D</b> ocument <b>F</b> requency
<b>NLP</b>	<b>N</b> atural <b>L</b> anguage <b>P</b> rocessing
<b>DL</b>	<b>D</b> eep <b>L</b> earning
<b>ML</b>	<b>M</b> achine <b>L</b> earning
<b>SMOTE</b>	<b>S</b> ynthetic <b>M</b> inority <b>O</b> ver-sampling <b>T</b> echnique
<b>CDF</b>	<b>C</b> umulative <b>D</b> istribution <b>F</b> unction
<b>SVM</b>	<b>S</b> upport <b>V</b> ector <b>M</b> achine
<b>NB</b>	<b>N</b> aive <b>B</b> ayes
<b>NLTK</b>	<b>N</b> atural <b>L</b> anguage <b>T</b> ool <b>K</b> it
<b>RE</b>	<b>R</b> egular <b>E</b> xpressions
<b>TP</b>	<b>T</b> rue <b>P</b> ositives
<b>TN</b>	<b>T</b> rue <b>N</b> egatives
<b>FP</b>	<b>F</b> alse <b>P</b> ositives
<b>FN</b>	<b>F</b> alse <b>N</b> egatives
<b>RMSProp</b>	<b>R</b> oot <b>M</b> ean <b>S</b> quare <b>P</b> ropagation
<b>AdaGrad</b>	<b>A</b> daptive <b>G</b> radient
<b>AdaM</b>	<b>A</b> daptive <b>M</b> oment



# Symbols

Symbol	Name
$\Sigma$	Sum
$\int$	Integral
$\infty$	Infinity
Y	Node Output
W	Weight
I	Node Input
b	Node Bias
Z	Activation Function
E	Total Error
H	Hidden State
T	Target Value
L	Learning Rate
C	Cell Memory
f	Forget Gate
X	Continuous Feature Value
F	Function Output
N	Number of Neurons
TP	True Positives
FP	False Positives
TN	True Negatives
FN	False Negatives
f	Class Probability Density Function
$\mu$	Mean
s	Standard Deviation





# Chapter 1

## Introduction

According to the Cambridge Dictionary, fake news are "stories that appear to be news, spread on the internet or using other media, usually created to influence political views or as a joke"[[Cam\(2020\)](#)]. Fake news is a problem of increasing concern in today's world. Fake news today spread like wildfire. They are usually spread by bots exponentially in social media. Creating news by manipulating videos and images is getting easier and easier. 'Anyone' can forge their own news today. Lately, fake news has become a problem of concern especially in Norway, where sharing fake news through memes on social media has become a trend.

Recently, there have been a lot of fake news regarding the Corona Virus from China. The most widespread ones are that the Chinese were infected from eating bats[[Reid\(2020\)](#)], and that there is already a vaccine for the virus[[Pol\(2020a\)](#)], which is not true, since there are seven kinds of Corona Viruses, and the one causing the outbreak now is a new kind of Corona Virus.

When it comes to fake news during elections, attention has mostly been diverted to the West, leaving the global south unchecked[[Chinchilla\(2019\)](#)]. This has led to an increase of forty percent spread of fake news in India during election times.

In Brazil, 2019, the vast majority of false information shared on WhatsApp in Brazil during the presidential election favoured the far-right winner, Jair Bolsonaro[[Avelar\(2019\)](#)]. Out of 11957 viral messages shared across 296 group chats on the instant-messaging platform in the campaign period, approximately forty-two percent of rightwing items contained information found to be false by factcheckers. On the southern hemisphere, the preferred messaging app is WhatsApp. WhatsApp offers encrypted peer-to-peer messaging and is very challenging to monitor, unlike apps like Facebook. Monitoring users' conversations violates the users' privacy, but is useful for detecting fake news and their source.

Multiple initiatives have been taken to battle fake news such as the French law against

information manipulation[[gov\(2018\)](#)]. The law prevents influencing of election results, which was done e.g. during Brexit, when newspapers incited hatred against immigrants and the EU[[Danzig\(2017\)](#)].

The law prevents political influencing by requiring “a ‘transparency obligation for digital platforms’, who must report any sponsored content by publishing the name of the author and the amount paid. Platforms exceeding a certain number of hits a day must have a legal representative in France and publish their algorithms.”[[gov\(2018\)](#)]. The law also requires a judge to qualify fake news based on the following three criterias:

- The fake news must be manifest.
- The fake news must be disseminated deliberately on a massive scale.
- The fake news must lead to a disturbance of the peace or compromise the outcome of an election.

The law also demands a cooperation between the different digital platforms during election periods. The French Broadcasting Authority is responsible for enhancing law enforcement for this requirement. Furthermore, Emmanuel Hoog, former president of the French Press Agency was entrusted with producing a press ethics body.

Measures against fake news have also been taken in Finland, Malaysia and Singapore. Finland launched an anti-fake news initiative that aims to teach how to identify fake news already in 2014[[Mackintosh\(2018\)](#)]. The Malaysian government passed the Anti-Fake News Act in 2018[[Buchanan\(2018\)](#)]. The Act was however accused of seeking to stifle criticism of the administration. In 2019, another law against fake news was passed in Singapore called the Protection from Online Falsehoods and Manipulation Act. This law received the same allegations as the law against fake news from Malaysia[[Newton\(2019\)](#)].

## 1.1 Motivation

Fake news has had a massive impact on the news industry since the dawn of the Internet. Journalists are under increasing pressure to produce more material faster, and today they tend to use social media as a source of information[[Nordheim\(2018\)](#)]. When using social media as a source of information, verifying the authenticity of the news is a big challenge for reasons such as the pressure to produce more. When journalists incorrectly verify news sources, their brand’s reputation is tarnished and people lose trust in the news media. A brand’s reputation takes years to build up, but only seconds to tear down. The pressure to produce material fast also leads to less serious journalism.

Fake news is a many-faced and complex issue. They can be spread to fuel conflicts, for economic gains, defamation or for political purposes. Also, fake news lead to people in general being less well informed.

In 2019, fake news regarding the death of twenty two year old Chinese student Alex Chow circulated, claiming that he had not committed suicide, but was either chased or pushed off a parking garage by the Hong Kong police force[Chi(2019)]. The news also claimed that officers had blocked off an ambulance from reaching Chow. The purpose of these fake news was to incite anti-government protests.

In 2013, fake news regarding two explosions in the White House that had injured the earlier US president Barack Obama led to the Dow Jones Industrial Average dropping by 143.5 points, which corresponds to the sum total of 130 billion US dollars in stock value[Cheo(2018)]. Donald Trump legislated CNN and Washington Post as "fake news" because they were not supportive of him (they referred to his followers as a cult and claimed that he had "bewitched" the republican party on multiple occasions)[Jardine(2018)].

Fake news has always been used in political contexts, such as spreading false poll numbers to discourage people from voting during political elections, e.g. in Mexico in 1988 by PRI (Institutional Revolutionary Party), but never on a scale and magnitude as with today's technology. During the 2016 presidential election in USA, which ended in the election of Donald Trump as the US president, there was a surge of fake news on social media. Investigations after the presidential election points to Russian influence during the campaign[Abrams(2019)]. Another case of using fake news as a means of political influence, is back in 2014 when ISIS started sharing propaganda on every social media imaginable. Democracy is built trust in the public's capacity for reasoned communication. Digital technologies have made information more accessible to the public, however as deep fake news is getting more and more advanced, people are getting less informed, thus democracy is threatened. Political influencing, which was done by PRI, Russia and Bolsonaro as discussed above, also threatens democracy by centralizing the power and taking it away from the people.

## 1.2 Problem Definition

This thesis investigates if metadata, article content and rumor propagation paths can be utilized to improve fake news detection. The goal is to achieve this by modelling a graph neural network in order to classify the articles as either real or fake based on patterns in article content, metadata and sharing of tweets. The metadata consists of a series of descriptive parameters for both the articles and their respective tweets and retweets such

	created_at	text	contributors_enabled_user	...	followers	following	label
0	2015-10-09 21:18:53+00:00	Missed @marcorubio in NH this week? Come to a ...	False	...			real
1	2013-03-22 06:36:15+00:00	2013-HB-4469 Public employees and officers; et...	False	...			real
2	2008-03-03 06:10:37+00:00	Strong Words in Ohio as Obama and Clinton Pres...	False	...			real
...	...	...	...	...	...	...	...
790	2017-08-29 17:54:38+00:00	Floyd Mayweather Jr. donates a whopping sum of...	False	...			fake
791	2018-02-21 18:47:48+00:00	@TylerHuckabee @LagBeachAntifa9 @AntifaNantuck...	False	...	1140757886706102273, 289419759, 1450422866, 11...	1140757886706102273, 289419759, 1450422866, 11...	fake
792	2018-02-23 18:45:47+00:00	Obama Announces Bid To Become UN Secretary Gen...	False	...			fake

Figure 1.1: Sample Data Set

as article id, tweet and retweet identification, tweet- and retweet content, and follower count.

### 1.3 Challenges

There are several challenges when identifying fake news. Fake news platforms usually imitate real news platforms by design and content.

Fake news articles are not always one hundred percent fake, they might have some true statements mixed with false statements.

With today's technology, images and videos can be fabricated, making it impossible to instantly verify the authenticity of news.

It is also easy to create fake entities to share fake news. There are free tools available online for this purpose, and existing photos can be used for face swap[Polyakov(2018)].

Fake news articles tend to have either novel or dramatic content, and they often contain more superlatives and loaded words (to play on emotions) than real news, hence, natural language processing (NLP) should be viable for identifying fake news. NLP is however a complex and time-consuming process as words can have different meanings in different contexts, and every language and dialect requires a separate version of NLP.

The amount of data available is limited, and there are more real than fake news available. This affects training of classification models.

### 1.4 Contributions

In the past, classification and regression models have mostly been used for fake news detection. This thesis, however, presents a deep neural network model based on Bidirectional Long Short-Term Memory (LSTM) which takes article content, rumor propagation

paths, in the form of time series, and metadata as input. Article content is word embedded. Continuous numeric attributes are normalized in the range zero to one, and then discretized. Time series are constructed based on rumor propagation paths. Finally the word embedded article content, the discretized numeric attributes, and the time series are concatenated and fed into the model which classifies news articles from the Politifact dataset as either real or fake.

CNN, C-LSTM (based on the C-LSTM model from

[Khan et al.(2019)Khan, Khondaker, Islam, Iqbal, and Afroz], using metadata and rumor path propagation in addition to article content), CNN+LSTM (based on the CNN+GRU model from [Liu and Wu(2018)], using article content and metadata in addition to rumor path propagation), Multinomial NB, SVM and the models from [Liu and Wu(2018)] and [Khan et al.(2019)Khan, Khondaker, Islam, Iqbal, and Afroz] with the best performance results are also implemented and tested for comparison.

## 1.5 Outline

**Chapter 2, Background**, introduces theories related to the work done in this thesis including neural networks. It also introduces related works.

**Chapter 3, Solution Approach**, describes the method used in the proposed model in detail, and explains why the given method is used.

**Chapter 4, Experimental Setup and Data Set**, presents applied technologies, the Politifact dataset, the experimental results, accuracy measures, and the experimental setup.

**Chapter 5, Discussion**, compares the proposed model with baseline models and discusses implications of the findings.

**Chapter 6, Conclusion**, and Future Directions points to future directions, and answers the research question.



## Chapter 2

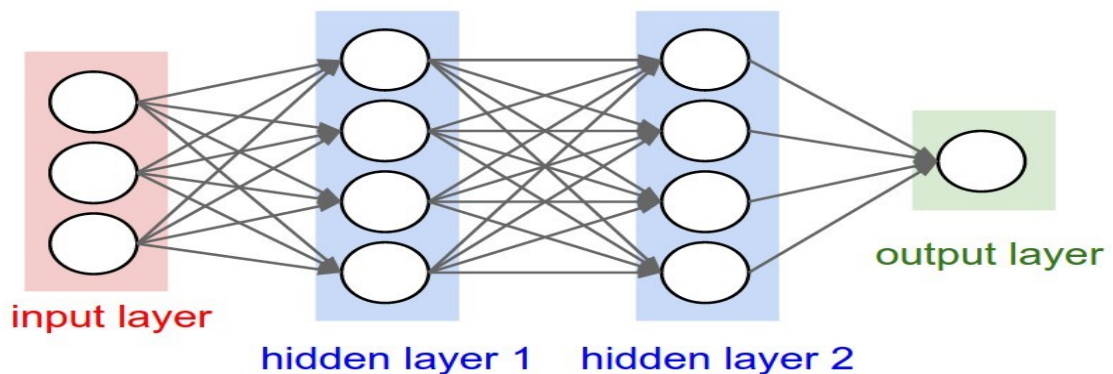
# Background

This chapter introduces relevant theories and baseline solutions for detecting fake news.

### 2.1 Neural Networks

Neural networks are based on the human brain in the sense that they are algorithms consisting of multiple nodes connected by edges, mimicking neurons[Panchal(2018)]. Neural networks with multiple hidden layers, are referred to as deep neural networks. Hidden layers are the layers between the input layer and the output layer (see Figure 2.1<sup>1</sup>). As of now, the required number of hidden layers for a neural network be considered ‘deep’ is more than one, but this number will likely increase soon. Deep neural networks enable computers to artificially learn, hence the term, Artificial Intelligence (AI)[Marr(2018)]. Deep neural networks can for example be used to learn features from data such as text, images, sound or they can be used for classification and regression.

<sup>1</sup><https://i.stack.imgur.com/Kc50L.jpg>



**Figure 2.1:** Neural Networks Layers

As mentioned above, neural networks have many uses, but in this case the target values are either real or fake, which means that this is a binary classification problem. All the nodes in the hidden layers and the output layer, have their own classifier. The classifiers activate nodes based on the inputs from the previous layer and an activation function. There are many activation functions, depending on numerous factors. For this thesis, ReLU (Rectified Linear Unit) followed by Sigmoid is used. ReLU is the most common activation function.

### 2.1.1 Feed-Forward (Forward-Propagation)

The first hidden layer receives input from the input layer, and then classification scores are passed on from layer to layer. Only activated nodes pass on their classification scores to the next layer. In the output layer, the target values (in this case real or fake) are classified based on the final classification scores, which means that the output is the predicted class. This process is referred to as feed-forward or forward-propagation. Node outputs,  $Y_i$ , are computed by first calculating node nets,  $Y_i$ , which is the sum of the input weights ( $W_{ij}$ ), multiplied with their respective inputs ( $I_{ij}$ ), and biases ( $b_i$ ) [[Hansen\(2019\)](#)]:

$$Y_i = \Sigma(W_{ij} * I_{ij} + b_i) \quad (2.1)$$

Then, if the target classes are not linearly separable, the activation functions,  $Z_i$ , are given by:

$$Z_i = \frac{1}{1 + E^{-Y_i(H_i)}} \quad (2.2)$$

, where  $E$  is the total error, and  $H_i$  are hidden nodes.

### 2.1.2 Back-Propagation

Back-propagation is the process of computing the gradients with respect to the weights (see equation 2.4). While forward-propagation is used for computing outputs, the purpose of back-propagation is to minimize the loss function. The loss function represents the differences between predicted values and target values. This is done by adjusting the weights and biases backwards. The weights are adjusted by first computing the total error ( $E$ ) [[Mazur\(2015\)](#)]:

$$E = \Sigma(0.5 * (T(H_i) - Z_i)^2) \quad (2.3)$$

, where  $T$  is the target value. Next, the learning rate ( $L_i$ ) multiplied with the derivative of the total error w.r.t weights are subtracted from the given weights:



$$\frac{dE}{dW_i} = \frac{dE}{dZ_{i+1}} * \frac{dZ_{i+1}}{dY_i} * \frac{dY_i}{dW_i} \quad (2.4)$$

, where

$$W_i = W_i - (L_i * \frac{dE}{dW_i})$$

The learning rate that gives the steepest loss function should be picked, or one can gradually reduce it after training epochs.

### 2.1.3 Gradient-Descent

For an unweighted neural network, a given set of inputs will always result in the same output. In a weighted neural network however, all the nodes have different weights, each resulting in a unique output. The gradient descent algorithm runs forward- and back-propagation multiple rounds or epochs, each time tweaking the weights in order to improve the classification accuracy as explained above[S.(2018)]. Running forward propagation with a high number of epochs is time-consuming and one should find a balance between accuracy and number of epochs, where accuracy is maximized, and number of epochs is minimized. Too many epochs can also result in overfitting. This can be prevented by plotting training- and testing loss. Overfitting is when the training loss is less than the testing loss, and underfitting is when the training loss is higher than the testing loss. The testing- and training loss should be about the same.

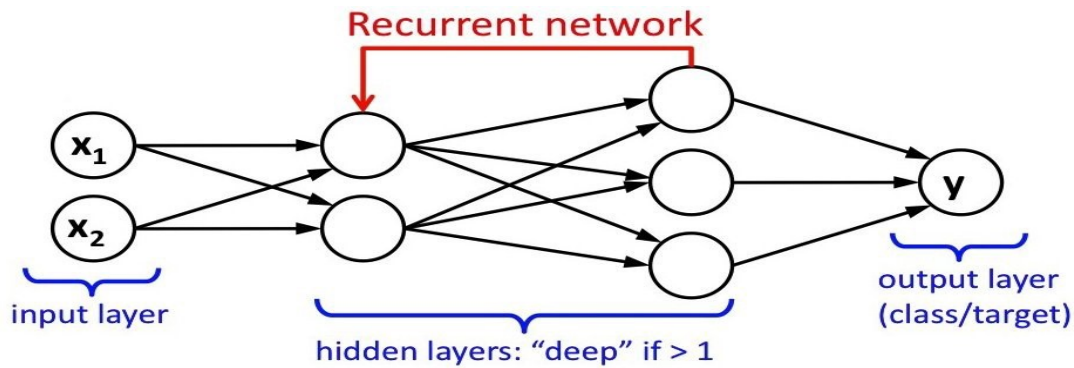
## 2.2 Neural Networks for Natural Language Processing

### 2.2.1 Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF is the product of TF and IDF. TF is the number of terms across all the documents for each term. IDF is the inverse number of terms per document. TF-IDF says how rare a given term is. The higher the TF-IDF corresponding to a term, the more rare the given term is. The most common words will thus have a low TF-IDF.

### 2.2.2 Word Embedding

Word embedding is the process of converting terms to vectors of float numbers [Agrawal(2019)]. This is done to improve the accuracy of Artificial Neural Networks (ANN), because ANNs are built for continuous numeric inputs, and see similarities



**Figure 2.2:** Recurrent Neural Network with Loops

between continuous vectors with more ease than similarities between terms. A common technique for word embedding is GloVe (Global Vectors). GloVe is an algorithm for converting unlabelled data such as words to continuous vectors which reduce dimensionality[Pennington(2014)]. GloVe vectors are pre-trained on Wikipedia and Gigaword 5, and are good at capturing semantics.

### 2.2.3 Recurrent Neural Networks

Recurrent neural networks consist of multiple copies of a feedforward neural network (FNNs), and each copy is considered a time-step[Brownlee(2017a)]. Recurrent neural networks use the gradient descent algorithm to minimize error. However, since there are many time-steps that all share the same parameters, backpropagation-through-time (BPTT) is used instead of normal backpropagation. There are loss between time steps, thus, there is an extra error gradient compared to normal backpropagation.

Traditional neural networks are not good for predicting based on sequential data such as speech, time series, or text, because they have no memory, and sequential data is ordered. Recurrent neural networks introduce memory by using loops between the hidden layers. The output from the loops are stored in the internal state (see figure 2.2<sup>2</sup>).

### 2.2.4 Gated Recurrent Unit (GRU)

GRU is an RNN architecture that controls flow of information by using the reset and update gates as depicted in figure 2.3<sup>3</sup>. The reset gate decides which information to use and which information to throw away, and thus, it can be considered a mix of the LSTM forget and input gates[Brownlee(2017b)].

<sup>2</sup><https://i.stack.imgur.com/Kc50L.jpg>

<sup>3</sup>[https://d2l.ai/chapter\\_recurrent-modern/gru.html](https://d2l.ai/chapter_recurrent-modern/gru.html)

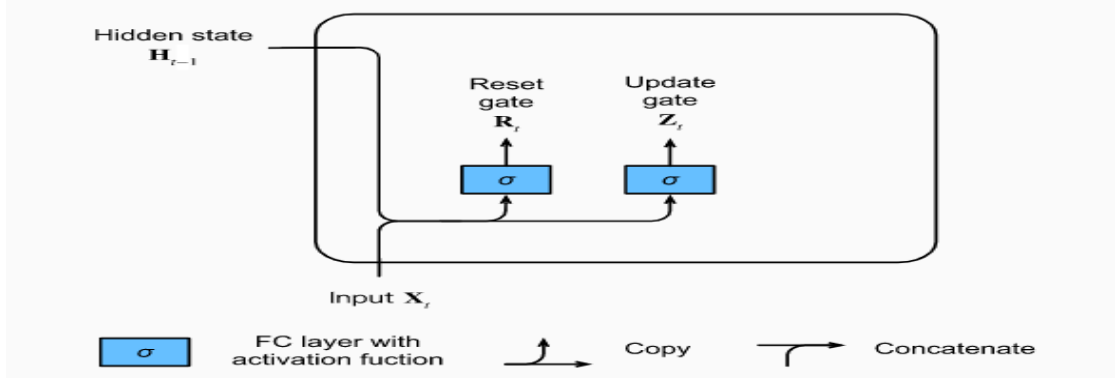


Figure 2.3: GRU Cell

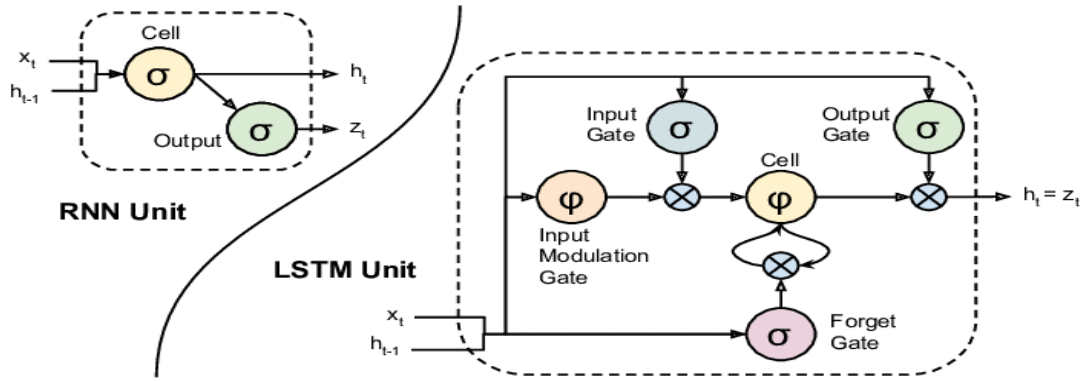


Figure 2.4: RNN Cell and LSTM Gates

### 2.2.5 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a type of recurrent neural network that solves the vanishing gradient problem by introducing long-term memory. As discussed in 2.2.3, recurrent neural networks have multiple time-steps and a cell which contains one network layer ( $\text{Tan}_h$ ) for each time-step. LSTM networks on the other hand have a memory cell which contains four network layers for each time-step (see figure 2.4<sup>4</sup>). Three of the layers are all Sigmoid, namely, the input-, output-, and forget gate. There is also one  $\text{Tan}_h$  layer.

The input gate chooses which information to remember, and the output gate decides which information to output. LSTM memory cells have two states, namely, hidden- and memory state. The hidden state is the same as the output. The memory state,  $C$ , is where memory input is stored. The equations of the different gates and states are explained in figure 2.5<sup>5</sup> below.

<sup>4</sup>[Donahue et al.(2014)Donahue, Hendricks, Guadarrama, Rohrbach, and Venugopalan]

<sup>5</sup><https://medium.com/deep-math-machine-learning-ai/chapter-10-1-deeplp-lstm-long-short-term-memory-networks-with-math-21477f8e4235>

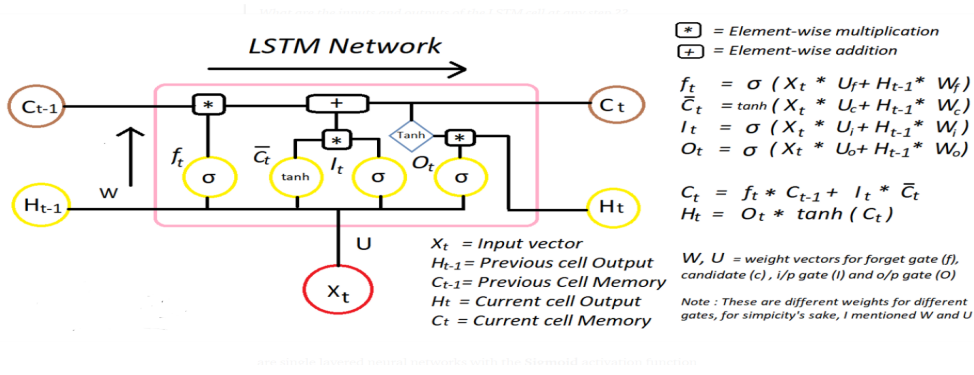


Figure 2.5: LSTM Cell Equations

The key to long-term memory is the forget gate. In traditional RNNs, all inputs are accepted. In this case, previously accepted relevant information might be substituted with new irrelevant information. In LSTMs however, the forget gate prevents this problem by filtering irrelevant information from the memory state, thus providing long-term memory[Phi(2018)]. This is done by multiplying the previous memory state with the forget gate,  $f$ . If  $f$  is zero, the memory state is completely forgotten, and only the current input ( $C'_t * I_t$ ) is used:

$$C_t = C_{t-1} * f_t + C'_t * I_t \quad (2.5)$$

Memory cells also have an input modulation gate (this gate is often considered a sub-gate of the input gate). This gate normalizes the input information to increase convergence speed.

RNNs (including LSTM) use an activation function to compute outputs,  $H_t$ , where  $t$  is timesteps. A common activation function is  $\tanh(H_t)$  (see figure 2.5), and it ensures that all the values stay between negative one and one:

$$H_t = \tanh(C_t) \quad (2.6)$$

The proposed model however uses the ReLU activation function in the early layers:

$$H_t = \max(0, x) \quad (2.7)$$

ReLU is used to avoid the vanishing gradient problem as explained in the next section. In the last two layers the Sigmoid activation function is used. Sigmoid is more suited for binary classification such as fake news detection than  $\tanh$ , because Sigmoid transforms value to the range zero to one:

$$H_t = \frac{1}{1 + e^{-x}} \quad (2.8)$$

The Sigmoid outputs are target class probabilities.

A con with LSTM is that it can cause redundant overhead because some predictions requires less context than others. An example is “the sun is a star”, when predicting the last word, ‘star’, no more information is needed. An example where more context is needed, and LSTM is not redundant, is “I like to watch soap operas... My favourite TV-show is ‘Friends’”. In this case the first statement is useful when predicting the last word of the second statement.

### 2.2.6 Vanishing Loss Gradient

When training RNNs with multiple layers a problem called the ‘Vanishing Loss Gradient’ can arise. The problem is as the name suggests, when the gradient ‘vanishes’, meaning that it’s close to zero[Wang(2019)]. The problem arises if many gradients are close to zero, because the gradients are products of previous gradients. The problem usually occurs in the earlier layers (gradients are computed using backpropagation). This is a problem because the weight updates are the subtraction of the gradients multiplied with the learning rate, from the weights themselves. This means that if the gradients are close to zero, the weight updates will have little to no effect at all. The vanishing loss gradient is not a problem for the ReLU activation function, hence the proposed model applies ReLU in the earlier layers and Sigmoid in later layers (see chapter 2.1). ReLU converts values above zero to the original value, and negative numbers to zero, while Sigmoid converts values above zero to one, and negative numbers to zero, and is thus more computationally taxing than ReLU. This also alleviates the exploding loss gradient problem, which is the opposite of the vanishing loss gradient problem, when the gradient reaches astronomically high values, because the gradient can never be higher than one.

### 2.2.7 Bidirectional Long Short-Term Memory (BiLSTM)

Forward LSTMs only have backward memory and remember the past, while backward LSTMs are trained on reversed inputs, and thus have forward memory and remember the future[Brownlee(2020a)]. Bidirectional LSTMs combine forward and backward LSTMs, which means it has both forward and backward memory. This can improve performance by providing more context and reducing training time. Consider the following scenario: a forward LSTM’s memory contains ‘The boys went to...’, while a backward LSTM’s

memory contains ‘...and then they caught two fish’. Predicting the next word, ‘fish’, is easier when combining the memories, than it is when only using the forward LSTM’s memory.

## 2.3 Related Work

The last few years, there has been a lot of research on detecting fake news in social media. Many of these works have used machine learning techniques like decision trees and linear Support Vector Machine (SVM) as [A.Lakshmanarao(2019)] and [Khan et al.(2019)Khan, Khondaker, Islam, Iqbal, and Afroz]. The works achieved prediction accuracies for fake news detection on social media of 82.7% and 65% respectively using decision trees. When using SVM, accuracies of 75.5% and 66% was achieved. Some of these works focus on article content like [Ibrain and Lloret(2019)], which achieved an accuracy of 62.4% using Fakebox (a machine learning model) and McIntire’s fake-real-news-dataset. Lately, research has also been conducted using deep learning. These works focus mostly on rumour propagation paths, and not so much on content, i.e. [Liu and Wu(2018)]. [Liu and Wu(2018)] claims to be able to detect fake news with accuracies of 85% and 92% on Twitter and Sina Weibo respectively in five minutes after they begin to spread. A graph neural networks-based work focusing on image data, obtained an accuracy of 94%, using a Convolutional Neural Networks (CNN) based model [Ibrain and Lloret(2019)]. [Ibrain and Lloret(2019)] also obtained an accuracy of 91% using LSTM based on content and title of articles. The work used fake news articles from the dataset, ‘Getting Real About Fake News’, and real news articles from sources such as ‘The New York Times’ and ‘The Washington Post’. Some works have also applied TF-IDF for fake news detection such as [Khan et al.(2019)Khan, Khondaker, Islam, Iqbal, and Afroz], where an accuracy of 95% was achieved using unigram Naïve Bayes on a combination of the ‘Liar Liar’- and ‘Fake or Real News’ datasets. The results mentioned indicate that fake news detection are generally most successful using deep learning (especially LSTM) as compared to machine learning algorithms, but also that machine learning algorithms can be better for small data sets. The related works are described further below.

### 2.3.1 Machine Learning-based Works

- **An Efficient Fake News Detection System Using Machine Learning**  
[A.Lakshmanarao(2019)] detects fake news using the following machine learning classifiers: SVM, K-Nearest Neighbors (KNN), Decision tree, Random forest (RF).

The best accuracy was obtained using NLP algorithms and a RF Classifier on the data set from the Kaggle Fake News Challenge. The reason machine learning algorithms was even better than deep learning models in this case, might be because deep learning models requires padding or shrinking the data, or maybe because the 'Liar liar' data set contains statements, and not full articles. The data set contains a number of articles as well as article id, title, author and label.

- **A Benchmark Study on Machine Learning Methods for Fake News Detection**

[Khan et al.(2019)Khan, Khondaker, Islam, Iqbal, and Afroz] detects fake news using a the following machine learning classifiers: SVM, LR, Decision Tree, Adaboost, Naive Bayes and K-Nearest Neighbours. A combination of the 'Liar liar pants on fire' dataset and the Kaggle Fake News competition data set was used. The 'Liar liar' data set contains statements from Politifact.com. The best accuracy achieved was 95%, using a Multinomial Naive Bayes (NB) classifier with unigram TF-IDF features. The dissertation recommends using n-gram TF-IDF features with NB for small data sets and unigram for large data sets and claims that LSTM classifiers are better at overcoming overfitting than NB classifiers.

### 2.3.2 Deep Learning-Based Works

- **Fake news detection using Deep Learning**

[Ibrain and Lloret(2019)] detects fake news using LSTM,CNN and Bidirectional Encoder Representations from Transformers (BERT). BERT, using Google's 'BASE' architecture and 'Wordpiece' word tokenization, achieved the best results, and CNN had slightly better accuracy than LSTM. BERT is a network-based technique for NLP. The fake news articles are gathered from the dataset 'Getting Real About FakeNews', and the real articles are gathered from esteemed sources such as 'The New York Times' or 'The Washington Post'.

- **Early Detection of Fake News on Social Media Through Propagation Path Classification with Recurrent and Convolutional Networks**

[Liu and Wu(2018)] detects fake news early after the articles are tweeted on Twitter using CNN, RNN, SVM, Difficult-to-Treat Resistance (DTR), Gated Recurrent Unit (GRU), RF, Propagation Tree Kernel (PTK) for propagation path classification. Three datasets containing social media data were used: Sina-Weibo, Twitter 2015 and Twitter 2016. The proposed model in the dissertation is a combination of RNN (GRU) and CNN using max pooling. RNN alone achieved slightly better results than CNN on it's own, but the difference is negligible.

- **A Benchmark Study on Machine Learning Methods for Fake News Detection**

[Khan et al.(2019)Khan, Khondaker, Islam, Iqbal, and Afroz] implements and assesses various machine learning- and deep learning methods for fake news detection. McIntire's 'Fake or Real News' dataset, Wang's 'Liar liar, pants on fire' data set, and a combination of the two were used. The work concluded that even though deep learning models, namely, C-LSTM (CNN-LSTM) and bidirectional LSTM, produce the best results, machine learning algorithms (linear SVM) might be best for small data sets with the right feature selection considering training speed.

## 2.4 Existing Approaches

The proposed model in this thesis takes metadata, rumor propagation paths in the form of time series and article content as input using bidirectional LSTM, thus combining the solution approaches from [Liu and Wu(2018)] and

[Khan et al.(2019)Khan, Khondaker, Islam, Iqbal, and Afroz].

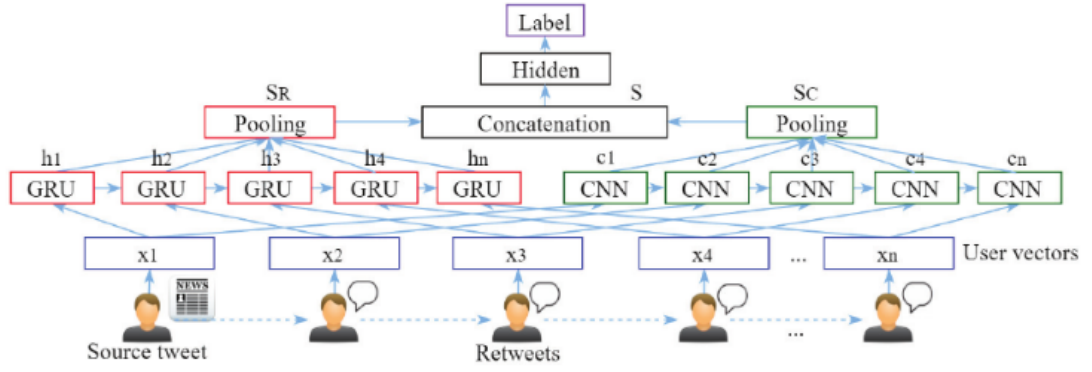
[Liu and Wu(2018)] first constructs propagation paths for each news article by identifying the users that propagated each of the articles and then constructing a time series based on user characteristics. Next, the time series is transformed into multivariate sequences of length  $n$ . If a time series' size is bigger than  $n$ , it is padded, otherwise it is randomly oversampled.

The multivariate sequences is fed into a Gated Recurrent Unit (GRU) network and converted into vectors using the Sigmoid and tangent functions. Finally, mean-pooling is applied to the outputs from GRUs to reduce the sequence of output vectors into a single vector. The goal of [Liu and Wu(2018)] was to detect fake news as early as possible, and GRUs are computationally efficient and simple because they don't need a memory unit like LSTM, which is why GRU and not LSTM was used in this model.

User characteristics are also transformed into multivariate feature vectors using one-dimensional CNN with the ReLU activation function, and reduced to one vector using mean-pooling.

The outputs from the two networks are concatenated and fed into a multi-layer feed-forward neural network that predicts class labels for their respective propagation paths using ReLU and Softmax (see figure 2.6<sup>6</sup>taken from [Liu and Wu(2018)]). Softmax is a simple, but effective classifier for linearly separable problems.





**Figure 2.6:** Combined GRU and CNN Architecture

[Khan et al.(2019)Khan, Khondaker, Islam, Iqbal, and Afroz] first preprocesses the textual input by removing stopwords and URLs from it, stemming to remove suffices, correcting spelling of words and tokenizing the input. Next features are extracted using pre-trained word embedding. Word embeddings are initialized with 100-dimensional pre-trained embeddings from GloVe. After extracting features, a bidirectional LSTM model is trained over ten epochs with a batch size of 128, an output dimension of a hundred, time steps set to three hundred, an ADAM optimizer with a learning rate of 0.0001 and the Sigmoid activation function in the final dense output layer.

## 2.5 Analysis

The problem statement in [Liu and Wu(2018)] was to detect news as early as possible. [Liu and Wu(2018)] detected fake news faster than state-of-the-art models, thus it solved the problem statement. The model detects fake news fast because it only takes time series and user characteristics as input, and not complex features such as linguistic features. However, fast training comes at the expense of low accuracy. The accuracy could be better using LSTM instead of GRU and by adding linguistic features as input, as is done in this thesis' proposed model. If one has access to a GPU, time is not of the essence.

[Khan et al.(2019)Khan, Khondaker, Islam, Iqbal, and Afroz] concluded that LSTM, spesifically C-LSTM (CNN-LSTM), showed promising results for NLP-based fake news detection with a best accuracy of 95% and that it demands further attention. Just like [Liu and Wu(2018)], [Khan et al.(2019)Khan, Khondaker, Islam, Iqbal, and Afroz] demonstrates good results, but leaves room for improvement by not utilizing more input features.

<sup>6</sup>[Liu and Wu(2018)]



## Chapter 3

# Solution Approach

This chapter presents the proposed model and explains the preprocessing steps and model layers in detail.

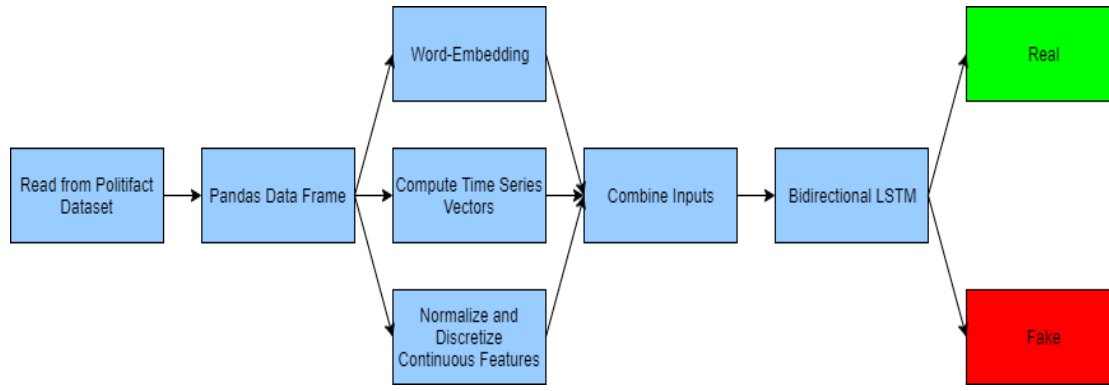
### 3.1 Proposed Model

The proposed model uses LSTM (from the Keras Sequential model), because LSTMs are efficient at remembering long sequences and modeling long-distance relations, which is preferable when using sequential textual data such as article content as input. The ReLU and Sigmoid activation functions are used (see 2.2.4). Bidirectional LSTM, which is an extension of the traditional LSTM, is used because it improves the accuracy by providing both forwards and backwards memory as explained in 2.2.6.

Article content, continuous attributes and rumor propagation have been successfully used in previous works, and this thesis combines these features (see figure 3.1). Content and metadata for tweets, retweets, and articles are first read from the Politifact dataset into a data frame (see figure 1.1). The article content is then word-embedded, time series for each rumor propagation paths are computed using the metadata, and continuous attributes from the metadata are normalized and discretized as explained in the next section. Then, these features are concatenated and used as input for Bidirectional LSTM, which, uses the input to classify articles as fake or real.

### 3.2 Preprocessing

The data undergoes multiple preprocessing steps in this thesis. First off, columns and rows with mostly NAN (Not A Number) values are dropped. Next, the remaining NAN



**Figure 3.1:** Proposed Model

values are estimated. NAN values for continuous features are estimated to the mean of the column, while Nan values for categorical features are estimated to the most common value. Then, symbols, punctuation and stopwords are removed. Finally, article content is word-embedded and continuous features are normalized as described below. Time series are also computed based on article metadata as explained in chapter 3.2.3. Before training the Bi-LSTM model and classifying, the data is split into a training set consisting of 80% of the data and a validation set consisting of the remaining 20%.

### 3.2.1 Natural Language Processing (NLP)

Article content is word-embedded (see chapter 2.2.2). The content of each article corresponds to a vector with a length of one thousand (including padding). The weights for each of the vectors are generated using 6B 100d GloVe vectors. The vectors are fed into an embedding layer (see figure 8), and output as a matrix. Embedding layers are good at capturing sentiments.

### 3.2.2 Continuous Attributes

Continuous attributes are min-max normalized, and discretized into three bins using the cut function from the Pandas library. Min-max normalization scales all the attributes to the same range (zero to one in this thesis), using the formula given below. The formula is sensitive to outliers.

$$Fi = [Xi - \min(X)] / [\max(X) - \min(X)] \quad (3.1)$$

Normalizing data ensures that all the attributes have the same influence. For neural networks, this decreases gradient descent convergence times, and thus, reduces training time. Discretizing denoises the data. Each bin smoothes out noise in sections of the data.

There are 13 continuous features:

- favorite counts for tweets
- favorite counts for users
- follower counts per user
- friend counts per user
- listed counts per user
- retweet counts per tweet
- statuses count per user
- offsets where URLs in tweet texts begin
- offsets where URLs in tweet text end
- offsets where URLs in tweet texts begin per user
- offsets where URLs in tweet text end per user
- offsets where user mentions in tweet texts begin per user
- offsets where user mentions in tweet texts end per user

### 3.2.3 Rumor Propagation

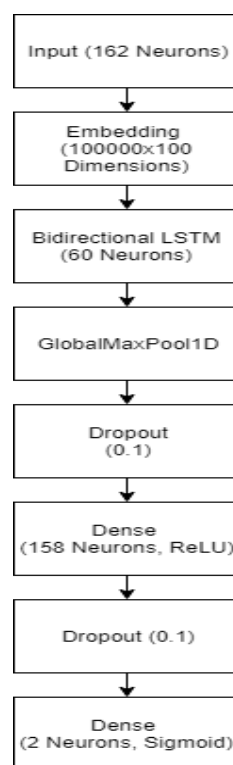
For each article, time series vectors are computed for all the corresponding rumor cascades. A rumor cascade is a chain of retweets for a given tweet of an article. The time series elements contain time passed since the first tweet for every element in a given rumor cascade, and are computed by first computing the difference in time between a given retweet and the previous retweet or tweet for all the retweets. These differences are added to a vector,  $D$ . The time series elements are then computed by finding the cumulative sums for each element in  $D$ . Studies have proven that fake articles are spread faster than real articles (see section 4.3.4), implying that a rumor propagation-based time series is viable as input for fake news detection.

### 3.3 Layers

After combining article contents, time series, and TF-IDF vectors, these combined features are passed as input for the input layer. The input layer is followed by the embedding layer (see 3.1.1). The next layer is a Bidirectional LSTM layer with 60 neurons. There is no fixed rules for setting the number of neurons in the hidden layers in the LSTM cell, but in order to prevent overfitting, the number should be less than

$$Nh = Ns / (\alpha * (Ni + No)) \quad (3.2)$$

where  $Ni$  is the number of inputs neurons,  $No$  is the number of output neurons,  $Ns$  is the number of samples in the training data set,  $\alpha$  is an arbitrary scaling factor, usually between two and ten. Next up, is a 'GlobalMaxPool1D' layer, which downsamples by computing the most present features (maximum values) for each feature map. After the Max-Pooling layer is a dropout layer with a dropout of '0.1', meaning that ten percent of the neurons or inputs are dropped. Dropout is a regularization technique that prevents overfitting by dropping a random set of hidden units or nodes at each update during training. Nodes are dropped by setting them to zero. Finally there are two dense layers with another dropout layer between them for further regularization that classifies the articles as either fake or real. The first of these dense layers has 158 neurons and applies the ReLU activation function. The last dense layer has two neurons and applies the Sigmoid activation function. The ReLU activation function is applied before Sigmoid to prevent the vanishing loss gradient (see chapter 2.2.4).



**Figure 3.2:** LSTM Graph





## Chapter 4

# Experimental Setup and Data Set

This chapter presents the technologies and the data set used in this thesis, as well as the experimental results.

### 4.1 Technologies

The code that produced the experimental results in this thesis was written in Python, because it has simple syntax rules which makes code easy to read, and it has many libraries that are useful for data science in general such as Pandas, Pickle, Numpy, and Pyplot. Pandas is good for data representation and can handle large amounts of data efficiently. Pickle can be used to save data structures as backups when running time demanding code (TQDM can offer progress bars for such cases). Numpy can be used for scalable and efficient computations. Pyplot offers a large variety of plots for visualization. Python also has good libraries for deep learning specifically, such as Keras.

#### 4.1.1 Natural Language ToolKit (NLTK)

NLTK is a package for natural language processing[NLT(2020)]. The package can be used for example to remove stopwords. Stopwords are commonly used words such as "the", "a", or "and". They do not give any context or meaning to a given sentence in the sense of NLP, and should thus be removed.

#### 4.1.2 Wordcloud

The Wordcloud package is used to visualize text, where the text size corresponds to significance or frequency[Wor(2019)]. Matplotlib is needed to visualize word clouds.

### 4.1.3 Keras

Keras has many advantages. The library is efficient on both CPU and GPU[Ker(2019)], it has preprocessing features such as tokenizing and padding, and it also has many models, such as max pooling, dropout, embedding, and bidirectional LSTM. These models can be combined, as they are in this thesis. Keras even have regularizers that can be used to avoid overfitting.

- **Keras Sequential model**

The proposed model uses the Keras Sequential model. The Keras Sequential model is easy to use and enables one to build models layer by layer, but it does not support multiple inputs.

- **Keras Functional API**

The combination of CNN and LSTM implemented for this thesis is based on the RNN+CNN model from [Liu and Wu(2018)] using articles in addition to time series and metadata as input and LSTM instead of GRU. The model was implemented using the Keras Functional API. The functional API is flexible and supports multiple inputs. This means that separate outputs from CNN and LSTM with each their inputs can be concatenated.

### 4.1.4 Regular Expressions (RE)

RE can be used to check if a given string matches a regular expression, and can be used to remove numerals from strings[RE(2020)].

### 4.1.5 Datetime

The Datetime package contains classes for manipulating dates and time by for example subtracting dates in order to compute time passed[Dat(2020)].

## 4.2 Dataset

All the data, including articles, tweets and retweets used for this thesis are retrieved from the Politifact dataset. Politifact was first started by the Tampa Bay Times in St. Petersburg, Florida in 2007[Pol(2020b)]. It is run by the Poynter Institute. Politifact is a fact-checking website that rates accuracy of claims by elected officials and others on its Truth-O-Meter. The Truth-O-Meter is a scale ranging from "True" (one hundred percent

real) to "Pants on Fire" (one hundred percent fake). The labels in this thesis however, have a binary distribution, either 'real' or 'fake'. The dataset has a total of 1056 articles, of which 432 is labelled fake and 624 is labelled real. A total of 19981 records (11528 fake and 8453 real) and 16 attributes (15 continuous including the time series and article content) was used for training. 80 % of the data set was used for training, and the rest for testing.

#### 4.2.1 Synthetic Minority Over-sampling Technique (SMOTE)

The data is imbalanced with a majority of fake articles. Classifiers tend to have worse performance on the minority class, hence, real records are oversampled to even the distribution. This is done using Keras' SMOTE. SMOTE generates new records based on existing records[Brownlee(2020b)]. A record is generated using a convex combination of the given record and one of the k nearest neighbours.

#### 4.2.2 Undersampling

Undersampling prevents skewed distributions like oversampling such as SMOTE, but does it by undersampling the majority class. This thesis uses RandomUnderSampler algorithm from the Python Imblearn API, which removes samples randomly.

#### 4.2.3 Gridsearch

Gridsearch is an SKLearn tool for tuning hyperparameters of a classifier[Gri(2019)]. Hyperparameters are constructor arguments that are not directly learnt within a given classifier. In this thesis, Gridsearch was used to estimate the number of epochs and batch size for LSTM and CNN. Gridsearch exhaustively searches over a grid of hyperparameter values for a classifier, and retains the combination of values with the best cross-validation score. The Gridsearch parameters tested for the proposed model (and CNN) is:

- **Epochs** 1, 2, 3, 4, 5
- **Batch Size** 8, 16, 32, 64, 128

## 4.3 Experimental Results

This subsection discusses and analyzes the data visualizations generated from the articles and statistics in the Politifact dataset.

### 4.3.1 Cumulative Distribution Function (CDF)

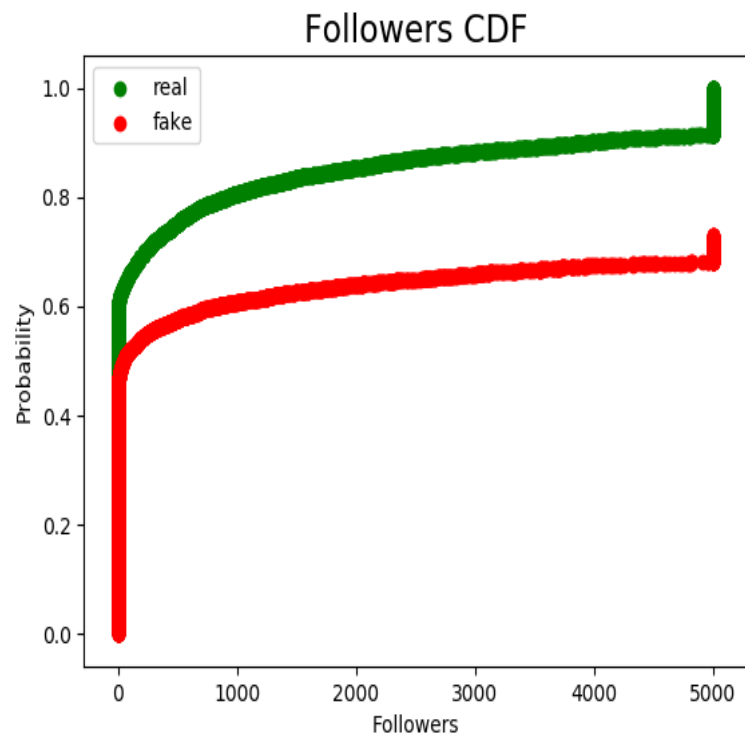
A Cumulative Distribution Function (CDF) is used to find the probability that a random variable is less than or equal to a given value[CDF()]. A probability,  $y$ , is computed by dividing the number of events that are less than or equal to  $y$  by the number of possible outcomes (all values less than or equal to  $y$ ).

### 4.3.2 Users

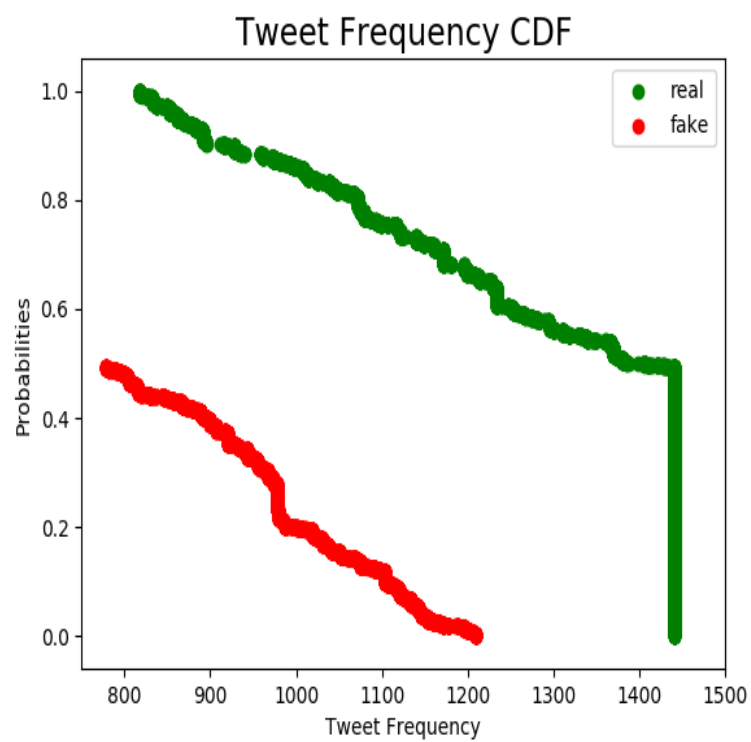
Figure 4.1 indicate that users that tweet fake articles have less followers than users that tweet real articles (users that tweet real articles have a higher probability of having many followers than users that tweet fake articles). Figure 4.2 shows that users that tweet real articles generally tweet more than users that tweet fake articles. The explanation might be that the users that tweet fake articles create new accounts as people learn that they tweet fake tweets. These results imply that metadata related to users is viable as input for classification.

### 4.3.3 TFIDF

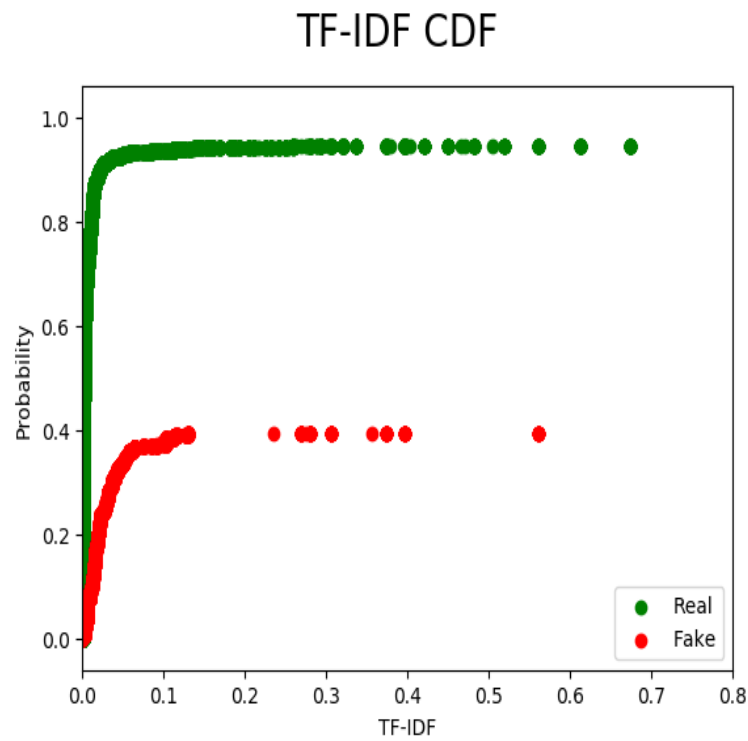
Figure 4.3 shows that terms from real articles tend to have higher TF-IDF than terms from fake articles. This means that fake articles have a smaller vocabulary than real articles and that they tend to reuse words. For example, in the Politifact dataset, a lot of fake articles revolve around presidential elections, which can be seen in figure 4.4 (Trump and Obama are among the most common words in fake articles). Figure 4.4 also demonstrates that fake articles tend to contain superlatives (i.e. supreme), and loaded words (i.e. suspended and killed), and that the content is usually more novel or shocking than in real articles, meaning that content in fake articles is usually about either celebrity gossip (i.e. Dogg as in Snoop Dogg) or serious criminal cases (i.e. killed). Figure 4.5 displays the most common terms in real articles for comparison, and it shows that the language and content is generally more mundane and generic than in fake articles.



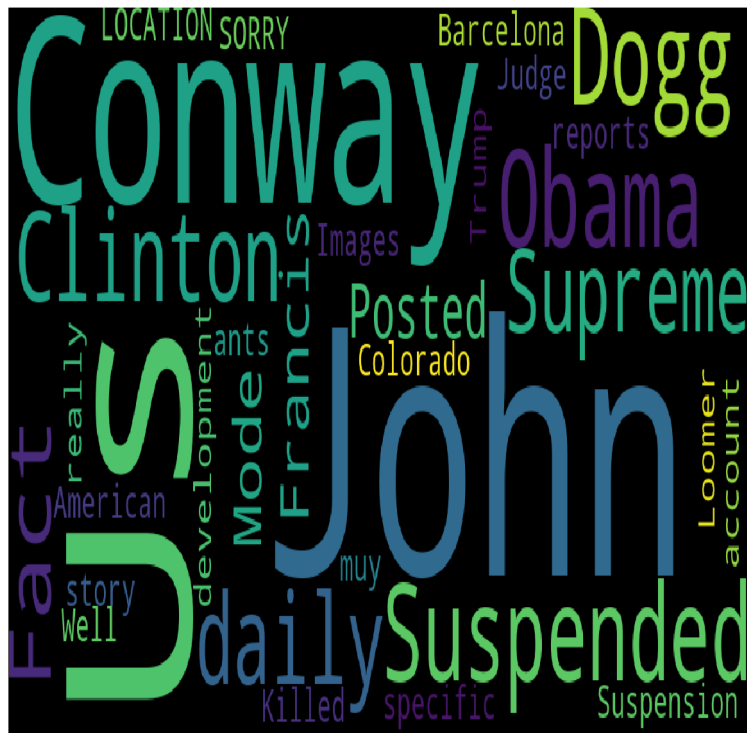
**Figure 4.1:** Followers CDF (Cumulative Distribution Function)



**Figure 4.2:** Tweet Frequency CDF



**Figure 4.3: TF-IDF CDF**



**Figure 4.4:** fake TFIDF Common Wordcloud



**Figure 4.5:** real TFIDF Common Wordcloud

#### 4.3.4 Rumour Propagation

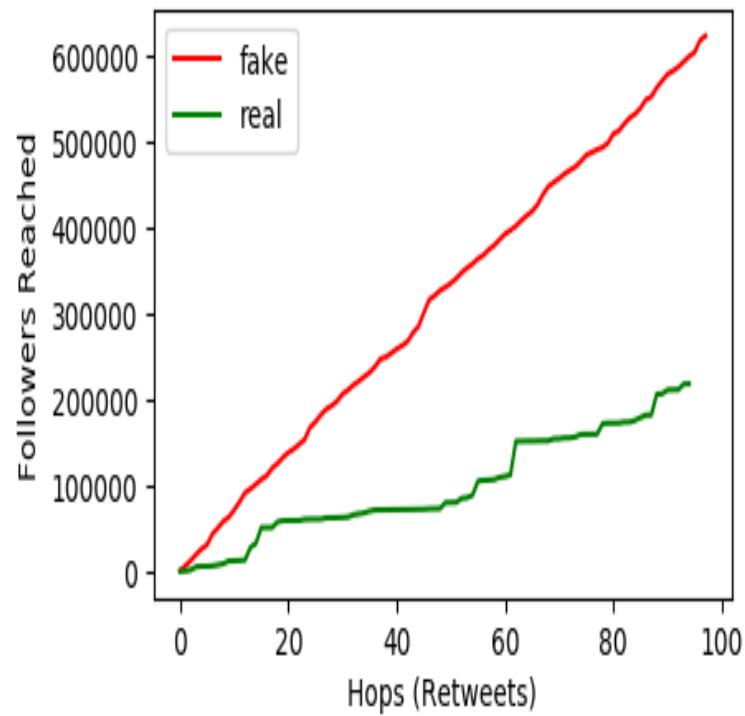
According to figure 4.6 and 4.7, tweets of fake news articles reach much more followers per retweet, and are also retweeted faster. The differences are significant, especially for the time cascade. Fake news spread a lot quicker than fake news. Fake news also start spreading much earlier than real news. Figure 4.7 also tells us that even though fake news spread faster than real news, real news actually spread quickly, when they first start to spread. This might be because followers of authors who often tweet real tweets tend to be a lot more loyal than followers of fake tweet authors, and that they receive notifications when the first tweet is tweeted, thus, they all retweet said tweet in the same time span.

### 4.4 Statistical Measures

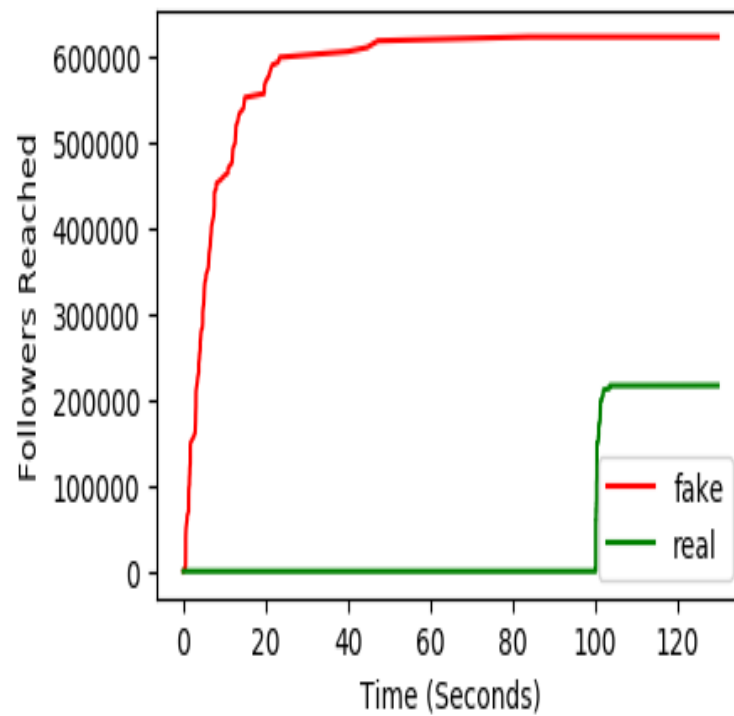
A **confusion matrix** is an accuracy measure for labelled data sets[Mishra(2018)].

Confusion matrices can be computed for multiple classes, but for a two-class problem, it will have four cells containing True Positives (TP), False Positives (FP), False Negatives (FN), and True Negatives (TN), see figure 4.8<sup>1</sup>.

<sup>1</sup><https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62>



**Figure 4.6:** Followers Reached per Hop



**Figure 4.7:** Time Cascade



		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figure 4.8: Confusion Matrix

**TP** is the number of correctly predicted positive (real) records (articles). **FP** is the number of wrongly predicted positive records. **FN** is the number of wrongly predicted negative (fake) records. **TN** is the number of correctly predicted negative records.

**Precision** is the number of correctly predicted positive records divided by all the records predicted as positive:

$$Precision = \frac{TP}{TP + FP} \quad (4.1)$$

**Recall** is the number of correctly predicted positive records divided by all the records that are actually positive:

$$Recall = \frac{TP}{TP + FN} \quad (4.2)$$

**Micro accuracy** is computed as

$$Micro = \frac{TP + TN}{TP + TN + FP + FN} \quad (4.3)$$

and is the percentage of correctly predicted records for both classes.

**Macro accuracy**[\[Raschka\(2020\)\]](#) is the mean of the recalls:

$$Macro = \frac{\frac{TP}{TP+FN} + \frac{TN}{TN+FP}}{2} \quad (4.4)$$

**F1-Score** is a useful accuracy measure if you want a balance between precision and recall, and the class distribution is uneven. The formula for F1-Score is given below:

$$F1 = \frac{2 * (Precision * Recall)}{Precision + Recall} \quad (4.5)$$

**Area Under Curve (AUC)** accuracy measures how well a model distinguishes between two classes. The 'Curve' in AUC is an ROC curve, which is a probability curve plotted with TP on the y-axis and FP on the x-axis. It is a good measure for binary classification problems with a skewed distribution. The formula for AUC is the same as for Micro accuracy, but AUC takes prediction probabilities as input instead of predicted labels. The formula for AUC is

$$AUC = \int_0^1 TPR(FPR^{-1}(x))dx \quad (4.6)$$

where

$$TPR(T) = \int_T^\infty f_1(x)dx$$

and

$$FPR(T) = \int_T^\infty f_0(x)dx$$

$f_0$  is the probability density function for the negative class and  $f_1$  is the probability density function for the positive class.

**Student's T-Test** determines if the means of two populations are significantly different or not by using hypothesis testing [Kenton(2020)]. The null hypothesis is that the means are the same. If the t-value is higher than a threshold depending on the alpha parameter, the null hypothesis is rejected. There are multiple types of t-tests, e.g. one-sample, two-sample (independent), and paired. In this thesis, a two-sample t-test is used, because the groups come from different populations (prediction probability distributions). The t-value is given by the following formula for paired t-tests:

$$T = \frac{\mu_1 - \mu_2}{\frac{s(d)}{\sqrt{n}}} \quad (4.7)$$

, where  $\mu_1$  and  $\mu_2$  is the means of the two populations,  $s(d)$  is the standard deviation of the differences between the values of the two populations, and  $n$  is the sample size.

## 4.5 Experimental Setup

The proposed models takes time series, article content and various metadata as input and classifies news articles as either real or fake using Bidirectional LSTM. The results was compared to models based on CNN, a combination of CNN and LSTM, SVM, NB, the RNN+CNN model from [Liu and Wu(2018)], the Bi-LSTM from [Khan et al.(2019)Khan, Khondaker, Islam, Iqbal, and Afroz], and the C-LSTM model from [Khan et al.(2019)Khan, Khondaker, Islam, Iqbal, and Afroz]. The results are listed in the chapter 5.1.

### 4.5.1 Support Vector Machine (SVM)

SVM classifies data by drawing a separator as a hyperplane[[Stecanella\(2017\)](#)]. If the data is not linearly separable, the data points are mapped to a higher dimension, making the problem linearly separable. SVM is effective when there are many features, but is best used on small data sets due to training time. The SVM model was trained using 10000 iterations and generated using SciKit-Learn (SK-Learn) with the 'c' parameter (classification margin) set to one, a linear kernel, 'degree' set to three, and 'gamma' set to auto. The 'degree' parameter is the degree of the polynomial kernel function and is ignored if a polynomial kernel is not used. Gamma is the kernel coefficient for the non-linear kernels. When set to auto, the kernel coefficient will be one divided by n, where n is the number of features. The SVM classifier developed for this thesis used a linear kernel. The most common kernels for SVM are listed and described below.

- **Linear** The linear kernel is a simple kernel that is used when data is linearly separable, meaning that the data can be separated by a single decision boundary[[Bajaj\(2018\)](#)]. The kernel is mostly used when the data set contains a lot of features.
- **Polynomial** A polynomial kernel is used for nonlinear data, because it considers similarity of records (as well as similarity of features).
- **Radial Basis Function (RBF)** An RBF kernel is a non-linear kernel. RBF kernels are used for curved decision boundaries. They are generally more accurate and slower at training than linear and polynomial kernels.

### 4.5.2 Naive Bayes (NB)

Naive Bayes classifiers are simple and fast probabilistic classifiers that classify labels based on probabilities using frequencies[[Nazrul\(2018\)](#)]. For NLP, non-naive Bayes classifiers compute zero probability for sentences not in the training set. Naive Bayes classifiers, however, treat terms independent of each other and compute per-term-probabilities. NB is best applied on TF-IDF vectors, because TF-IDF reflects term importance per document. Smoothing is also important when using NB, to avoid zero values. Multinomial Naive Bayes is used because term frequencies are discrete values, meaning that the distribution is multinomial.

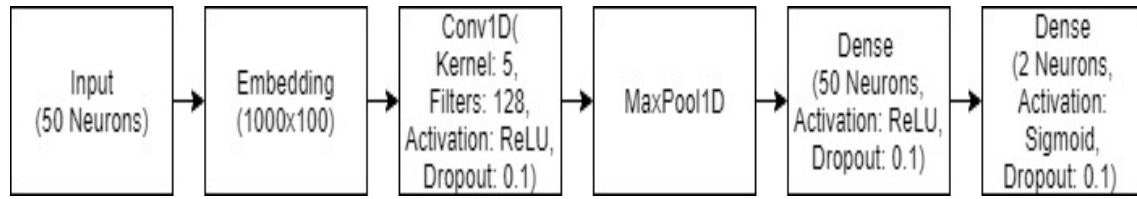


Figure 4.9: CNN Graph

### 4.5.3 Convolutional Neural Networks (CNN)

CNN is a type of neural networks that convolutes a kernel filter to estimate data points. For each convolution, the filter is used to estimate the value of the center point. CNN is generally best used to estimate pixels for computer vision, i.e. image classification. However, it can also be used for NLP. CNNs are fast to train and can have decent accuracy, even though they don't have memory like LSTM. CNNs can be used for NLP by sliding the sliding window or kernel filter over patches of the embedding matrix or a matrix of one-hot vectors [Newatia(2019)]. In the CNN-based model used for comparison in this thesis, the kernel filter is slid over the embedding matrix consisting of GloVe vectors, which is of the dimensions, 1000x100. On a general basis, LSTM outperforms CNN on NLP problems, but CNNs can perform better than LSTM on feature detection problems with short texts. LSTM performs best on problems that involve long sequences and require some sort of memory such as translations.

### 4.5.4 Long Short-Term Memory (LSTM)

See chapter 2.2.5 for information on LSTM. The layers are depicted in figure 3.2.

### 4.5.5 Optimization Techniques

Optimization techniques are used to update the weights in a neural network, in order to minimize the loss function. The optimizers described below were tested, and the one that produced the best performance results (AdaM) is used in the proposed model.

- **Root Mean Square Propagation (RMSProp)**

Root Mean Square Propagation (RMSProp) [Kathuria(2018)] is an optimization algorithm for neural network that uses different learning rates for each parameter. The learning rates are adjusted using weight gradient averages, hence, the technique performs well on noisy data.

- **Adaptive Gradient (AdaGrad)**

Adaptive Gradient (AdaGrad) also uses per-parameter learning rates. AdaGrad learning rates are low for frequent data and high for in-frequent data[Ada()], and thus improve performance on problems with sparse data (i.e. NLP problems).

- **Adaptive Moment (AdaM)**

The best result was obtained using Bidirectional LSTM with the Adaptive Moment (AdaM) optimizer which yielded a best prediction accuracy of 0.941 using oversampling for detecting fake news from the Politifact dataset. Like RMSProp, Adam also uses weight gradient averages, and are thus effective on noisy data. However, Adam also uses variance, thus, combining the perks of RMSProp and ADAgrad. Adam is also easy to implement and computationally efficient.



## Chapter 5

# Discussion

This chapter compares the results of the proposed model with baseline solutions and discusses the implications of the experimental results.

### 5.1 Baseline Comparison

The results for the proposed model are listed without any sampling, with SMOTE oversampling, and with undersampling using RandomSampler from Imblearn in the tables below. The data is sampled because there are more real records than fake records in the data set.

	Accuracy		Fake Tweets			Real Tweets			
Method	Micro	Macro	Precision	Recall	F1	Precision	Recall	F1	AUC
BiLSTM	0.943	0.987	0.989	0.911	0.948	0.894	0.986	0.938	0.989
C-LSTM	0.937	0.985	0.986	0.903	0.943	0.882	0.983	0.930	0.987
CNN	0.938	0.990	0.991	0.903	0.945	0.878	0.989	0.930	0.989
CNN + LSTM	0.934	1.000	1.000	0.884	0.938	0.869	1.000	0.930	0.990
SVM	0.624	0.619	0.488	0.407	0.444	0.684	0.750	0.716	0.500
NB	0.703	0.677	0.604	0.628	0.616	0.767	0.749	0.758	0.688
RNN+CNN (Liu)	0.585	0.293	0.585	1.000	0.738	0.000	0.000	0.000	0.576
BiLSTM (Khan)	0.939	0.983	0.985	0.906	0.944	0.889	0.982	0.933	0.988
C-LSTM (Khan)	0.938	0.982	0.984	0.907	0.944	0.885	0.980	0.930	0.987

**Table 5.1:** Performance Results (no Sampling)

The performance results of the proposed model are compared to CNN, a combination of CNN and LSTM, SVM, Multinomial NB, and the classifiers with the best results from the related works, [Khan et al.(2019)Khan, Khondaker, Islam, Iqbal, and Afroz] and [Liu and Wu(2018)]. The accuracy measures are explained in chapter 4.4.

	Accuracy		Fake Tweets			Real Tweets			
Method	Micro	Macro	Precision	Recall	F1	Precision	Recall	F1	AUC
BiLSTM	0.941	1.000	1.000	0.899	0.947	0.877	1.000	0.934	0.988
C-LSTM	0.936	0.987	0.988	0.900	0.942	0.879	0.985	0.929	0.986
CNN	0.937	1.000	1.000	0.892	0.943	0.869	1.000	0.930	0.987
CNN + LSTM	0.936	1.000	1.000	0.892	0.943	0.866	1.000	0.928	0.991
SVM	0.587	0.803	0.719	0.286	0.409	0.554	0.888	0.683	0.592
NB	0.664	0.725	0.699	0.576	0.631	0.639	0.752	0.691	0.664
RNN + CNN (Liu)	0.580	0.290	0.580	1.000	0.734	0.000	0.000	0.000	0.501
BiLSTM (Khan)	0.932	0.989	0.990	0.893	0.939	0.867	0.987	0.923	0.988
C-LSTM (Khan)	0.934	0.999	1.000	0.887	0.940	0.864	0.999	0.927	0.987

**Table 5.2:** Performance Results (Oversampling)

The confusion matrices corresponding to the classifiers with the configurations that demonstrated the best results are listed in table 5.4-12.



	Accuracy		Fake Tweets			Real Tweets			
Method	Micro	Macro	Precision	Recall	F1	Precision	Recall	F1	AUC
BiLSTM	0.939	0.981	0.984	0.909	0.945	0.887	0.979	0.931	0.988
C-LSTM	0.920	0.897	0.914	0.950	0.932	0.929	0.879	0.904	0.988
CNN	0.935	0.964	0.969	0.919	0.944	0.894	0.959	0.925	0.987
CNN + LSTM	0.939	1.000	1.000	0.893	0.943	0.876	1.000	0.934	0.991
SVM	0.926	0.971	0.969	0.880	0.922	0.890	0.972	0.929	0.929
NB	0.700	0.733	0.720	0.653	0.685	0.683	0.746	0.713	0.700
RNN + CNN (Liu)	0.590	0.295	0.590	1.000	0.742	0.000	0.000	0.000	0.4996
BiLSTM (Khan)	0.703	0.677	0.604	0.628	0.616	0.767	0.749	0.758	0.688
C-LSTM (Khan)	0.935	0.984	0.986	0.901	0.941	0.879	0.982	0.928	0.986

**Table 5.3:** Performance Results (Undersampling)

		Actual	
		Positive	Negative
Predicted	Positive	2073	202
	Negative	24	1697

**Table 5.4:** BiLSTM Confusion Matrix

		Actual	
		Positive	Negative
Predicted	Positive	2079	223
	Negative	29	1665

**Table 5.5:** C-LSTM Confusion Matrix

		Actual	
		Positive	Negative
Predicted	Positive	2113	227
	Negative	19	1637

**Table 5.6:** CNN Confusion Matrix

		Actual	
Predicted	Positive	Positive	Negative
	Negative	2027	244
		0	1725

**Table 5.7:** CNN + LSTM Confusion Matrix

		Actual	
Predicted	Positive	Positive	Negative
	Negative	914	125
		29	1010

**Table 5.8:** SVM Confusion Matrix

		Actual	
Predicted	Positive	Positive	Negative
	Negative	648	384
		424	1264

**Table 5.9:** Multinomial NB Confusion Matrix

		Actual	
Predicted	Positive	Positive	Negative
	Negative	2356	0
		1640	0

**Table 5.10:** RNN + CNN (Liu) Confusion Matrix

		Actual	
Predicted	Positive	Positive	Negative
	Negative	2048	213
		32	1703

**Table 5.11:** BiLSTM (Khan) Confusion Matrix

		Actual	
		Positive	Negative
Predicted	Positive	2088	215
	Negative	34	1659

**Table 5.12:** C-LSTM (Khan) Confusion Matrix

## 5.2 Implications of Findings

The proposed model demonstrated the best results. LSTM networks detects patterns over time, and is best for problems with long texts. CNN also showed promising results and as mentioned in 4.5.3, CNNs can perform better than LSTM on feature detection problems with short texts. This means that whether LSTM is the best alternative or not, depends on the dataset used.

The number of retweets per tweet in the dataset is insufficient, and thus also the number of rumor propagation paths. The low prediction accuracy of RNN+CNN from [Liu and Wu(2018)] confirms this, because it only takes time series and metadata as input. A higher number of retweets will most likely result in better prediction accuracies for all the classifiers using RNN, because the time series are sequential.

Undersampling leads to loss of information because records from the majority class are deleted to even the distribution. Oversampling can lead to overfitting, because new records are added to the minority class to even the distribution, and these new records might not be good representatives of the true minority class. On the one hand, oversampling produced the highest prediction accuracy (94.1%) compared to undersampling, most likely because of the information loss in regards to undersampling. This problem can be solved by using more data. On the other hand, undersampling showed better results in general. This is probably because SMOTE synthesizes new records old records, which can lead to overfitting. Overfitting due to oversampling can be avoided by implementing cross-validation.

The inputs, besides the metadata, are sequential, and as expected, Bidirectional LSTM demonstrated the best results with an accuracy of 94.3 %. CNN also performed well. CNN performing well supports the hypothesis that fake articles contain more superlatives than real articles, because CNNs work well for feature detection (detecting terms in this case). CNN is faster at training than LSTM, and since fake news spread much faster than real news, CNN might be a better choice for fake news detection.



## Chapter 6

# Conclusion and Future Directions

### 6.1 Future Directions

Most of the tweets are not retweeted at all, meaning that there is not much data provided for the rumor propagation time series and some of the metadata.

More data can be gathered from other social media platforms. This should improve the prediction accuracy especially when using undersampling. The data set also contains many other features that were not used in this work, which can be further explored. Alternative data sets can also be explored.

User characteristics can be used to identify naive users who easily fall for fake news and share them.

The data can be denoised further (i.e. remove irrelevant text from article content).

Emoticons can be removed from tweets and retweets. Tweets and retweet content can be used as input together with the content of their respective articles.

Many of the records contained video URLs instead of articles, and had to be discarded. These videos can be used for video classification using CNN. Images such as user profile pictures can also be added to the data set for image classification using CNN. Then, a combination of LSTM and CNN, using LSTM on sequential data, and CNN on image data or metadata.

Cross-validation can be implemented to avoid overfitting due to oversampling.

The proposed solution can be further developed by creating a web application with a GUI, that takes article content and/or a number of features as input, and detects if the given article is fake or not using the proposed model.

## **6.2 Conclusion**

This thesis successfully utilizes article content, metadata and rumor propagation paths to improve fake news detection by modeling a graph neural network. The proposed model, Bidirectional LSTM, scores better than state-of-the-art solutions from related works and the classifiers implemented in this thesis with an accuracy of 94.3%. The performance results are tested using the students t-test with an alpha of 0.01 and prediction probability distributions for each of the models as input. The results of the t-test agreed with the conclusion that the proposed model is the most accurate classifier, but also that it is not significantly better than the CNN+LSTM model when using undersampling.

# List of Figures

1.1	Sample Data Set	4
2.1	Neural Networks Layers	7
2.2	Recurrent Neural Network with Loops	10
2.3	GRU Cell	11
2.4	RNN Cell and LSTM Gates	11
2.5	LSTM Cell Equations	12
2.6	Combined GRU and CNN Architecture	17
3.1	Proposed Model	20
3.2	LSTM Graph	23
4.1	Followers CDF (Cumulative Distribution Function)	29
4.2	Tweet Frequency CDF	29
4.3	TF-IDF CDF	30
4.4	fake TFIDF Common Wordcloud	30
4.5	real TFIDF Common Wordcloud	31
4.6	Followers Reached per Hop	32
4.7	Time Cascade	32
4.8	Confusion Matrix	33
4.9	CNN Graph	36





# List of Tables

5.1	Performance Results (no Sampling)	39
5.2	Performance Results (Oversampling)	40
5.3	Performance Results (Undersampling)	41
5.4	BiLSTM Confusion Matrix	41
5.5	C-LSTM Confusion Matrix	41
5.6	CNN Confusion Matrix	41
5.7	CNN + LSTM Confusion Matrix	42
5.8	SVM Confusion Matrix	42
5.9	Multinomial NB Confusion Matrix	42
5.10	RNN + CNN (Liu) Confusion Matrix	42
5.11	BiLSTM (Khan) Confusion Matrix	42
5.12	C-LSTM (Khan) Confusion Matrix	43



# Bibliography

- [Cam(2020)] Fake news, 2020. URL <https://dictionary.cambridge.org/dictionary/english/fake-news>.
- [Reid(2020)] Leia Reid. Fake news week 2020: Questionable covid-19 cures and causes are being shared tens of thousands of times online, 2020. URL <https://www.brandwatch.com/blog/react-fake-news-week-covid-19/>.
- [Pol(2020a)] The infodemic: Did china deliver a covid-19 vaccine to africa?, 2020a. URL <https://www.voanews.com/covid-19-pandemic/infodemic-did-china-deliver-covid-19-vaccine-africa>.
- [Chinchilla(2019)] Laura Chinchilla. Post-truth politics afflicts the global south, too, 2019. URL <https://www.kofiannanfoundation.org/supporting-democracy-and-elections-with-integrity/annan-commission/post-truth-politics-afflicts-the-global-south-too/>.
- [Avelar(2019)] Daniel Avelar. Whatsapp fake news during brazil election ‘favoured bolsonaro’, 2019. URL <https://www.theguardian.com/world/2019/oct/30/whatsapp-fake-news-brazil-election-favoured-jair-bolsonaro-analysis-suggests>.
- [gov(2018)] Against information manipulation, 2018. URL <https://www.gouvernement.fr/en/against-information-manipulation>.
- [Danzig(2017)] Jon Danzig. How fake news caused brexit, 2017. URL <https://eu-rope.ideasoneurope.eu/2017/11/14/fake-news-caused-brexit/>.
- [Mackintosh(2018)] Eliza Mackintosh. Finland is winning the war on fake news. what it’s learned may be crucial to western democracy, 2018. URL <https://edition.cnn.com/interactive/2019/05/europe/finland-fake-news-intl/>.
- [Buchanan(2018)] Kelly Buchanan. Malaysia: Anti-fake news act comes into force, 2018. URL <https://www.loc.gov/law/foreign-news/article/malaysia-anti-fake-news-act-comes-into-force/>.

- [Newton(2019)] Casey Newton. Singapore’s fake news law should be a warning to american lawmakers, 2019. URL <https://www.theverge.com/interface/2019/12/3/20991422/singapore-fake-news-law-censorship-politics-usa>.
- [Nordheim(2018)] Gerret Von Nordheim. journalists quote social media content ever more frequently, 2018. URL <https://en.ejo.ch/research/journalists-quote-social-media-content-ever-more-frequently>.
- [Chi(2019)] How fake news and rumors are stoking division in hong kong, 2019. URL <https://www.bloomberg.com/news/articles/2019-11-11/how-fake-news-is-stoking-violence-and-anger-in-hong-kong>.
- [Cheo(2018)] James Cheo. Fake news can make - or break - stock prices, 2018. URL <https://www.businesstimes.com.sg/opinion/fake-news-can-make-or-break-stock-prices>.
- [Jardine(2018)] Eric Jardine. Beware fake news, 2018. URL [https://www.cigionline.org/articles/beware-fake-news?gclid=EAIaIQobChMl\\_X4nZn-5wIVkRsYCh11MQueEAAYASAAEgI7LPD\\_BwE](https://www.cigionline.org/articles/beware-fake-news?gclid=EAIaIQobChMl_X4nZn-5wIVkRsYCh11MQueEAAYASAAEgI7LPD_BwE).
- [Abrams(2019)] Abigail Abrams. Here’s what we know so far about russia’s 2016 meddling, 2019. URL <https://time.com/5565991/russia-influence-2016-election/>.
- [Polyakov(2018)] Alexander Polyakov. Detecting fake content: One of the biggest challenges for 2020, 2018. URL <https://www.forbes.com/sites/forbestechcouncil/2020/01/02/detecting-fake-content-one-of-the-biggest-challenges-for-2020/#8c82b4e1219d>.
- [Khan et al.(2019)Khan, Khondaker, Islam, Iqbal, and Afroz] Junaed Younus Khan, Md Khondaker, Tawkat Islam, Anindya Iqbal, and Sadia Afroz. A benchmark study on machine learning methods for fake news detection. *arXiv preprint arXiv:1905.04749*, 2019.
- [Liu and Wu(2018)] Yang Liu and Yi-Fang Brook Wu. Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [Panchal(2018)] Shubham Panchal. Artificial neural networks — mapping the human brain, 2018. URL <https://medium.com/predict/artificial-neural-networks-mapping-the-human-brain-2e0bd4a93160>.

- [Marr(2018)] Bernard Marr. What is deep learning ai? a simple guide with 8 practical examples, 2018. URL <https://www.forbes.com/sites/bernardmarr/2018/10/01/what-is-deep-learning-ai-a-simple-guide-with-8-practical-examples/#23fe9e648d4b>.
- [Hansen(2019)] Casper Hansen. Neural networks: Feedforward and backpropagation explained optimization, 2019. URL <https://mlfromscratch.com/neural-networks-explained/#/>.
- [Mazur(2015)] Matt Mazur. A step by step backpropagation example, 2015. URL <https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/>.
- [S.(2018)] Suryansh S. Gradient descent: All you need to know, 2018. URL <https://hackernoon.com/gradient-descent-aynk-7cbe95a778da>.
- [Agrawal(2019)] Samarth Agrawal. What the heck is word embedding, 2019. URL <https://towardsdatascience.com/what-the-heck-is-word-embedding-b30f67f01c81>.
- [Pennington(2014)] Jeffrey Pennington. Glove: Global vectors for word representation, 2014. URL <https://nlp.stanford.edu/projects/glove/>.
- [Brownlee(2017a)] Jason Brownlee. A gentle introduction to backpropagation through time, 2017a. URL <https://machinelearningmastery.com/gentle-introduction-backpropagation-time/>.
- [Brownlee(2017b)] Jason Brownlee. Understanding gru networks, 2017b. URL <https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be>.
- [Donahue et al.(2014)] Donahue, Hendricks, Guadarrama, Rohrbach, and Venugopalan] Jeff Donahue, Lisa Hendricks, Sergio Guadarrama, Marcus Rohrbach, and Venugopalan. Long-term recurrent convolutional networks for visual recognition and description. *Arxiv*, PP, 11 2014. doi: 10.1109/TPAMI.2016.2599174.
- [Phi(2018)] Michael Phi. Illustrated guide to lstm’s and gru’s: A step by step explanation, 2018. URL <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>.
- [Wang(2019)] Chi-Feng Wang. Illustrated guide to lstm’s and gru’s: A step by step explanation, 2019. URL <https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484>.

- [Brownlee(2020a)] Jason Brownlee. How to develop a bidirectional lstm for sequence classification in python with keras, 2020a. URL <https://machinelearningmastery.com/develop-bidirectional-lstm-sequence-classification-python-keras/>.
- [A.Lakshmanarao(2019)] T. Srinivasa Ravi Kiran A.Lakshmanarao, Y.Swathi. An effecient fake news detection system using machine learning. 2019. ISSN 2278-3075.
- [Ibrain and Lloret(2019)] Álvaro Ibrain and Lara Lloret. Fake news detection using deep learning. *arXiv preprint arXiv:1910.03496*, 2019.
- [NLT(2020)] Nltk (natural language toolkit) tutorial in python, 2020. URL <https://www.guru99.com/nltk-tutorial.html>.
- [Wor(2019)] Generating wordclouds in python, 2019. URL <https://www.datacamp.com/community/tutorials/wordcloud-python>.
- [Ker(2019)] Keras, 2019. URL <https://keras.io/>.
- [RE(2020)] re - regular expression operations, 2020. URL <https://docs.python.org/3/library/re.html>.
- [Dat(2020)] datetime — basic date and time types, 2020. URL <https://docs.python.org/3/library/datetime.html>.
- [Pol(2020b)] Politifact, 2020b. URL <https://www.allsides.com/news-source/politifact>.
- [Brownlee(2020b)] Jason Brownlee. Smote for imbalanced classification with python, 2020b. URL <https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/>.
- [Gri(2019)] Gridsearchcv, 2019. URL [https://scikit-learn.org/stable/modules/generated/sklearn.model\\_selection.GridSearchCV.html](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html).
- [CDF()] Cumulative distribution function, probability density function. URL <https://confluence.ecmwf.int/display/FUG/Cumulative+Distribution+Function%2C+Probability+Density+Function>.
- [Mishra(2018)] Aditya Mishra. Metrics to evaluate your machine learning algorithm, 2018. URL <https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234>.
- [Raschka(2020)] Sebastian Raschka. Machine learning faq, 2020. URL <https://sebastianraschka.com/faq/docs/multiclass-metric.html>.

- [Kenton(2020)] Will Kenton. T-test, 2020. URL <https://www.investopedia.com/terms/t/t-test.asp>.
- [Stecanella(2017)] Bruno Stecanella. An introduction to support vector machines (svm), 2017. URL <https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/>.
- [Bajaj(2018)] Prateek Bajaj. Creating linear kernel svm in python, 2018. URL <https://www.geeksforgeeks.org/creating-linear-kernel-svm-in-python/>.
- [Nazrul(2018)] Syet Sadat Nazrul. Multinomial naive bayes classifier for text analysis (python), 2018. URL <https://towardsdatascience.com/multinomial-naive-bayes-classifier-for-text-analysis-python-8dd6825ece67>.
- [Newatia(2019)] Rajat Newatia. How to implement cnn for nlp tasks like sentence classification, 2019. URL <https://medium.com/saarthi-ai/sentence-classification-using-convolutional-neural-networks-ddad72c7048c>.
- [Kathuria(2018)] Ayoosh Kathuria. Intro to optimization in deep learning: Momentum, rmsprop and adam, 2018. URL <https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/>.
- [Ada()] Adagrad. URL <https://databricks.com/glossary/adagrad>.