UNIVERSITY OF GHANA
COLLEGE OF BASIC & APPLIED SCIENCE

A FRAMEWORK TO DETERMINE SARCASTIC SENTIMENTS IN OPINION POLLS

BY

FREDRICK BOAFO

(10495750)

THIS THESIS IS SUBMITTED TO THE UNIVERSITY OF GHANA, LEGON IN

PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF

MPHIL COMPUTER SCIENCE DEGREE

DEPARTMENT OF COMPUTER SCIENCE

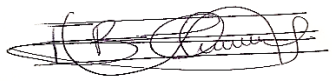
JULY, 2020

DECLARATION

I hereby declare that this thesis is my original research work carried out at the Department of Computer Science, University of Ghana, Legon under the supervision of my thesis supervisors. Additions from other people have been duly cited and acknowledged.

STUDENT

Name: Fredrick Boafo

Signature: 

Date 23rd July 2020

SUPERVISOR

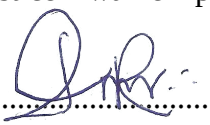
Name: Dr. Solomon Mensah

Signature: .....

Date: .....03/02/2021.....

CO-SUPERVISOR

Name: Dr. Justice Kwame Appati

Signature: 

Date: .....3rd February, 2021.....

DEDICATION

To my family and loved ones.

ACKNOWLEDGMENT

I express my profound thanks to my LORD, God Almighty, for granting me His wisdom, knowledge, and understanding and guiding and protecting me throughout the writing of this thesis successfully.

I wish to tender my profound gratitude to my supervisors Dr. Solomon Mensah and Dr. Justice Kwame Appati and all other lecturers for their support and guidance throughout my studies and thesis work. May the Almighty Lord bless and replenish all that they have lost in the course of executing their duties.

My next appreciation goes to the entire staff of the Department of Computer Science for their vital help and support in making my thesis a success.

Finally, to my family members, Mr. Alfred Koomson, Margaret Cudjoe, Rev. and Mrs. Anthony Owusu Sekyere Kwarteng, Rev. and Mrs. Gideon Boadu-yirenkyi, Christian Bossu, Priscilla Adjei-Amoako, Serwaa Owusu-Donkor, Simon Sackey, Samuel Abedu, Abigail Wiafi, Rev Father William Abeiku Apprey, Jacqueline Kumi, Melody Kakraba, Akon Ekpezu, Dr. Ferdinard Katsriku, Dr .Abdulai Jamal, Dr. Winfred Yaokumah, Dr. Isaac Wiafi, Mr. Chris Armefio and Auntie Cynthia, Uncle Dan, Paul Yidana and all my colleagues in the Department who helped in assorted ways to make this work a success, I say God richly bless you and take you to higher heights in your academic and life endeavors.

ABSTRACT

An examination of the expression of opinions and attitudes by social media and internet users toward a specific topic is what sentiment analysis is about. In diverse fields such as commerce, politics and education, it is predominantly utilized to aid decision making. Sarcasm delivers an implicit information that is opposite to what one positively declares or writes. It is often viewed as a witty language that expresses scorn, insult and reprimand in a hilarious manner. Sarcastic sentiments by people become a problem if they cannot be detected in a sentiment classification since that can affect judgment and decision making. Previous works have indicated that the detection of sarcasm during sentiment analysis can be a very tedious task and may consume a lot of time.

To classify a sentiment into sarcastic or non-sarcastic, and determine the level of predicted accuracy and loss. By so doing, one can be able to investigate the significant impact and effect of sarcastic sentiment in an opinion poll through a theoretical and empirical experiment. The study reviews different approaches for detecting sarcasm in sentiment analysis and how they perform. Lastly, the study introduces a framework that will assist in the identification and classification of sarcastic sentiments in an opinion poll by classifying sentiments as sarcastic or non-sarcastic.

A framework is introduced that comprises of three operators namely Cluster+Expert Judgement, Train & Validate and Classify & predict; that facilitates sentiment classification. Based on our empirical findings on state-of-the-act deep learning techniques such as BLSTM and sAtt-BLSTM, we employed a simple, efficient and straight forward approach by employing purely LSTM to produce the framework that helps in the determination of sarcastic sentiments. The driverless car dataset composed of the emergence of driverless cars being powered by Google is considered for

the empirical analysis. We applied the X-means clustering algorithm with expert judgment to ensure the efficient labeling of the chosen unsupervised dataset into sarcastic or non-sarcastic classes. We also employed a broad-spectrum twitter dataset consisting of both sarcastic and non-sarcastic tweets. The performance of our model has been evaluated using performance measures such as recall, precision, accuracy and F1-score emerging from a generated confusion matrix.

Our test result for the sarcastic statement using a sarcastic tweet has a prediction value of 0.998875 while that of the non-sarcastic tweet has a prediction value of 0.000055. The results provide an accurate, efficient, and reliable prediction based on the generated confusion matrix and loss figures derived. The result, therefore, indicates that though LSTM is relatively cheap as compared to BLSTM, it resulted in improved classification performance.

Based on our empirical investigations through a thorough review, a framework has been introduced to classify sentiments as sarcastic or non-sarcastic. X-means clustering algorithm with an expert judgment approach has been applied to label extracted dataset without complete labels. A deep learning technique such as LSTM has been adopted because it is efficient but cheaper as compared to most state-of-the-art deep learning techniques.

Keywords: *Sentiment analysis, Sarcasm, Classification techniques, Long Short Term Memory, Opinion poll*

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

LSTM	Long Short Term Memory
RNN	Recurrent Neural Network
SVM	Support Vector Machine
TF-IDF	Term Frequency and Inverse Document Frequency
NB	Naïve Bayes
LR	Logistic Regression
API	Application Program Interface
DNN	Deep Neural Network
CNN	Convolutional Neural Network
GRU	Gated Recurrent Unit
POS	Part of Speech
SLR	Systematic Literature Review
GPU	Graphics processing Unit
SAtt-BLSTM	Soft Attention-Based Bidirectional Long Short-Term Memory Model With Convolution Network

CHAPTER 1

INTRODUCTION

Sentiment analysis is the expression of opinions and attitudes by social media and internet users towards a specific topic or subject. It is predominantly applied in the area of commerce, politics, and education, and in the entertainment industry amongst others to help in decision making. Sarcasm, on the other hand, is when someone delivers implicit information which is opposite to what was said or written (Bouazizi & Ohtsuki, 2015). Most people view sarcasm as a witty language that conveys scorn or insult and as a language used to reprimand something or someone hilariously. The authors Parwal et al. (2018) and Bharti et al. (2017), consider sarcasm analysis as a very tedious task. The feature of sarcastic sentiments that renders it tedious in its detection is the gap that exists between literal and intended meaning. Sarcasm is employed often in the day to day speech and it is prevalent in the contexts of online (Teh, Boon, Chan, & Chuah, 2018). Sarcasm detection and scrutiny have become one of the core problems in natural language processing and detection of sarcastic sentiment in online media platforms including Twitter, Facebook, online blogs and others has become critical as they go a long way to influence decision making in organizations (Bharti et al., 2018). Most researchers ignore sarcasm when undertaking sentiment analysis because they see it as a complex task that consumes time and effort (Parmar et al., 2018; Rahayu et al., 2018; Chandankhede & Chaudhari, 2018). Some researchers also believe that sarcasm is for criticisms and mockery (Bouazizi & Otsuki, 2016; Porwal et al., 2019).

Our focus in this research is on sarcasm in sentiment analysis. Sarcasm in sentiment analysis is an expression where a person directs their opinions opposite to what they truly mean during an opinion solicitation on a particular platform (Rendalkar & Chandankhede, 2018a). In most cases, sarcastic sentiment or statement can be very funny and irritating (Bhan & D'Silva, 2018). Sarcasm

actually modifies the extremity of a positive or negative articulation to the opposite of it (Khullar & Singh, 2019). Due to the complexity of sarcasm detection, there is quite a few research undertaken on the topic. This thesis, therefore, seeks to undertake further research on sarcasm in sentiment analysis by undertaking a systematic review and an in-depth analysis of sarcasm in sentiment analysis over the past 5 years and then develop a system that helps in the identification of a sarcastic statement when undertaking sentiment analysis.

1.1. Background and Motivation

Sentiment analysis is among the trending areas in artificial intelligence (AI) whereby researchers are focusing much attention on conducting a series of studies. The opinions of individuals are very critical in planning and decision making. Decision-makers for that matter cannot make suitable decisions or lead appropriately if they cannot understand the sentiment of the general public. Social media is flourishing and sentiments of individuals are made known and exposed to these platforms. What then is Sentiment Analysis? According to Agarwal et al. (2015), Sentiment analysis basically involves analyzing people's sentiments towards a particular subject matter.

In one among the quarters of 2015, the social media network was having three-hundred and five million monthly users who were very active , and in 2018 the quarterly statistics of active monthly users upgraded to three-hundred and thirty-five million Losada & Benito(2018). To obtain real-time information for processing, social networks like Twitter, Facebook and the likes become inevitable (Paredes-valverde, Colomo-palacios, Salas-zárate, & Valencia-garcía, 2017). Information on social media reveals users' emotions and attitudes on every topic that they will actually be finding readers and listeners (Li et al., 2013). Twitter which was established in 2006 has experienced the rapid increase of users within the first years of operations. Currently, it has

over five hundred million registered users and over two hundred million active monthly users (Romanowski, 2015). In graceful of this efficacious Twitter drive, all major contenders and political parties are now having some sort of existence on social media. These growths of Twitter usage during elections by politicians, campaigner and even the public has led to ever-increasing research within the areas of social media sentiment analysis and data analytics (Park, Sung, Sharma, Jeong, & Yi, 2017). Applications of sentiment analyses are in the area of E-commerce, Politics, Business, Education, Media, etc. Many researchers have produced works that help in the classification of sentiments by classifying them as positive, negative, and neutral as seen from our related works.

Current research in the field of sentiment analyses is contributing to solving the matter of sarcasm in the analysis of sentiments. Manohar and Kulkarni (2018) proposed a natural language processing and corpus-based approach to figure out sarcasm on Twitter. Others such as Lunando and Purwarianti (2013) used machine learning algorithms in their classification by proposing the number of interjection words features in the determination of sarcasm and negative information. (Kumar et al., 2020), claim that using a multi-head Attention based long Short term Memory (LSTM) could yield better results than SVM. However they failed to perform a comparative analysis using LSTM in other to make a general claim. According to (Son et al., 2019), Bidirectional LSTM with convolutional will produce a better results, however their experiment seemed more tedious and time consuming and could do no better than multi-head attention based LSTM.

In this study therefore, we seek to use LSTM to introduce a framework to determine whether a sentiment extracted from an opinion poll is sarcastic or not. The study will make use of the Twitter dataset on a given opinion poll and apply a text mining algorithm to perform the sentiment analysis.

1.2. Problem Statement

Previous studies have shown that a sentiment or an opinion classified during sentiment classification may be a sarcastic sentiment and not the connotation of the word. But the question is, how can one be able to tell whether a sentiment is indeed positive and not just a sarcastic statement?

1.3. Aim

This study, therefore, seeks to classify sarcastic and non-sarcastic sentiments.

1.4. Objectives

The study seeks:

1. To conduct a systematic literature review of sarcasm in sentiment analysis
2. To classify a sentiment into sarcastic or non-sarcastic**.**
3. To investigate the impact of sarcastic sentiment in an opinion poll.

1.5. Scope

The focus of this study covers sarcastic sentiment analysis using datasets on opinion polls. The Twitter dataset covers issues on driverless car dataset and an extensive twitter dataset.

1.6. Research Contribution

The major contribution of this study are:

1. Theoretical and empirical prove of the efficiency of LSTM
2. Application of X-means clustering and expert judgement exclusively on sarcastic data labelling
3. Develop a framework that classifies sentiments into sarcastic or non-sarcastic.

1.7. Conclusion

This study focuses on developing a framework that will help in the identification of sarcastic sentiment in an opinion poll when undertaking sentiment analyses. This will assist to investigate the significant impact of the sarcastic statements in opinion polls. Twitter datasets have been used for this study.

1.8. Organization of Study and Research Plan

Chapter 1 presents fundamental information on the background and motivation for the study, problem statement, scope of our study, research objectives and contribution. Chapter 2 presents a summary of related works pertaining to sarcasm in sentiment analysis. The chapter shows clearly the clarity of differences between previous and current studies. Chapter 3 does an empirical analysis. A detailed description of the datasets and the preprocessing techniques are discussed. Moreover, a theoretical foundation has established a solid foundation to address the sarcasm detection problem using LSTM. Finally, the experiment setup and performance evaluation measures are discussed. Chapter 4 provides the experimental results and discussion from our empirical analysis using the defined datasets. Evaluation measures using confusion matrix including precision, recall and f1-score are used to measure the performance of our results. Finally, chapter 5 gives a summary of this thesis preceded by a concluding remark on the output of our analysis. The threats that can possibly affect the validity of our results are discussed and they include external validity claims, internal validity claims, construct validity claims and conclusion validity. Finally, future directions arising from this thesis are discussed

CHAPTER 2

SYSTEMATIC LITERATURE REVIEW

The chapter presents a set of preliminaries to understand this dissertation. The chapter also presents a review of related works and a systematic literature review concerning sarcasm in sentiment analysis.

2.1. Introduction

A lot of works have been undertaken in the area of sentiment analysis as a whole taking into consideration different approaches, classification techniques, and feature selection of which most are going to be discussed.

2.2. Related Works on Sentiment Analysis

In their study, Khasawneh et al. (2013), performed a comparison between two free online sentiment analysis tools: Social Mention and SentiStrength which support Arabic. The subsequent phases described the methodology that proceeded during this study: Build an outsized dataset of Arabic opinions with their emoticons, which are classified manually into five main categories: sports, economics, health, education, and news. Build twelve dictionaries; ten of them should be for the positive and the other should be negative for the categories. Moreover, there should be one dictionary for positive emoticons and the other for negative emotions. Run the set of Arabic sentiment analysis tools using the dataset to detect the polarity for every collected Arabic opinion within the dataset and compare these tools, and evaluate them under several considerations. Support Vector Machine (SVM) and Naïve Bayes (NB) were used as classifiers. No provision was made for sarcasm detection in their study and hence its potential to help in better judgement becomes limited. Alayba et al. (2020), presented an Arabic dataset, which talks about opinions on

our health services and has been collected from Twitter. During this experiment, a mixture of "Unigram" and "Bigram" techniques were used for text feature selection. TFIDF (Term Frequency and Inverse Document Frequency) and every feature within the corpus and the maximum one thousand weighted features were fed to their machine learning algorithm.TF-IDF has been considered in our implementation. There are three machine learning algorithms that were used are: Logistic Regression (LR), Support Vector Machines and Naïve Bayes, to seek out the foremost influential parameter in getting good results (Wp et al., 2017). The detailed process involved the retrieving of data in the sort of a tweet using Application Program Interface (API) from twitter then data is stored in CSV format and then data testing or training of data is executed by preprocessing. The method of preprocessing entails the deletion of URL, checking of punctuations, deletion of stop words, a change of the word slang to raw, and then stemming. The implemented feature extraction process on a tweet was a result of the clean results of preprocessing. The feature extraction process includes word grouping with the Bag of Words method and feature weighting with Term Frequency and Inverse Document Frequency (TF-IDF). The classification method produced a recall value, precision and accuracy of the sentiments. In his studies, Saeed (2018), proposed a Deep Neural Network architectures that classify the overlying sentiments with high accuracy. More so, the author showed that the proposed classification framework did not require any laborious text pre-processing and was capable of handling text pre-processing (E.g. stop word removal and feature engineering). The traditional text classification pipeline included text pre-processing, learning model, text encoding, and evaluation. The primary and also the baseline architecture in the study was Convolutional Neural Network (CNN) with 1D convolutions. After then four other architectures were studied, in which two were based on CNNs, one on Long Short Term Memory (LSTM) and the other on Gated Recurrent Unit (GRU). It is noted that the deep

learning methodology which requires no pre-processing due to its inherent feature engineering capabilities was actually the one which was proposed. Therefore based on their empirical results and with the consideration of overlapping text classification, they recommended there was not a need to spend excessive amount of time in data pre-processing. Stop words, punctuation, etc. proved to be an important constituent of data when it involves the training of the models. Every DNN model shows improvement in model performance measurement metrics when trained using unprocessed data. Munandar et al. (2018), investigated the general public opinion and sentiment from Indonesian twitter toward the utilization problem. The method included dataset description, multichannel-convolution neural network (CNN) model, word embedding, and Stop word and model experimentation. Since they achieved higher results in accuracy, the proposed CNN as the best as compared to SVM. The proposed methodology of Harvinder et al. (2015) utilizes only two features to detect sentiments, features like ngrams, bigrams, twitter specific features and semantic features. In his study, Adri (2016), aimed to reach more reliable public opinion measurements. They however made a claim after their experiment that Twitter data are not consistent especially when inference method is applied on them. It is however the case that most sentiment analyses research utilizes Twitter data.

Troussas et al. (2016), evaluated the basic ensemble methods that could be utilized for effective sentiment analysis. Their experiment tries to increase the efficiency of machine learning algorithms and they prove that they can perform way better that the traditional algorithms.

The work of Paredes-valverde et al. (2017), proposed a deep-learning-based approach that permits companies and organizations to detect opportunities for improving the quality of their products or services through sentiment analysis. Convolutional neural networks (CNN) and word2vec were

used. Eighty-eight point seven percent (88.7 %) precision from 100,000 tweets from experiments was piloted with different sizes of twitter and an f-measure of 88.7%.

It should however be noted that, none of the reviewed papers under this section extensively considered sarcasm detection in the analysis of sentiments.

Figure 1 depicts the different sentiment analysis approaches. However, Akhoundzade & Devin (2019), and Sanagar & Gupta (2020), proposed an unsupervised learning approach in undertaking sentiment analysis.

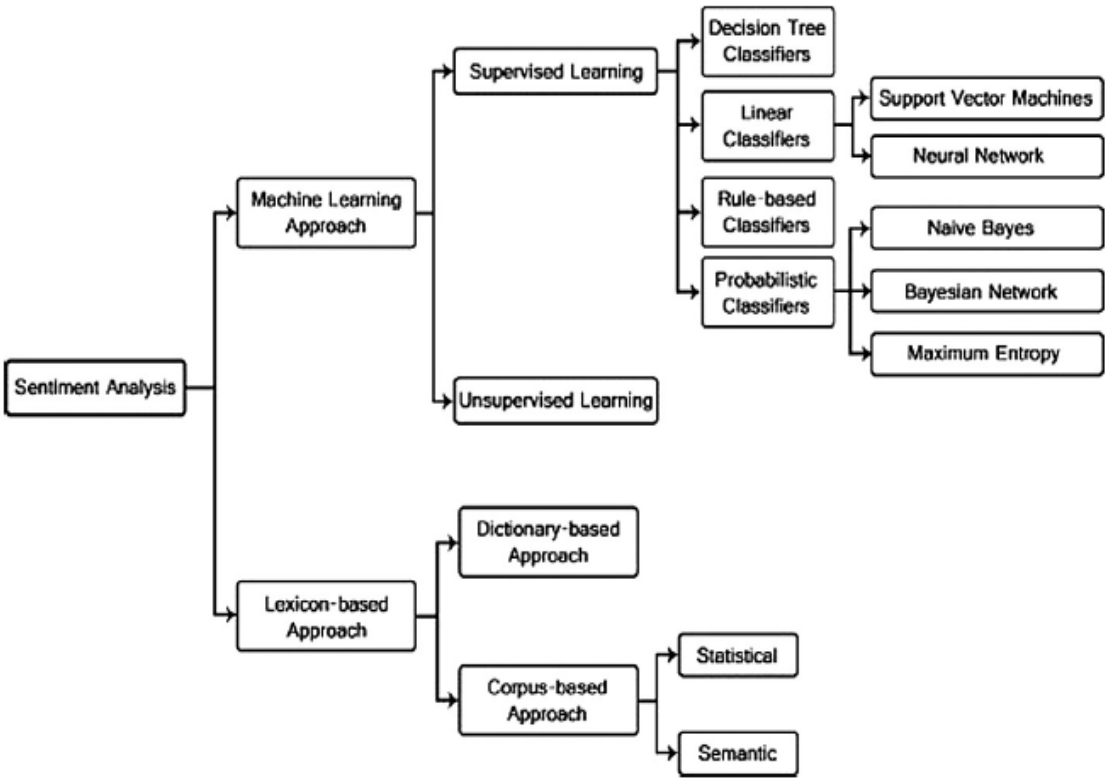


Fig 1. Sentiment Analysis Approaches (Serrano-Guerrero, Olivas, Romero, & Herrera-Viedma, 2015)

2.3. Reviews on Sarcasm in Sentiment Analysis

In Bouazizi et al. (2015), the researchers came up with a method that makes a minimal set of features but then efficiently classifies tweets regardless of the topic. The study analyzed the relevance of determining sarcastic tweets automatically, signifying the correctness of sentiment analysis which can be enhanced due to the knowledge of sarcastic and non-sarcastic sentiment. Their study however failed to make provision for unsupervised dataset. Bharti et al. (2015), use two approaches to detect sarcasm in a text. These are the parsing –based lexicon generation algorithm with the occurrence of the interjection word. The approaches were compared with the existing state-of-the-art approach to detect sarcasm. (Lunado and Purwariati, 2013), proposed two features to detect sarcasm after a sentiment analysis is done. The features are the negative information and the number of interjection words. The sentiwordnet was used in the classification of sentiment. Bhan et al. (2018), likewise proposed a system that measures sarcasm using tweets from twitter. Different algorithms were proposed to detect the effect of sarcasm on texts and generate a score. Different features are generated from the received tweets which helped to generate a score. The study also provides a separate portal to check the score of the sentence entered by the user and determine the score. Prasal et al. (2017), undertakes a comparison of various classification algorithms to detect sarcasm in tweets from the twitter streaming API. The basset classifier is chosen and paired with various pre-processing and filtering techniques using emoji and slang dictionary mapping to produce the best accuracy. Most of the reviewed papers under this section however failed to make provision for unsupervised dataset or dataset that comes unlabeled. Others still resort to traditional and machine learning algorithms.

According to Magumater et al. (2019), the authors hold the view that knowledge in sarcasm detection can be relevant to sentiment classification and vice versa. The paper shows that the two

tasks are correlated, and present a multi-task learning-based framework using a deep neural network that models the correlation in other to improve the performance of both tasks in a multitask learning setting. Razali et al. (2018), Studies the trends of sarcasm detection and their proposed techniques. It focused on the detection of sarcasm and the argument that more than text is needed to properly detect sarcasm. The authors, Dharmavarapu and Bayana (2019) employed Naïve Bayes classification and AdaBoost algorithms to detect sarcasm on Twitter. Whiles the AdaBoost algorithm was used in making weak to a strong statement by iteratively considering the subject of training data, Naïve Bayes classifies tweets into sarcastic and non-sarcastic. In the study conducted by Khullar and Singh (2019), bagged gradient boosting is proposed with particle swarm optimization as feature selection. It is compared with other classifiers such as random forest, gradient boosting, and bagged gradient boosting. After the emoji and acronym dictionary mapping is done, part of speech (POS) labeling is introduced. Hashtags and stop words are recognized and removed. Particle swarm optimization is employed in the removal of noisy data. Moreover, the authors, Losada and Benito (2018) did further research to refine sentiment tools to enhance their sensitivity and capability and also cause an optimization with sophisticated sarcasm detection. In the research undertaken by Bharti et al. (2017), the authors also proposed six algorithms to examine the sarcasm in tweets of twitter. The experiment outputs were compared with some of the existing state-of-the-art. Porwal et al. (2019) uses recurrent neural network (RNN) model for sarcasm detection since it automatically extracts features required for machine learning approaches and also uses long short term memory cells on tensor flow to capture syntactic and semantic information over twitter tweets to detect sarcasm. In Rendalkar et al. (2018b), the aim was to develop a system that groups posts based on emotions, sentiment and figure out sarcastic posts if it exists. They proposed to develop a prototype that will aid in coming to an inference about the

emotion of posts. Moreso, Parmar et.al (2018), used a Hadoop based framework that utilized live tweets, processes it, and uses a hybrid algorithm that determines sarcastic sentiment efficiently. It is noted that the hybrid approach used lexical and hyperbole feature to improve the performance of the system by increasing accuracy, precision and F-score. Raghav et al. (2018), likewise talked about the approaches, features, datasets and issues associated with sarcasm detection. Chaudhari (2018), enumerated approaches, issues, challenges and future scopes in sarcasm detection. Hiai et al. (2016), uses three stages; judgment process based on rules for 8 classes, boosting rules and rejection rules in analyzing and classifying sentences into 8 classes by focusing on evaluation expressions and then generate classification rules for every class to extract sarcastic statements.

To reiterate, most of the reviewed papers under this section tactlessly failed to make establishment for unsupervised dataset or dataset that comes unlabeled. Some current researchers still resorted to traditional and machine learning algorithms**.** A few work, has been done on determining sarcasm in sentiment analysis using different classification models, methodology, feature selection algorithms and datasets.

2.4. A Systematic review of sarcasm in sentiment analysis

A protocol was followed to undertake a systematic review of sarcasm in sentiment analysis within the preceding few years by clearly defining the inclusion and exclusion criteria used and the threat to its validity.

2.4.1. Research Methodology

The systematic literature review method provides a means to categorizing, discovering and probing the present research linked to any questions of concentration and research areas.

2.4.2. Research Problem

In most situations, a sentiment or an opinion classified during sentiment classification may be a sarcastic sentiment and not the exact connotation of the word. But the question is, how can one be able to tell whether a sentiment is indeed positive and not just an ironic statement? This research seeks to pinpoint various sarcasm detection methods and approaches in sentiment analysis particularly the classification models used, challenges encountered, techniques used and among others.

2.4.3. Research Questions

- *What are the classification techniques that could be employed in undertaking sentiment analysis?*

  Motivation:

  The motivation here is to find out the popular classification technique considered especially when undertaking sarcasm in sentiment analysis. We would want to know the different classifiers that are considered by researchers and the widely used technique. For instance, the paper produced by Prasad et al. (2017), help us identify different classifiers and then proposed the simple and widely used technique.

- *What are the feature selection techniques that could be used?*

  Motivation:

  To help us know some feature selection approaches considered when undertaking sentiment analysis. There may be the non-textual and textual feature selection approach. This would also help us know the set of features in a preprocessed text such as Unigram. Consequently, we will get to know some proposed set of features.

- *What type of dataset or what dataset could be used?*

Motivation:

Different datasets are considered when undertaking sentiment analysis. What are these datasets? It may be a set of publicly available tweets or obtained private tweets that are classified and manually annotated by humans. The dataset also would pertain to a particular topic which we would want to know.

- *What are the challenges that could be encountered when undertaking sarcasm in sentiment analysis?*

Motivation:

Bharti et al. (2015b) indicated that sarcasm detection is a very challenging task. We would want to find out the challenges that make sarcasm in sentiment analyses very tedious.

- *What performance evaluation could be obtained?*

Motivation:

We would wish to ascertain the prediction performance attained using diverse evaluation measures? During the scrutiny of the prediction performance in text mining, we might have four possible outputs namely true positives, true negatives, false positives and false negatives. Computation of these might be based on precision, recall, f-score and accuracy.

2.4.4. Research Boundaries

In this systematic review, the authors consider how sarcasm is detected when conducting sentiment analysis. The population group that will be observed consists, therefore, of those publications which consider sarcasm in sentiment analysis during an opinion poll.

2.4.5. Review Method

The review method is based on the research protocol and it is in this section that the search strategy, the sources, the studies selection, and the selection execution are defined. The objective of this section is to provide the sources in which searches for primary studies will be executed. All references used in this research were included for analysis based on the following criteria:

1. All publications are between 2015 and 2020.
2. Journal Publications, conferences, magazines and books.
3. Papers with most of the keywords in the title.
4. Publications whose title contains sarcasm and sentiment analysis.
5. Publications whose abstract provide much enlightenment on sarcasm in sentiment analysis.

2.4.6. Classification of Papers

Papers used for this research are classified according to the publishers that published them. In this review, the following publishers were used to search for the research material;

1. IEEE Xplore
2. The ACM digital library
3. Science Direct
4. Scopus
5. Elsevier
6. Oxford Academic

Table 1 presents the distribution of the search results from each of the aforementioned publisher's site.

Table 1 Distribution of search results obtained from different Publisher's sites.

Publishers	Database search results of papers obtained	Final results after all exclusion mechanism applied
IEEE Xplore	47	7
ACM Digital library	2	2
Science Direct	263	1
Scopus	40	6
Elsevier	0	0
Oxford Academic	20	0
Total	368	16

2.4.7. Research Process

The databases chosen in this research study included publisher's sites which consist of published studies from their database. The search strings that were used in the databases were on the bases of keywords and when least results are obtained, some alternative words in research questions. Some keywords were concatenated to form a search string. All the selected databases were searched using the search string; Sarcasm in sentiment analyses

2.4.8 Inclusion Criteria

The studies considered are those that focus on sarcasm in sentiment analyses. The selected articles must be available in English and must be full-text articles. These articles are expected to be conferences, journals, magazines or book articles. These studies have more weight that gives empirical evaluations. The intention was not about rating any work but to ascertain the importance

of the work according to the domain proposed. Selected studies to be searched are papers from IEEE, ACM digital library, Science Direct, Elsevier, Scopus and Oxford Academic.

2.4.9 Exclusion Criteria

The exclusion criteria consisted of the elimination of the duplicate results of articles. Further exclusions of studies that did not give many details about sarcasm in sentiment analysis have been made.

2.4.10 Studies Selection

Once the sources have been defined it is necessary to describe the process and the criteria for studies selection and evaluation to reduce the likelihood of bias. Selection criteria should be decided during the protocol definition. The inclusion and exclusion criteria are based on the research questions. The researchers have therefore established that the studies must present new initiatives (from a maximum of approximately 5 years ago) which consider all kinds of discussions about sarcasm in sentiment analysis. The keywords are sentiment analysis, sarcasm, classification techniques, feature selection and an opinion poll. So the phrase, "Sarcasm in sentiment analysis" was used for the search in all the publisher's site.

In IEEE, conference papers and journals were considered and 47 of such papers showed up in the search. A resolution was consequently made to consider both the journal and conference papers together with a magazine. The further exclusion was done using the formatting conditions in Microsoft spreadsheet using the keywords "sarcasm" and "sentiment analysis". More so, the exclusion was considered by going through the abstracts of each of these articles to understand whether there were needed information as far as the topic and the keyword was concerned. 7 papers were obtained from the search. From the search using the keywords sarcasm and sentiment analysis in the ACM Digital Library database, only 2 conference papers published from 2015 to 2020 were

obtained from the search. No search result was obtained from Elsevier. 40 articles were obtained from the search in the Scopus database. Excel conditional formatting was conducted taking into consideration highlight cell rules with texts that contain both sarcasm and sentiment analysis. The authors retrieved five (5) publications from this exercise. But the paper "Hybrid method for sarcasm target identification to assist the sentiment analysis systems" could not be accessed for free and hence excluded. In the oxford academic database, 20 papers were retrieved from the search all other things being equal. However, advanced search by observation with much emphasis on the keywords; sarcasm and sentiment analyses did not fetch us any results. After 263 results being obtained from Science Direct only one article was considered after further excluding with emphasis on article titles consisting of both sarcasm and sentiment analysis. This review got rid of duplicate papers from different databases. All downloaded ACM digital library papers were also found in IEEE and hence discarded. Twelve different papers from all other databases were considered for the studies. Figure 2 depicts the statistics on the papers from the different databases primarily considered before and after the inclusion and extraction criteria application whilst Figure 3 shows the percentage statistics of papers obtained after the application of our inclusion and exclusion criteria. Figure 4 also shows the selection process used to derive our papers and lastly, Figure 5 is a graph showing the results of the scores of research questions answered.

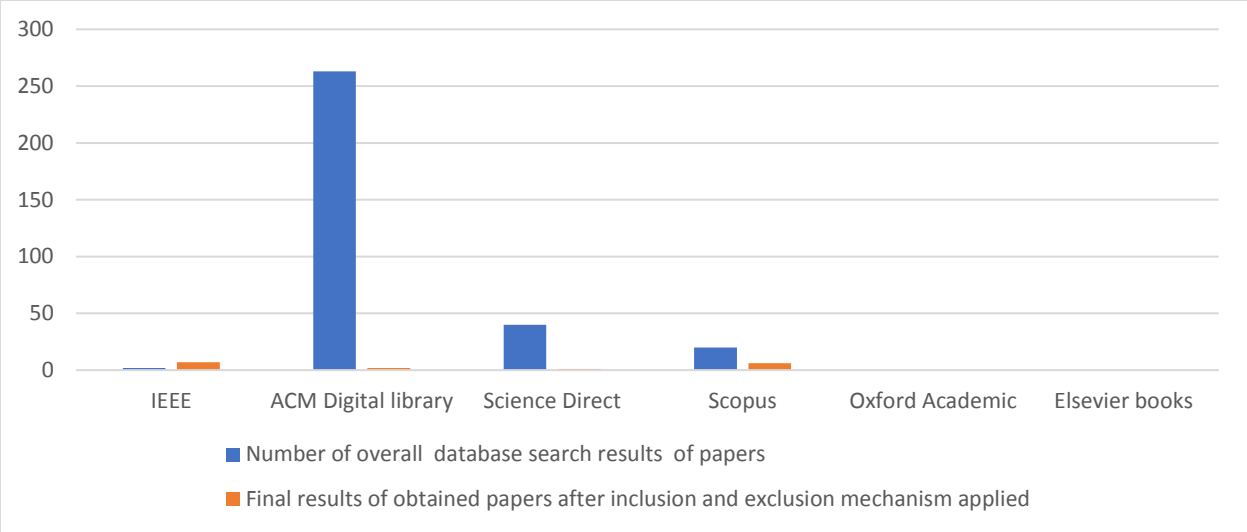


Fig 2. Database search results of overall papers obtained after the application of inclusion and exclusion criteria

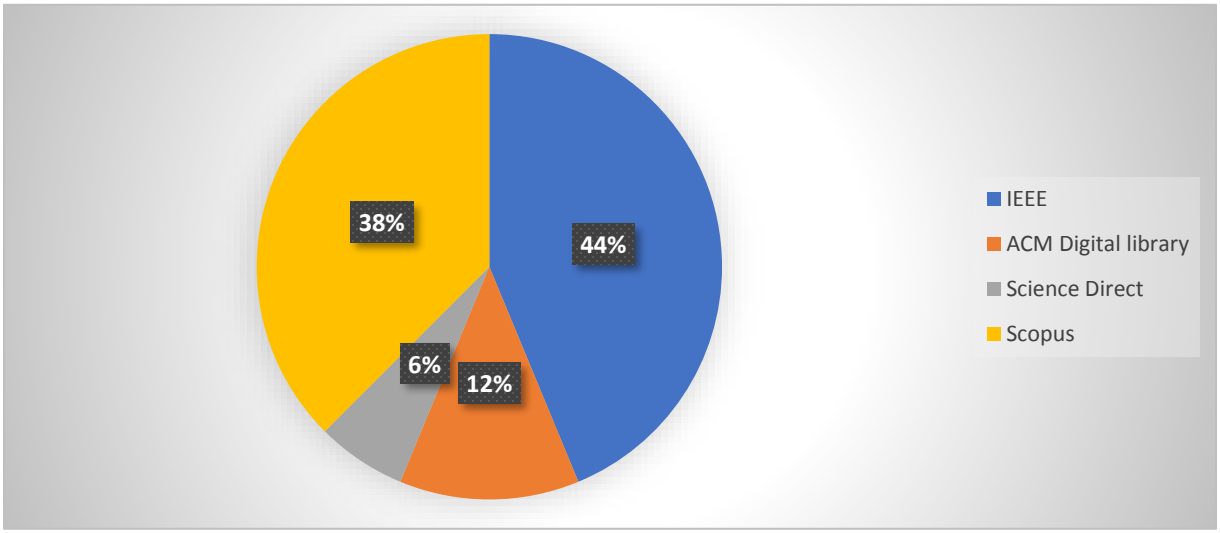


Fig 3. Statistics on papers obtained after inclusion and exclusion criteria applied.

Fig 4. Chronology of the selection process (SLR protocol)

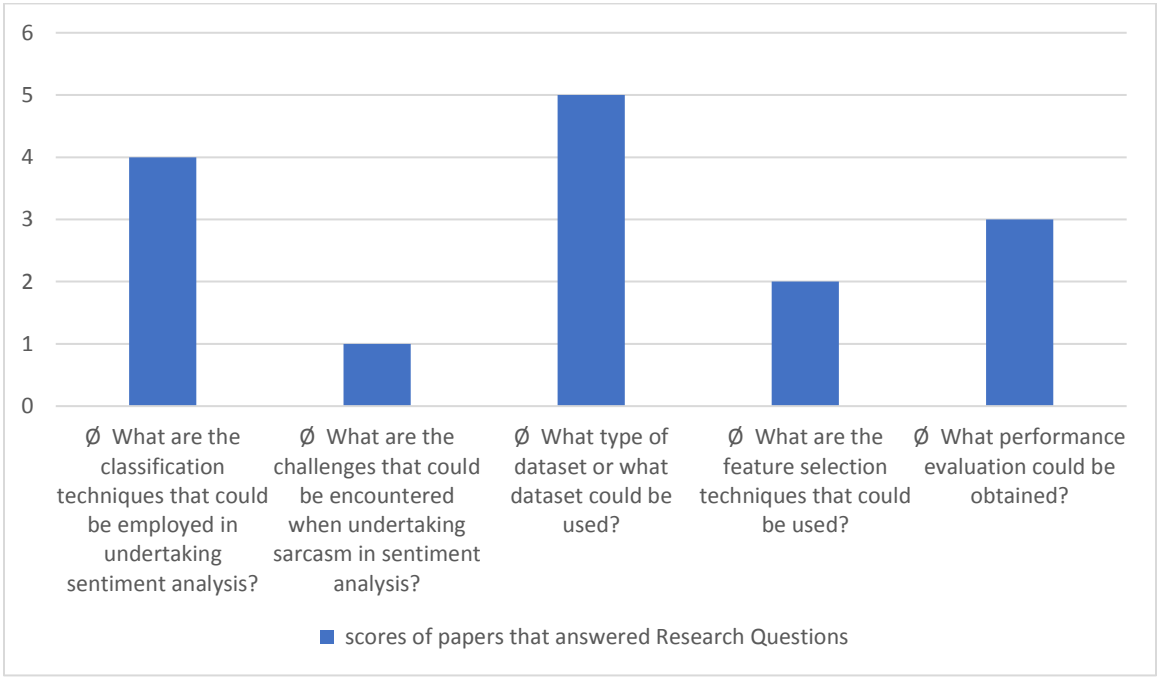


Fig 5. Scores of papers that were considered for research questions

2.4.11. Results and Discussion

RQ1: What are the classification techniques that can be employed in undertaking sentiment analysis?

Four (4) papers namely [LT1, LT11, LT16, LT14] see Table 3 in the appendix, were considered for this question. In the research conducted by Bouazizi et al. (2015), they performed the classification using Naive Bayes, Support Vector Machine (SVM), and Maximum Entropy classifiers [LT1]. In the paper produced by Prasad et al. (2017), the sarcasm detection in the proposed model is done using classifiers such as Decision Tree, Random Forest, Gradient Boosting, Adaptive Boosting, Logistic Regression and Gaussian Naive Bayes. It was concluded that the Decision Tree Classifier is a simple and widely used classification technique [LT11]. Magumater et al. (2019), applied the ultimate softmax classification [LT16]. In their study Dharmavarapu and Bayana (2019), extensively utilized AdaBoost and Naïve Bayes classification[LT14].

RQ2: What sort of dataset or what dataset could be used?

Five (5) papers namely [LT1, LT2, LT11, LT16, LT15] see Table 3 within the appendix, were considered for this question. Bouazizi et al. (2015), collected a group of publicly available tweets "classifiable" by humans and manually annotated them into "positive" and "negative". Tweets are selected to belong to one(1) of the ensuing topics: politics, phone reviews, sports, movie reviews, and electronic products [LT1]. Bharti et al. (2015a), did Database Collection for training 50,000 tweets which were obtained with the sarcasm hashtag (#sarcasm) from Twitter with keyword love, amazing, good, hate, sad, happy, bad, hurt, awesome, excited, nice, great, sick, etc. For testing, Tweets are collected in two categories (i) tweets with sarcasm hashtag and (ii) tweets without a hashtag [LT2]. In the study by Prasad et al. (2017), the dataset contains a collection of 2000 tweets

that have class labels of 1 or 0, where 1 means sarcastic and 0 means non-sarcastic. The dataset taken is that of about 2000 pre-classified tweets. The dataset contains two columns, Tweet and Label. The Tweet column contains the tweet, and the Label contains a binary label indicating whether the tweet is sarcastic or not [LT11]. According to Magumeter et al. (2019), their dataset16 consisted of 994 samples, each sample containing a text snippet labeled with sarcasm tag, sentiment tag, and eye movement data of seven readers. The authors ignored the eye-movement data in the experiments. Of those samples, 383 are positive and 350 are sarcastic [LT16]. In the paper produced by Suhaimin et al. (2019), they took into consideration a dataset which is manually arranged, the tweets in the dataset are physically named sarcastic and non-sarcastic dependent on human instinct which has organized an exact dataset for preparing. The physically ordered dataset is one of the presentation in this paper. The dataset contains an accumulation of 1000 tweets. The dataset taken is that of around 1000 pre-characterized tweets [LT15].

RQ3: What feature selection is being used?

Two (2) papers namely [LT5, LT7] see Table 3 within the appendix, were considered for this question. Different papers use different feature selection approaches. In the feature selection of the paper produced by Parwal et al. (2019), two types of features were extracted:

1) Non-textual features: From the "raw" tweets they first extracted 6 features by counting the number of positive and negative Hashtags, that of positive and negative Emoticons, and that of positive and negative slang words.

2) Textual features: After extraction of "non-textual features". There are several features taken from the pre-processed text: Unigram, Negativity, number of interjection words [LT5]. To measure sarcasm accurately, Bhan et al. (2018), proposed a set of features namely: Ngrams, Sentiments, Topics, Pos-tag, and Capitalization. This system uses sentiwordnet Dictionary to assign negative

and positive scores to each word and store it using its POS-ID. Using the above features, they trained their topic modeler using all tweets, then it generated the features for all tweets and then trained a classifier using these features [LT7].

RQ4: What are the challenges that could be encountered when undertaking sarcasm in sentiment analysis?

One paper which is [LT8] see Table 3 within the appendix, answered this question. The study conducted by Khullar and Singh (2019), identified these challenges with sarcasm detection: a) it would be utilized indirectly, also the authors might have the type of incongruity which makes it hectic and tedious to comprehend the sentiments.

b) The snide tweets communicate negative guesstimate utilizing positive words. In this way the classifier would erroneously dispense sentiments to these tweets

c) There is a wide usage of slang words, abbreviations, smileys, special symbols, and unstructured data which makes it quite tedious to identify sentiments [LT8].

RQ5: What performance evaluation could be obtained?

Three (3) papers namely [LT1, LT4, LT16] see Table 3 within the appendix, were considered for this question. Bouazizi et al. (2015), compared their proposed method to the baseline one presented by the n-grams model. They evaluated the two methods using one Key Performance Indicator (KPI) which is the accuracy. The results showed that their approach outperforms the baseline one. They obtained an accuracy that exceeded 80% using the 3 algorithms. However, SVM accuracy is better than that of Naive Bayes and Maximum Entropy [LT1]. In the study conducted by Bharti et al. (2015a), the first approach attains a 0.89, 0.81 and 0.84 precision, recall and f − score respectively. The second approach attains 0.85, 0.96 and 0.90 precision, recall and f – score

respectively in tweets with the sarcastic hashtag [LT4]. Magumeter et al. (2019), stated that their method outperformed the state-of-the-art by 3–4% in the benchmark dataset [LT16].

2.4.12. Threats to Validity

Since the researchers considered only a few databases with few papers being obtained and considered for the research questions, there is a possibility of a narrow study of research. Papers that were not written in English were not considered which implies the authors will miss key information in articles written in different languages. Inevitably, there were biases since some articles were not selected because their abstracts and conclusions were not conveying our expectations. Since the motivations behind the research questions are subjective, there are possibilities of data extraction inaccuracy and data synthesis biases. Only a few individuals carried out this research, the possibility of limitations in the research is probable because the knowledge domain in the subject matter may not be as broad as expected.

2.4.13 Conclusion

Research in sentiment analysis affirms how tedious the determination of sarcasm in sentiment is. Quite a few works have been done on this topic using different techniques, classifiers and methodologies. The trends have been studied and basic questions asked are answered taking into consideration some selected articles from different databases. The authors looked at the classifiers used, challenges encountered, the datasets used, their performance evaluation and the feature selection used. The study shows that 25% of the reviewed papers produce details on the classification techniques being used, 31.25% delivers more details on the datasets and preprocessing techniques, 12.25% offers detailed information on their feature selection, and 6.25% throw light on the performance evaluation and the different challenges encountered in undertaking sarcasm detection. Our results indicate that much has not been done in the area of sarcasm in

sentiment analysis. Therefore more research on determining sarcastic sentiment in sentiment analysis can be considered using different techniques, classifiers, tools and methodologies.

Table 3 in the appendix A, presents a selected article using a systematic literature review (SLR)

CHAPTER 3

METHODOLOGY

3.1. Introduction

Both qualitative and quantitative research approach is adopted. The data being used is secondary data obtained from an online repository and on twitter since raw data collection is labor-intensive and time-consuming. The interpretation of the values derived is seen as a major scientific proof of how the phenomenon works. This study is a design science research in that it aims at creating artifacts that provide services to serve a human purpose. As such, a system is designed and programmed to help solve a challenge. We go through the design cycle by meeting all basic requirements of problem identification and stakeholder consideration which forms the environment, a design and implementation of a model and its evaluation, and a consideration of the knowledge base which involves scientific theories and methods (Hevner, 2007).

3.2. Description of Datasets

Two different datasets were considered for our study. They included the driverless car dataset and the extensive twitter dataset which are all described in detail.

3.2.1. Driverless Car dataset:

Dataset used is driverless car data which talks about the emergence of driverless cars being powered by Google. The dataset is about 700 tweets obtained through twitter's social media platform. Since the tweets originally did not come with any labels such as sarcastic (1) or non-sarcastic (0), we utilized X-means clustering algorithm (due to its repetitive nature to form partition to achieve the Bayesian Information Criterion) with expert judgment in our quest to generate labels for our training; and therefore, we were able to represent zero (0) as a label for non-sarcastic tweets

and one (1) as a label for sarcastic tweets. The dataset was retrieved online via the UC Irvine (UCI) Machine learning repository in 2019. In applying the X-means clustering, different clusters were achieved, after which stratification was achieved based on these clusters. Each cluster consisted of a group of chronologically positioned data. Human expert judgement based on intuition was applied after the clustering in other to confirm the results obtained based on the clustering. The driverless car dataset was purposefully considered because it enables easier demonstration of how to undertake the determination of sarcastic sentiments when handling or using unlabeled tweets with a moderate total size.

3.2.2. Extensive Twitter Dataset

A broad-spectrum twitter dataset consisting of both sarcastic and non-sarcastic tweets with their labels; 0 for non-sarcastic and 1 for sarcastic was obtained online via the UCI machine learning repository in January 2020. The dataset consisted of 31,962 labeled tweets for testing.

3.3. Experimental language Used

We considered python for this project because it is stable, flexible, and has all the necessary tools and packages needed for the implementation of this project. Python as a language enables implementation from development to deployment and maintenance. It is simple and consistent, there is accessibility to enough and powerful libraries for AI particularly for machine learning and deep learning. More so, it is platform-independent and has a wide community of users which makes support and assistance easier and hence makes python popular.

It has concise and readable codes, enables the development of reliable systems, easier to learn, has easier to build models for machine learning, more intuitive, and facilitates the implementation of different functionalities and recommendable for collaborative implementation when multiple

developers are involved. It is a general-purpose language that implements complex machine learning, deep learning tasks, and enables one to build prototypes quickly and faster.

These and other reasons make python preferable for this study to other programming languages such as R, Scala, Julia, and Java

3.4. Deep Learning

A deep neural network is a feedforward artificial neural network architecture and is often a single hidden layer with their respective interconnection of neurons. Deep learning models have created enhanced prediction accuracy in a range of earlier studies (Mensah, Keung, Bennin, & Bosu, 2016; LeCun, Bengio, & Hinton, 2015; LeCun, Bengio, & Hinton, 2015; LeCun, Bengio, & Hinton, 2015; LeCun, Bengio, & Hinton, 2015; LeCun, Bengio, & Hinton, 2015; LeCun, Bengio, & Hinton, 2015)

3.5. Deep Learning versus Machine Learning

We learned from the research conducted by Srikanth (2017) that Sentiment analysis is one of the prominent areas researched in natural language processing. With the improvement in artificial intelligence, machine learning algorithms played a critical role in sentiment analysis applications after the age of conventional lexicon-based processing. Currently, nonetheless, Deep learning is the latest approach being used in the prediction of sentiments and research has been undertaken using the deep learning approach.

The paper produced by Goularas (2019), presents a comparison of assorted deep learning approaches used for conducting sentiment analysis using Twitter data. According to their research, two main categories of neural networks are used, convolutional neural networks (CNN) and the recurrent neural networks (RNN). The convolution neural network is performant especially in the area of image processing. On the other hand, recurrent neural networks are used in natural language

processing tasks. More so, the researchers evaluated and compared ensembles and combinations of convolutional neural networks and a category of RNN called the long short term memory (LSTM) networks. El-jawad & Hodhod (2017), in their paper also introduced a new hybrid system that uses text mining with neural networks for sentiment classification. They claimed that the hybrid learning approach was more efficient than a standard supervised learning approach like SVM and this conclusion arrived as a result of the 83.7% accuracy obtained. A deep learning approach entailing of the use of RNN with and word2Vec obtained an accuracy of 83%. Their study however failed to take into consideration sarcastic sentiments. Their work also did not consider combination of emotions and text for sentiment analysis.

3.6. Recurrent neural network

We learned from the paper by (Srikanth, 2017) that RNN employs a back propagation, unlike CNN which uses a feedforward network. RNN, therefore, processes sequential data with the aid of internal memory and it is built based on the principle that it is not always that human's reason from the onset. RNN, therefore, makes use of a previous word to forecast and predict the next word in the context. Natural language processing text analysis, voice recognition, and language translation systems are amongst the few applications that employ RNN in its development. Baktha et al. (2017), employs the use of RNN in undertaking sentiment analysis. They analyzed the performance of the three RNNs which are vanilla RNNS, long short-Term memory and Gated Recurrent Units. The dataset used was pre-trained word vectors from the Google News and the Amazon health product review dataset was used to evaluate its performance. Figure 6 gives a diagrammatical illustration of RNN.

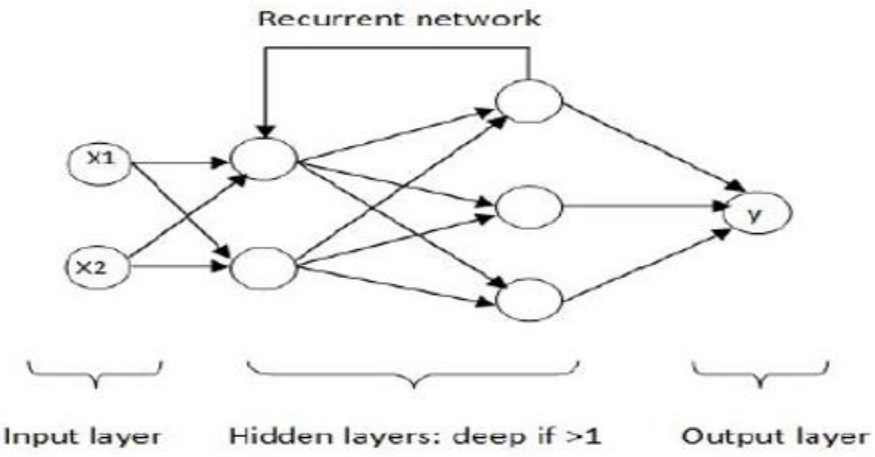


Fig 6. Diagrammatical presentation of RNN (Srikanth, 2017)

3.7. LSTM

According to Srikanth (2017), LSTM is an extension of RNN that can recall inputs over a long period. LSTM nevertheless has an advanced memory and not a simple internal memory. It can read, write, and delete data from its memory. It also provides an antidote to the challenges of RNN including vanishing gradient, and can decide on the information to keep or eradicate and the memory can be gated. The gates are input, forget, and output. Research by Ayata et al. (2017), shows that the application of LSTM in sentiment analysis produces better evaluation in performance than using machine learning approaches like SVM. Figure 7 gives a pictorial representation of LSTM with the gates.

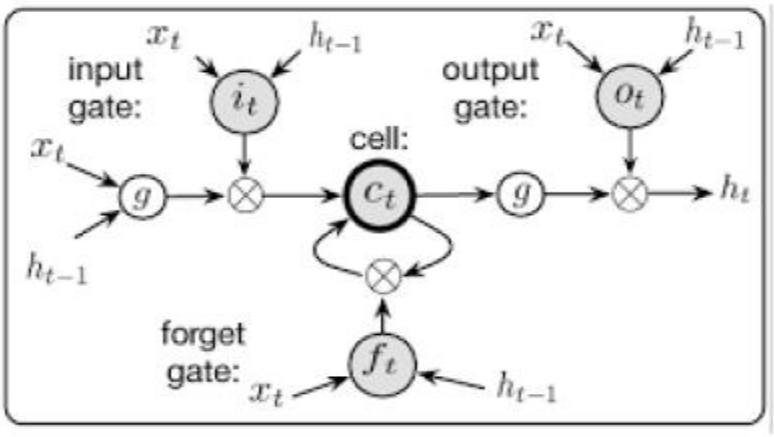


Fig 7. Diagrammatical presentation of LSTM with gates (Srikanth, 2017)

According to (Kumar et al., 2020), their experiment results discovered that a multi-head attention mechanism improves the performance of BiLSTM, and it accomplishes better than feature-rich SVM models. However they failed to perform a comparative analysis using LSTM in other to make a common claim.

From the study of (Son et al., 2019) using the training and test set accuracy, metrics were acquired to compare the proposed deep neural mode(Soft Attention-Based Bidirectional Long Short-Term Memory Model With Convolution Network) with convNet, LSTM, and bidirectional LSTM with/without attention. It was observed that the novel Soft Attention-Based Bidirectional Long Short-Term Memory Model with Convolution Network (sAtt-BLSTM) convNet model outperformed others with a superior sarcasm classification accuracy using the proposed datasets. Their paper reveals that their proposed model is obtained from LSTM with a connection of multiple hidden layers. This would consequently lead to higher processing cost and complexities as compared to LSTM. Considering their SemEval dataset, LSTM performs better than BLSTM as far as precision is concerned. The precision for LSTM was 86.78 whilst that of BLSTM was

81.61.This indicates that LSTM can perform better depending on the type of dataset. It can therefore be concluded that LSTM though cheaper could also perform efficiently and better.

3.8. Sentiment Analysis with LSTM

In this project, there is an implementation of a Long Short Term Memory that performs sentiment analysis and the detection of sarcastic comments.

Using LSTM rather than a strict feedforward network is more accurate since we can include information about the sequence of words.

There is the usage of a driverless car dataset, accompanied by sentiment labels: sarcastic or non-sarcastic.

3.9. Implementation Using LSTM

This section provides detailed and pragmatic discussion on the implementation of our framework by using LSTM.

3.9.1. Loading in and visualizing the data

The data is loaded through the opening of text files and reading in data as text from our directory. Data is read from an excel file and could be printed in shape and columns. This is possible in python with the help of libraries such as numpy and pandas. Libraries such as seaborn and matplotlib also help us with our data visualization to view the loaded data. This is more clear when newline characters from each record in a data frame are removed. The results of our visualized data presented a graph with labels of counts and sarcastic or non-sarcastic.

3.9.2. Data pre-processing

Getting your data into the proper form to feed into the network is the first step when building a neural network model. Since we are using embedding layers, there is the need to encode each word with an integer. Cleaning up the data is necessary.

The processing steps are:

- Periods and extraneous punctuation must be get rid of.
- Also, you might notice that the reviews are delimited with newline characters \n. split the text into each comment using \n as the delimiter, to deal with those.
- Then a combination is necessary to get all the comments back together into one big string.

To commence with, punctuations are removed. Then all the text without the newlines is obtained and split into individual words. This is possible with the help of string and punctuation libraries.

3.9.2.1. Encoding the words

The embedding lookup requires that we pass in integers to our network. The easiest way to do this is to create dictionaries that map the words in the vocabulary to integers. After then, there can be a conversion of each of the comments into integers so they can be passed into the network. We do this by encoding the words with integers and build dictionaries that map words to integers. The integers ought to start at 1 before its storage in a list so that input vectors can be padded with zeros. So by implementation, we make use of the collection and counter libraries and then build dictionaries that map words to integers before storage of the tokenized reviews. Text implementation of our dictionary produces 3,084 unique words for the driver's car dataset and then 47,521 for the broad-spectrum dataset.

3.9.2.2. Removing Outliers

As an additional pre-processing step, the comments or reviews ought to be in good shape for standard processing. That is, the network will expect a standard input text size, hence, there is the need to shape the reviews or comments into a specific length. This task can be approached in two main steps:

1. Getting rid of extremely long or short comments; the outliers
2. Padding/truncating the remaining data so that we have reviews of the same length.

Before padding the comment or review text, we should check for reviews or comments of extremely short or long lengths; outliers that may mess with our training.

With the driverless car dataset, a zero-length review of 1 and a maximum review length of 30was obtained. On the other hand, a zero-length review of 1 and a maximum review length of 30 was obtained using the extensive twitter dataset.

The challenge here is that we get a zero review length and the maximum review lengths become too many steps for our LSTM; hence, the removal of any super short review is expedient and that of super long review is also crucial. By so doing, our model can be trained efficiently as a result of the removal of the outliers.

To implement this:

1. remove any review with zero-length from the review list
2. Get indices reviews with the length being zero
3. Remove the zero-length reviews and their labels.

3.9.2.3. Padding sequences

In other to tackle with both short and very long comments and reviews, a step will be taken to pad or truncate our comments to a specific length (seq_length). For comments smaller than some seq_length, we decided to pad with zeros. Again, considering reviews longer than seq_length, we can cut them into the first seq_length words. A good seq_length, in this case, is 71 (Max length of a tweet) for driverless car dataset and 71 seq_length was also defined for the extensive twitter dataset. Maximum length of each tweet for both datasets do not exceed 71, and hence it is a justifiable value to be considered as a good sequence length to reduce processing time. Our final features array should be a 2D array, with as many rows as there are comments or reviews, and as many columns as the specified seq_length. This is not trivial and there are a bunch of ways to do this. But, if you are going to be building your deep learning networks, you're going to have to get used to preparing your data.

By implementation, our data should come from an integer array of reviews to be fed into our network. Each road ought to have the maximum length of tweets (seq_length), thus it should be seq_length elements long.

If the review is shorter than the seq_length words, there should be a left pad of zeros. On the contrary, if the review length is longer than the seq_length, use only the first seq_length words as the feature vector. For instance if the seq_length is 12 and the review is [10, 112, 127,130], the resultant padded sequence will be [0, 0, 0, 0, 0, 0, 0, 0, 10, 112,127, 130]. Our features for both driverless car and the extensive twitter datasets could be saved as a CSV file by the aid of NumPy, asarray and savetxt libraries

3.9.2.4. Training, Validation, Test

With our data in nice shape, there is the need to divide it into training, validation, and test sets.

There is the creation of training, validation, and test sets. There is the need to create sets for the features and the labels, train_x and train_y, for example. Next is to define a split fraction, split_frac as the fraction of data to keep in the training set. Usually, this is set to 0.8 or 0.9. Whatever data is left will be split in half to create the validation and *testing* data. The fraction 0.8 or 0.9 are just experimental values that give us accurate results and model for our training without any problem such as overfitting unlike other values such as 0.3 ,0.4 or 0.5.

3.9.2.5. DataLoaders and Batching

Following the creation of training, test, and validation data, it becomes imperative to create Data Loaders for the data by following two steps:

1. Create a known format for accessing our data, using the Tensor Dataset which takes in an input set of data and a target set of data with the same first dimension, and creates a dataset.
2. Create DataLoaders and batch our training, validation, and test Tensor datasets.

This is a substitute for creating a generator function which is for batching the data into full batches. The output is displayed as a tensor matrix.

Libraries such as a torch, torch.utils.data, TensorDataset, Data Loader, learn, oversampling and SMOTE should be imported to create Tensor datasets, define data loaders batch size, ensure shuffling of training data and perform sampling. More so, length derived after sampling will be used to obtain a batch of training data.

3.9.3. Sentiment Network with PyTorch

This is where the network will be defined.

The layers are as follows:

1. A layer that converts tokens into a particular embeddings
2. hidden state size and number of layers that defines an LSTM layer
3. The LSTM layer outputs to the desired output size that is mapped by a fully-connected layer of an output
4. The return of only the last sigmoid output as the output of this network that is done by a sigmoid activation layer that converts the results to 0 and 1.

3.9.4. The Embedding Layer

It is recommendable to add an embedding layer because there are 74000+ words in our vocabulary. It creates a lot of inefficiencies to one-hot encode that numerous classes. So, instead of one-hot encoding, one can have an embedding layer and use that layer as a lookup table. You could train an embedding layer using Word2Vec, then load it here. But, it is fine to just make a new layer, using it for only dimensionality reduction, and let the network learn the weights.

3.9.5. The LSTM Layer

There is the creation of a Long Short Term Memory to be utilized in our recurrent network, this takes in an input size, a hidden dim, several layers, a dropout probability (for dropout between multiple layers), and a batch first parameter.

Most of the time, the network will have a better performance with more layers; between two to three. Adding more layers allows the network to learn complex relationships.

The creation of LSTM is implemented by first checking if Graphics processing Unit

(GPU) is available and if it is then we do the training on GPU. Here we also define the LSTM model that will be used to execute sentiment analysis. The model is initialized by setting up the layers. This is where we employ an embedding and LSTM, dropout, linear and sigmoid layers. A

forward pass of our model on some input and hidden state is performed. This leads to defining a forward function that considers embedding and LSTM, stack up LSTM outputs, dropout and fully-connected layer, sigmoid function, shaping of batch size, and a return of last sigmoid output and hidden state. The hidden states are initialized by creating two new tensors.

3.9.6. Instantiating the Network

Here, the network is instantiated. This is commenced by defining the hyperparameters and the model hyperparameters. Some terminology to understand here:

- vocab_size: represents size of our vocabulary or the range of values for our input, word tokens.
- output size: represents size of our desired output; the number of class scores we want to output
- embedding dim: represents number of columns in the embedding lookup table; the size of our embedding.
- hidden_dim: represents number of units in the hidden layers of our LSTM cells. Usually, larger is better performance-wise. Common values are 128, 256, 512, etc. These are just common experimental index of two values being considered to give us accurate results.
- n_layers: represents number of LSTM layers in the network. Typically, it is between 1-3. This is one of the basic rules in choosing the number of hidden neurons. However, 2 layers are considered better Marco (2013).

3.9.7. Training

It is expedient also to be using a new kind of cross-entropy loss, which is designed to work with a single sigmoid output. BCELoss, or Binary Cross-Entropy Loss, applies cross-entropy loss to a single value between 0 and 1. We also have some data and training hyperparameters defined as:

- LR: represents learning rate for our optimizer.
- epochs: represents number of times to iterate through the training dataset.
- Clip: represents the maximum gradient value to clip at (to prevent exploding gradients).

3.9.8. Testing

The few ways for one to test the network are as follows:

- **Test data performance:** we will see how our trained model performs on all of our defined test data above. We will calculate the average loss and accuracy over the test data.
- **Inference on user-generated data:** Second, we will see if we can input just one example comment at a time and without a label, and see what the trained model predicts. **The inference is** looking at new, user input data like this, and predicting an output label.

3.9.9. Trying out test

Now that we have a trained model and a predict function, you can pass in *any* kind of text and this model will predict whether the text has a sarcastic or non-sarcastic sentiment.

3.10. Network Architecture

This section presents a discussion on the network architecture used and how it was passed to the LSTM cells.

3.10.1. Passing in words into an embedding layer.

There was a necessity for an embedding layer since we were having lot of words, so therefore needed an efficient representation of our input data preferably to a single encoded vector. We can train an embedding with the Skip-gram Word2Vec model and use those embedding as input. Nevertheless, it is recommendable to have the embedding layer and let the network learn a different embedding table by their own. When this happens, we say the embedding layer is for dimensionality reduction, and not learning of semantic representations.

The new embedding that was passed to LSTM cells following the input of words were passed to an embedding layer. LSTM cells add recurrent connections to the network and enable us in our ability to include information about the sequence of words in the data under consideration.

Additionally, our LSTM results will go to a sigmoid output layer. The sigmoid function will be used because non-sarcastic and sarcastic = 0 and 1, respectively, and a sigmoid will output predicted sentiment values between 0-1.

The emphasis is not on the sigmoid outputs except for the very last one; the rest can be ignored. Calculation of the loss will be done by comparing the output at the last time step and the training label (1 or 0).

3.11. Performance Evaluation

Table 2 Confusion Matrix for computing precision and recall

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

For each corpus of extracted sentiment from a given opinion poll, we respectively partition the corpus into q strata or subsets. We calculate the confusion matrix and assess the prediction performance by utilizing the precision, F1-Score and recall evaluation indicators. When probing into the prediction performance of text mining, we could have four possible outcomes - true positives, true negatives, false positives, and false negatives. For instance, in a classification task, when the text mining algorithm appropriately classifies the relevant documents for a given query, then the number of relevant documents is labelled to as the true positives (Jain, Agrawal, Goyal, & Aggrawal, 2018). Again, when trivial documents are wrongly classified as relevant documents after the search process, the number of such irrelevant documents then is stated to as false negatives. In the same way, if relevant documents are wrongly classified as trivial documents after the search process, then the number of such relevant documents is termed as false positives (Jain et al., 2018). When irrelevant documents are decorously classified as trivial documents after the search process, then the number of such irrelevant documents is referred to as true negatives (Jain et al., 2018). The Confusion matrix for computing the precision, F1-Score and recall measures are expressed in Table 2.

Precision is defined as the percentage of the number true positives (correctly predicted positives) by the search process to the total number of misclassified relevant documents observation (false positives) plus the number of relevant documents (true positives) that was recovered by the same

search process. On the other hand, recall unlike the precision is the probability that a relevant document is recovered from the search process. That is, recall is the ratio of the correctly predicted positive observations to the total number of existing or actual relevant documents. F1 score is defined by (Jain et al., 2018), as the weighted average of the precision and accuracy. It considers both false positives and false negatives (Mensah, Keung, Svajlenko, Ebo, & Mi, 2018)

.

$$Precision\ (P) = \frac{TP}{TP+FP} \tag{3.11.1}$$

$$Recall\ (R) = \frac{TP}{TP+FN} \tag{3.11.2}$$

$$F1\ Score = 2 * \frac{Precision(P)*Recall(R)}{Precision+Recall} \tag{3.11.3}$$

To be able to plot our confusion matrix, we import the intertools, NumPy and matplotlib library. It could be normalized or without normalization and the nearest interpolation can be shown with their labels being true or predicted. For stacked data for the confusion matrix of training and validation, we track loss. The other processes involve tracking loss, initializing hidden state and getting predictions, iterating over test data and stacking predictions and labels before calling a get confusion matrix with it associated data loader as a parameter. Data is trained for the confusion matrix and the class involves sarcastic and non-sarcastic statements.

3.12. Framework for Sarcasm Detection

The pseudocode for framework section below contains pseudocode for our framework whilst Appendix F presents a general overview of Sarcasm detection operator flowchart. Our prediction

model employs the Long Short Term Memory learning algorithm. If the dataset being used is unlabeled, a CLUSTER+EXPERT JUDGEMENT operator helps us to do the labelling through Clustering and the application of human expert Judgement. The data is then trained and validated by the assistance of the Train & Validate operator and then finally, Classify & predict function is employed in the determination of sarcastic sentiments.

So the three major operators which are considered are Cluster+Expert Judgement, Train & Validate and Classify & predict:

3.12.1. Cluster+Expert Judgement (Clustexpert)

Once the dataset obtained from the repository is unlabeled, we utilize the X-means clustering algorithm due to its repetitive nature to form partition to achieve the Bayesian Information Criterion. In applying the X-means clustering, different clusters are obtained, after which stratification is attained based on these clusters. Each cluster consists of a group of chronologically positioned data. The result of our clustering is noted and then an expert judgment that is based on human intuition and wisdom is applied manually for verification. These two steps would assist us in generating accurate labels for our training and validation. Consequently, there would be a representation of zero (0) as a label for non-sarcastic sentiment and one (1) as a label for the sarcastic sentiment.

3.12.2. Train & Validate

With our data in nice shape sets of features, we can proceed with training and validation for an unbiased evaluation using a supervised learning method. Following the creation of training and validation data, it becomes imperative to create Data Loaders to batch our dataset and apply it to a defined Sentiment Network. In our case, The LSTM layer employed converts the tokens into a

particular embedding using a defined hidden state size and number of layers. The layer outputs to the desired size which is mapped by a fully-connected layer of output and a sigmoid activation layer do the conversion.

3.12.3. Classify & Predict New Instances

You can pass in any kind of text and this model will predict through our prediction function as to whether the text has a sarcastic or non-sarcastic sentiment. This can be done by using our data performance or by inference on user-generated data. In our case, we calculated the average loss and confusion matrix indicators like accuracy over the test data to analyze the prediction values and classified instances of the sentiments as sarcastic or non-sarcastic.

3.12.4. Pseudocode for Framework

```
Procedure determineSarcasm (Dataset)
        1. Begin
        2. foreach Dataset do
        3.      if Dataset_labelled do
        4.            train_validate(Dataset)
        5.      Else applyClustExpert(Dataset)
        6.            if Dataset_labelling_succesful do
        7.                  train_validate (Dataset)
        8.      foreach sentiment in dataset do
        9.            Classify_predict (sentiment)
        10.           //Determine whether the statement is sarcastic
        11. Ends
```

3.13. Conclusion

In conclusion, this section provides descriptive details of our methodology. A framework has been introduced that comprises of three operators namely Cluster+Expert Judgement, Train & Validate and Classify & predict; that facilitates sentiment classification. We considered LSTM for the implementation of our framework. The study of (Kumar et al., 2020) and (Son et al., 2019), employed multi-head Attention-based and Bidirectional LSTM respectively. However, their

experiment reveals that their methodology is expensive and much complicated. By inference from their study, it could be deduced that a lot of resources need to be utilized to ensure optimization. We therefore, employed a cheap, simple, efficient and straight forward approach by employing purely LSTM to produce a framework that helps in the determination of sarcastic statements when undertaking sentiment analysis.

CHAPTER 4

RESULTS AND DISCUSSION

4.1. Visualization

The dataset is read from an excel file and displayed in data shape and columns. About seven hundred (700) driverless dataset and thirty-one thousand nine hundred and sixty-two (31,962) extensive tweets are displayed with their associated sarcastic or non-sarcastic label. The use of libraries such as seaborn and matplotlib helps us visualize label values as seen in Figure 8 and 9.

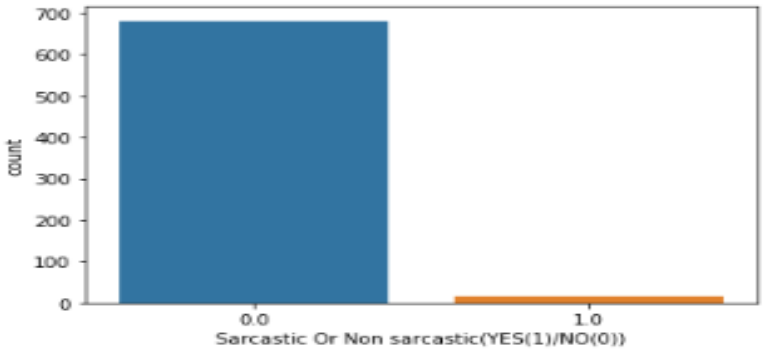


Fig 8. Visualization of sarcastic and non-sarcastic sentiments for the driverless dataset

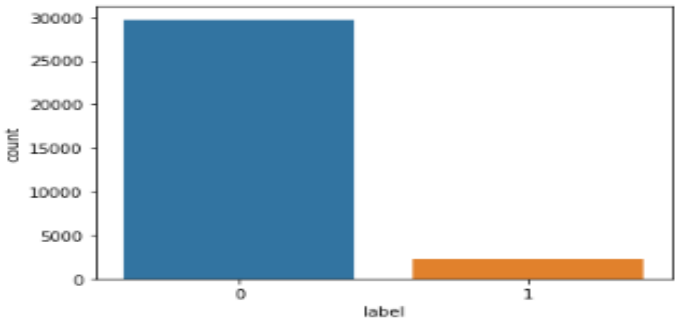


Fig 9. Visualization of sarcastic and non-sarcastic sentiments for extensive twitter dataset

More so, results may be put in a data frame and then written to a CSV file which can later be read as a text file. During data preprocessing, the result of padding sequences undertaken after encoding and removing outliers ought to be a 2D array features, with as many rows as there are reviews and as many columns as the specified sequence length as depicted in Figure 10 using the driverless car dataset.

```
[[ 261  200   52  152  953    4  580   28    5]
 [ 410  956    2  411   50   27  202  323  160]
 [ 585   12  959    6   44  413  262  586  960]
 [   0    0    0    0  324    1   88   71   18]
 [ 968  263  587  969  970  133    4  264  118]
 [   0    0    0    0    0   31   33  326    1]
 [ 116   10   34   23  153  106  178  588   51]
 [   0    0    0   99    5  107  266  589  972]
 [   0    0    0   74   80    1  973   80  327]
 [  42   13   26  976   81  977  233  978  979]
 [ 981  154  100  329  154  982    2   82  983]
 [ 125  101  984  119  985    6  986  987  234]
 [   0    2  180    1  330   14  134  988    9]
 [   0    0    0    0    0  989   20  593    4]
 [   0    0  416  235    4  990  991  992    4]
 [ 993   14  994  594  995    8  181   22  126]
 [   0  997  998   34   45   20   45    1  999]
 [   0    0    0    0    0  595    7  596   14]
 [ 102  420  598   12    9 1002   16   44   40]
 [1003  155    4   22 1004    2 1005    8    3]
 [1007  599  135 1008 1009    6 1010 1011 1012]
 [   0    0    0   83  332   37  156  164  182]
 [   0    0  270  600   37  333  422    2  423]
 [ 603    8    1   72 1015  136   93  205 1016]
 [   0    0    0    0   16    1  604 1017 1018]
 [   0   37  425  426 1020 1021  135  136 1022]
 [  43 1028    1 1029   11  427   41  606  203]
 [  47  180  271 1033  272   23   19    3   35]
 [1036 1037 1038   37  608 1039  580 1040 1041]
 [   0  335  127 1043   11    1  120   13   38]]
```

Fig 10. 2D array features

4.2. Training, Validation and Test Sets

Normally, there is a need to split our data into training, validation, and testing. Again, using the driverless car and the extensive twitter dataset, the result obtained after creating the training, validation, and test sets are depicted in Figures 11 and 12 in Table 4 respectively for driverless and extensive twitter datasets.

Table 4. Training, validation and test set results

```
                    Feature Shapes:
Train set:          (559, 71)
Validation set:     (70, 71)
Test set:           (70, 71)
```

Fig 11. Feature shapes of train, validation and test sets for driverless car dataset

```
                    Feature Shapes:
Train set:          (25569, 71)
Validation set:     (3196, 71)
Test set:           (3197, 71)
```

Fig 12. Feature shapes of train, validation and test sets for extensive twitter dataset

```
(tensor([[  0,   0,   0, ...,  28,   5, 581],
        [  0,   0,   0, ..., 323, 160, 957],
        [  0,   0,   0, ..., 586, 960, 961],
        ...,
        [  0,   0,   0, ...,   3,  35,  36],
        [  0,   0,   0, ..., 771,  29, 719],
        [  0,   0,   0, ..., 326,   1, 265]]), tensor([0, 0, 0,  ..., 1, 1, 1]))
```

Fig 13. Data loading and batching results using tensor Dataset considering driverless car dataset

```
(tensor([[   0,    0,    0, ...,   93, 7974,  472],
        [   0,    0,    0, ..., 7976, 16393, 10400],
        [   0,    0,    0, ...,   63,   25,  3339],
        ...,
        [   0,    0,    0, ...,  236,  1144,  240],
        [   0,    0,    0, ...,  899, 29308,  2015],
        [   0,    0,    0, ...,  366,  1022,   47]]), tensor([0, 0, 0,  ..., 1, 1, 1]))
```

Fig 14. Data Loading and Batching results using tensor Dataset considering the extensive twitter dataset

Following the creation of training, testing, and validation of the data, we create Data Loaders by defining a batch size for the data using the tensor dataset, then constructed data loaders and batch the dataset. In the process of doing this, the training data needs to be shuffled and a batch of training data needs to be obtained. This is an alternative approach to creating a generator function for batching our data into full batches.

Figures 13 and 14 in Table 4 depict the results after Data Loading and Batching using the tensor dataset considering the driverless car dataset and the extensive twitter data respectively.

4.3. Sampling

Appendix C shows the result on our sampling considering the driverless dataset. Length after sampling: 1094. Count 0's (zeros) sarcastic statements after sampling: 547. Appendix D shows the result on our sampling considering the extensive or broad-spectrum dataset.

Length after sampling: 47542

Count 0's sarcastic statements after sampling: 23771

4.4. Training

In building a model, a new kind of cross entropy loss was used, which is designed to work with a single sigmoid output. BCELoss, or binary Cross Entropy Loss, applies cross entropy loss to a single value between 0 and 1.

Some data and training hyparameters we had are:

Learning rate for optimizers which is represented by Lr

Number of times to iterate through the training dataset and which is our Epoch

The maximum gradient value to clip at which is our Clip

In defining our loss and optimization function we used a learning rate (lr) of 0.001 that is lr=0.001 and the Binary Cross Entropy Loss function being our criterion. The learning rate as a hyper parameter which is used in training neural network that has positive value normally range between 0.0 and 0.1. The defined learning rate is not too large and hence it will not converge too quickly to a suboptimal solution and also the process does not get stuck. Our optimizer therefore made use of the defined learning rate and other net parameters functions needed to ascertain the optimizer.

More so, during the training of the program, an epoch of 4 was assigned. Three to four (3-4) epoch is approximated when validation loss stop decreasing is noticed. After defining an initial counter and every print point the paramount element known as the clip or gradient clipping is defined to prevent exploding gradients.

With these in place, we move our model through a gpu then call on the net training function, train by initializing hidden state for quite a number of epochs and performed a bath loop to create new variables for the hidden state or backdrop over the whole training history. We also obtained the zero accumulated gradients, get the output from the model, calculate the loss and perform backprop. There is the exploding gradient problem in RNN and LSTM and hence, the use of clip_grad_norm function assists to avoid the gradient problem.

Loss Statistics:

We can do the loss statistics by obtaining the validation loss. In getting the validation loss, we create backprop through the entire training history or new variables for the hidden state or and then undergo the net training to print each Epoch, Step, Loss and validation Loss in the defined format by calling the format functions we define in our program.

Appendix E shows the results obtained using our driverless car dataset without that of the extensive twitter dataset due to its enormousness in Appendix B.

4.5. Testing

As already discussed, our network can be tested by test data performance which helps us view our trained model performs on all of the defined test data and then calculate the average loss and accuracy over the test data.

Another way to test our network is by inference on user-generated data which enables us to have the possibility of inputting just one comment or review as an example at a time and without a label and see what the trained model predicts. The fact is that we can look at new and user input data as such and predict an output model; we are justified to refer to this as inference.

So in the course of the testing, we get the data loss and accuracy by tracking loss, initializing hidden state using the batch size, iterating over the test data by here also creating a new variable for hidden state or back prop through the entire training history, getting predicted output, calculating loss, converting output probabilities to a predicted class of (0 or 1) and comparing predictions to true label

We can then obtain and print the average test loss and get accuracy over all the test data. Using our driverless dataset, a test loss of 0.136 and a test accuracy of 0.757 was obtained. More so, a test loss of 0.309 and a test accuracy of 0.905 was obtained using our extensive twitter dataset. These results depends on the size of the datasets and its composition. We are getting lesser test loss for driverless dataset because the dataset size is small than the extensive twitter dataset and therefore few loss will be attained. More so, we are obtaining higher test accuracy for the extensive twitter dataset comparably to the driverless dataset because the dataset size for the extensive twitter

dataset is far greater than the driverless car dataset and hence, a higher probability of obtaining more sarcastic sentiments in the tweets.

4.5.1 Inference on Test Review:

An inference on our test review can now be conducted by changing the test review or comment to any desired test and then check or test manually if the model does a correct prediction. Considering these examples:

i. Sarcastic statement-“The day I never have to deal with Fred will be one of the best days of my life indeed”

ii. Non-sarcastic review-“everything will alright”.

How does the test work with each of these examples? It starts by tokenizing the review by getting rid of punctuations and splitting by spaces to get the needed tokens in an array and then derive their integer values. We then undertake test sequence padding. We generate the features by passing in the generated integers and the sequence length. The feature can then be printed. Figure 15 is our obtained feature using our test review

```
[[   0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    1   59    2  130   19    4
   789   15  570  158   48 3074   45   77   11    1  124  399   11    9
   141]]
```

Fig 15. Obtained feature from test review

We can then do test alteration to tensor and pass it into our model. Here we obtained torch. Size ([1, 14]) as feature tensor size. So basically, a predict function accepts as parameters test review, sequence length, and net.

Through a predict function, we tokenize the review, pad tokenized sequence, convert to tensor to pass into our model, initialize hidden state, get the output from the model, convert output probabilities to a predicted class of (0 or 1) even before the output value is rounded and then print custom response as sarcastic or non-sarcastic. So with our reviews, we defined a sequence length of 141 and pass it through our predict function. Our test result for the sarcastic statement was a prediction value pre-rounding 0.998875 when we used a sarcastic tweet! And that of the non-sarcastic was a prediction value pre-rounding 0.000055. Same pre-rounding values will be obtained in terms of prediction value if we use different values for sequence length. In other words, a sequence length of 141 was defined because it is the maximum number of characters a tweet can contain. Any number less than that will just be padded with zeros.

4.6. Confusion Matrix

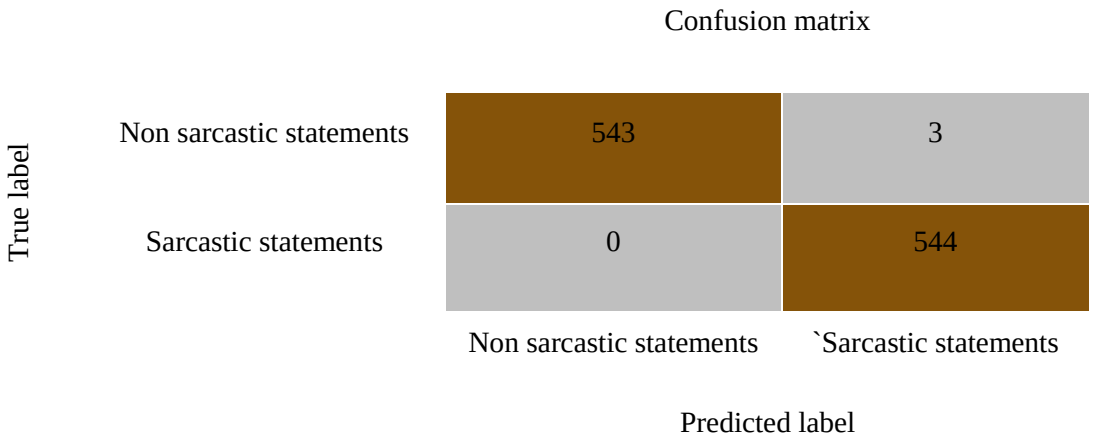
The Confusion matrix for computing the precision, F1-Score and recall measures has been expressed as defined our methodology.

Precision is derived by the percentage of the number true positives (correctly predicted positives) by the search process to the total number of misclassified relevant documents observation (false positives) plus the number of relevant documents (true positives) that was recovered by the same search process.

The recall is derived by the ratio of the correctly predicted positive observations to the total number of existing or actual relevant documents. Whilst F1 score is obtained from the weighted average of the precision and accuracy. It considers both false positives and false negatives.

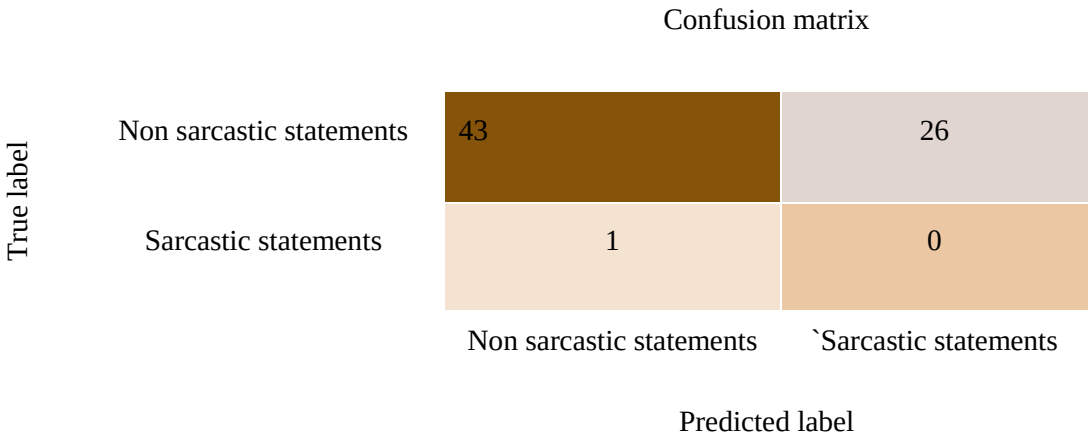
With our confusion matrix without normalization using the driverless car dataset for training, our derived precision was 0.995, the recall was 1.000 and the F1 Score was 0.997. Table 5 shows our tensor matrix for trained data using a driverless car dataset.

Table 5 Confusion matrix for trained driverless car dataset



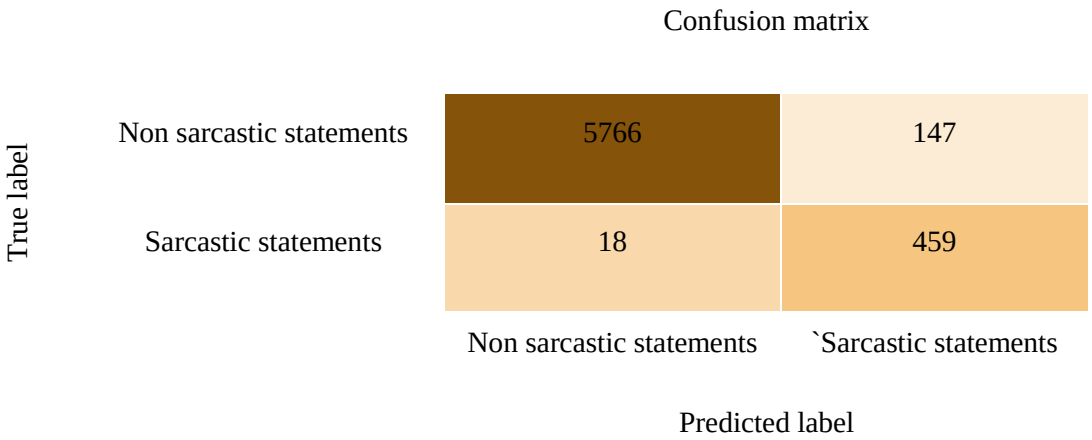
With the validation data, our precision was 0.632, the recall was 0.977 and the F1 Score was 0.761 using the driverless car dataset. Table 6 depicts the matrix results of the validated data using the driverless car dataset involving its true and predicted label.

Table 6 Confusion matrix for validated driverless car dataset



Using the extensive twitter dataset, our precision was 0.975, the recall was 0.997 and the F1 Score was 0.986. These different results for the different datasets depends on the size of the datasets and their composition for each performance evaluation measure considered. Table 7 depicts the results of our validated data without normalization using the extensive twitter dataset.

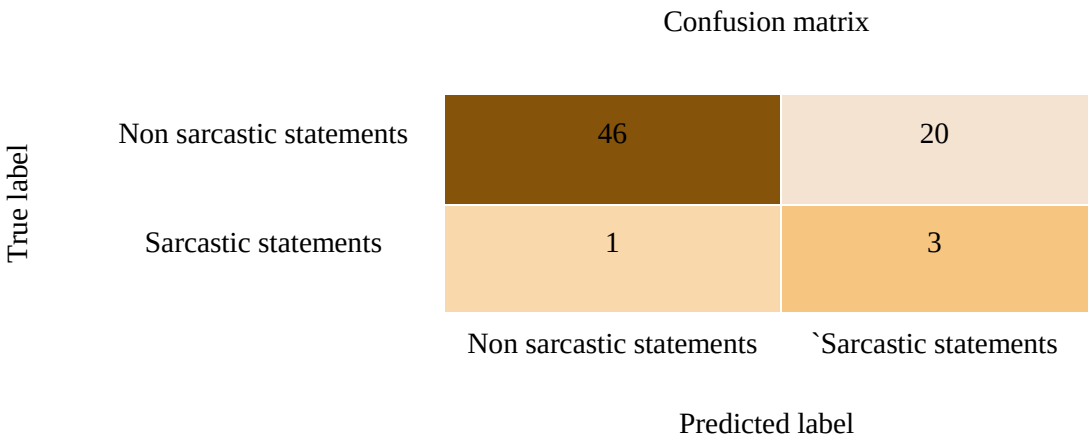
Table 7 Confusion matrix for validated extensive twitter dataset



Additionally, there is a plot of a confusion matrix for the test data with the same defined formula for precision, recall, and F1 Score. Precision was 0.679, the recall was 0.939 and the F1 Score was 0.800 using the driverless car dataset. The different values are more so as a result of the different

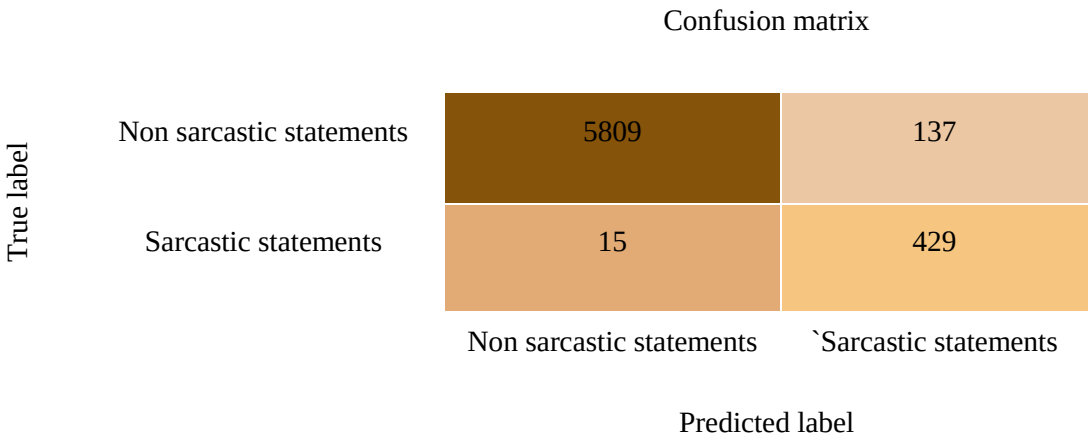
parameters in the principle of the performance evaluation measures. Table 8 depicts the results of the confusion matrix for the test using the driverless car dataset.

Table 8 Confusion matrix for test data using the driverless car dataset.



On the other hand, the extensive twitter car dataset generated a precision of 0.977, Recall of 0.997, and F1 score of 0.987 after data testing. Table 9 depicts the results of the confusion matrix for the test using the extensive twitter dataset.

Table 9 Confusion matrix for test data using the extensive twitter dataset.



Accuracy is determined by the number of test predictions over the length of the total test dataset. On the other hand, the test loss which considers the average test lost is the mean of the test losses.

The result of our setup shows a test loss of 0.309 and an accuracy of 0.905 using the extensive twitter dataset. That of the driverless car dataset shows a test loss of 0.136 and a test accuracy of 0.757. The figures are rounded to 3dp.

Again, it should be noted that these results depends on the size of the datasets and its composition. We are getting lesser test loss for driverless dataset because the dataset size is small than the extensive twitter dataset and therefore few loss will be attained. More so, we are obtaining higher test accuracy for the extensive twitter dataset comparably to the driverless dataset because the dataset size for the extensive twitter dataset is far greater than the driverless car dataset and hence, a higher probability of obtaining more sarcastic sentiments in the tweets.

The results provide an accurate, efficient, and reliable prediction based on the confusion matrix derived.

CHAPTER 5

CONCLUSION

5.1. Summary and Conclusion

Our study concentrates on developing a framework that will help in the identification of sarcastic sentiment in an opinion poll when undertaking sentiment analysis. This would assist to investigate the significant impact of sarcastic statements in opinion polls. Twitter datasets were used for this study. The study consists of five chapters and an extensive discussion has been done in each of the chapters. Sentiment analysis has been defined as the expression of opinions and attitudes by social media and internet users towards a specific topic or subject. Sarcasm detection during sentiment analysis is not an easy task and hence few works have been done on the subject matter. It has been viewed as consuming time and effort. The study undertakes a review and an in-depth analysis of sarcasm in sentiment analysis over the past 5 years and then develop a system that helps in the identification of a sarcastic statement when undertaking sentiment analysis.

The study brings about the development of a framework that enables us to classify sentiments as sarcastic or non-sarcastic, determine the level of predicted accuracy and loss and investigate the significant impact of the sarcastic sentiments in an opinion poll. The significant impact can be known because as we get to know the number of sarcastic statements, it would enable us to easily identify the impact they have on the holistic opinion poll. Theoretically and empirically, the effect of sarcastic statements in an opinion poll has been proven and a system has been developed that classifies sentiments into sarcastic and non-sarcastic. Consequently, we will be able to determine the entire level of sarcasm as high, low, or medium based on the number of detected sarcastic sentiments to the total comments in an opinion poll.

A framework has been introduced that comprises of three operators namely Cluster+Expert Judgement, Train & Validate and Classify & predict; that facilitates sentiment classification. Dataset used was a driverless car data which talked about the emergence of driverless cars being powered by Google which we applied X-means clustering algorithm (due to its repetitive nature to form partition to achieve the Bayesian Information Criterion) with an expert judgment that is based on intuition to label, and a broad-spectrum twitter dataset consisting of both sarcastic and non-sarcastic tweets.

The study considered LSTM for the implementation of our framework. The study of (Kumar et al., 2020) and (Son et al., 2019), employed multi-head Attention-based Bidirectional LSTM and sAtt-BLSTM respectively. However, their experiment reveals that their methodology is expensive and much complicated. We therefore employed a cheap, simple, efficient and straight forward approach by employing purely LSTM to produce a framework that helps in the determination of sarcastic statements when undertaking sentiment analysis. BLSTM would consequently lead to higher processing costs and complexities as compared to LSTM based on the experiment results of (Son et al., 2019). Considering their SemEval dataset, LSTM performs better than BLSTM as far as precision is concerned. The precision for LSTM was 86.78 whilst that of BLSTM was 81.61.This indicates that LSTM can perform better depending on the type of dataset and did perform well when we passed our datasets through our model.

We considered LSTM for our studies because it is effective and efficient in language processing. LSTM will make a prediction of sarcasm easy as compared to conventional lexicon-based and machine learning approaches. The employment of LSTM will extensively help to ensure better performance. There was an implementation of LSTM that could detect sarcastic comments. Using LSTM rather than a strict feedforward network is more accurate since we can include a sequence of

words information. In our descriptive implementation using LSTM, we started with loading and visualizing the data. There was data pre-processing which involved encoding the words, removing outliers, and padding sequences. A decision is made on how to split training, validation, and test data. We proceeded with creating data loaders with full batching. The network was defined using PyTorch and the layers were an embedding layer that converts tokens into embedding of a hidden state size, specific size, the number of layers which defines LSTM, a fully-connected output layer that maps the LSTM layer outputs to the desired output size and a sigmoid activation that returns only the last sigmoid output as the output of the network. We then instantiated the network by the creation of hyperparameters.

In training our data it was advantageous to use a new kind of cross-entropy loss, which was designed to work with a single sigmoid output. BCELoss, or Binary Cross-Entropy Loss, applies cross-entropy loss to a single value between 0 and 1. The hyperparameters where defined. Two ways were proposed for the testing which was testing data performance and inference on user-generated data. A confusion matrix was derived and values for precision, recall, f-score, and accuracy for the different datasets and stages were recorded accordingly. The results of our experiment are shown in our results and discussion section of our work. The lost statistics were obtained through the validation loss which is derived by the creation of new variables for hidden state or back prop through the entire training history and undergoing the net training to print each Epoch, Step, Loss, and validation loss. Using our driverless dataset, a test loss of 0.136 and a test accuracy of 0.757 was obtained. More so, a test loss of 0.309 and a test accuracy of 0.905 was obtained using our extensive twitter dataset. We could then do test conversion to tensor and pass it into our model. Here we obtained torch. Size ([1, 14]) as feature tensor size. So basically, a predict function accepted as parameters test review, sequence length, and net. So with our reviews,

we defined a sequence length of 141 and pass it through our predict function. Our test result for the sarcastic statement was a prediction value pre-rounding 0.998875 using a sarcastic tweet. And that of the non-sarcastic was a prediction value pre-rounding 0.000055. The results indicates that the prediction accuracy for sarcastic detection was very high for the tweet considered. On the other hand, a very slight loss was obtained. This could be confirmed by an expect judgment. The results therefore provide an accurate, efficient, and reliable prediction.

5.2. Threats to Validity

5.2.1 External Validity

This study took into consideration two different independent twitter datasets. The datasets were convenient samples and do not represent commonly every datasets since some datasets may contain images and other emoji. Outcomes from this study therefore might not be boldly widespread beyond other datasets.

5.2.2 Internal Validity

Threats to internal validity consist of factors that relate to the selection of the independent variables used for setting up the predictive models that could impact the results of this study. This study chose independent variables that are known to be existing before the starting of the project.

5.2.3 Constructive Validity

This study considered LSTM for building the prediction model when sampling. The model was considered as a result of the fact that it has been revealed to improve prediction accuracy and hence can be considered as an effective and reliable model.

5.2.4. Conclusion Validity

The models employed in the study were programmed and hence can result in automation biasness. When automating a process, it involves sequences of assumptions made which can end in biasness. Nonetheless, the assumptions made in this study are based on prior knowledge from already specified previous studies in building data mining and prediction models (Kumar et al.,2020 ; Son et al.,2019). The validity of the results therefore can be trusted.

5.3 Future Work

In a future study, further theoretical and empirical studies will be conducted on the determination and differentiation of some images and emoji as sarcastic or otherwise. More so, there should be a differentiation of sarcastic from an ironic statement. There could be an implementation using an unsupervised approach by other clustering algorithms and applying expert judgment algorithm on all raw data collected. Sarcasm has become very predominant in Ghanaian society. Therefore a future study could also consider the identification of sarcasm in Akan when undertaking sentiment analysis in Ghanaian Akan language

REFERENCES

Adri, J. (2016). A Sentiment Analysis System of Spanish Tweets and Its Application in Colombia 2014 Presidential Election. https://doi.org/10.1109/BDCloud-SocialCom-SustainCom.2016.47

Agarwal, B., Mittal, N., Bansal, P., & Garg, S. (2015). Sentiment Analysis Using Common-Sense and Context Information. *Computational Intelligence and Neuroscience*, *2015*, 1–9. https://doi.org/10.1155/2015/715730

Akhoundzade, R., & Devin, K. H. (2019). Persian sentiment lexicon expansion using unsupervised learning methods. *2019 9th International Conference on Computer and Knowledge Engineering, ICCKE 2019*, (Iccke), 461–465. https://doi.org/10.1109/ICCKE48569.2019.8964692

Alayba, A. M., Palade, V., England, M., & Iqbal, R. (2020). Arabic Language Sentiment Analysis on Health Services, 114–118.

Ayata, D., Saraclar, M., & Ozgur, A. (2017). Uzun-K Õ sa Süreli Bellek Yinelemeli A ÷ lar ile Politik Yönelimlerin / Duygular Õ n Twitter üzerinden Tahminlenmesi Political Opinion / Sentiment Prediction via Long Short Term Memory Recurrent Neural Networks on Twitter. *2017 25th Signal Processing and Communications Applications Conference (SIU)*, 1–4. https://doi.org/10.1109/SIU.2017.7960733

Baktha, K., Tripathy, B. K., Member, S., & Rnn, A. V. (2017). Investigation of Recurrent Neural Networks in the field of Sentiment Analysis, 2047–2050.

Bhan, N., & D'Silva, M. (2018). Sarcasmometer using sentiment analysis and topic modeling. *International Conference on Advances in Computing, Communication and Control 2017, ICAC3 2017*, *2018-Janua*, 1–6. https://doi.org/10.1109/ICAC3.2017.8318782

Bharti, D. K., Pradhan, R., Babu, K. S., & Jena, S. K. (2017). Sarcastic Sentiment Detection Based on Types of Sarcasm Occurring in Twitter Data, (August). https://doi.org/10.4018/IJSWIS.2017100105

Bharti, S. K., Babu, K. S., & Jena, S. K. (2015a). Parsing-based sarcasm sentiment recognition in Twitter data. *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2015*, 1373–1380. https://doi.org/10.1145/2808797.2808910

Bharti, S. K., Babu, K. S., & Jena, S. K. (2015b). Parsing-based sarcasm sentiment recognition in Twitter data. *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2015*, 1373–1380. https://doi.org/10.1145/2808797.2808910

Bharti, S. K., Naidu, R., & Babu, K. S. (2018). Hyperbolic Feature-based Sarcasm Detection in Tweets: A Machine Learning Approach. *2017 14th IEEE India Council International Conference, INDICON 2017*. https://doi.org/10.1109/INDICON.2017.8487712

Bouazizi, M., & Ohtsuki, T. (2015). Opinion Mining in Twitter How to Make Use of Sarcasm to Enhance Sentiment Analysis. *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 1594–1597. https://doi.org/10.1145/2808797.2809350

Chaudhari, P., & Chandankhede, C. (2018). Literature survey of sarcasm detection. *Proceedings of the 2017 International Conference on Wireless Communications, Signal Processing and Networking, WiSPNET 2017*, *2018-Janua*, 2041–2046. https://doi.org/10.1109/WiSPNET.2017.8300120

Dharmavarapu, B. D., & Bayana, J. (2019). Sarcasm Detection in Twitter using Sentiment Analysis, (1), 642–644.

El-jawad, M. H. A., & Hodhod, R. (2017). Sentiment Analysis of Social Media Networks Using Machine Learning. *2018 14th International Computer Engineering Conference (ICENCO)*, 174–176.

Goularas, D., & Kamis, S. (2019). Evaluation of Deep Learning Techniques in Sentiment Analysis from Twitter Data. *2019 International Conference on Deep Learning and Machine Learning in Emerging Applications (Deep-ML)*, 12–17. https://doi.org/10.1109/Deep-ML.2019.00011

Harvinder, M., & Kaur, J. (2015). Sentiment Analysis from Social Media in Crisis Situations. *International Conference on Computing, Communication & Automation*, 251–256. https://doi.org/10.1109/CCAA.2015.7148383

Hevner, A. R. (2007). A Three Cycle View of Design Science Research A Three Cycle View of Design Science Research, *19*(2).

Hiai, S., & Shimada, K. (2016). A sarcasm extraction method based on patterns of evaluation expressions. *Proceedings - 2016 5th IIAI International Congress on Advanced Applied Informatics, IIAI-AAI 2016*, 31–36. https://doi.org/10.1109/IIAI-AAI.2016.198

Jain, T., Agrawal, N., Goyal, G., & Aggrawal, N. (2018). Sarcasm detection of tweets: A comparative study. *2017 10th International Conference on Contemporary Computing, IC3 2017*, *2018-Janua*(August), 1–6. https://doi.org/10.1109/IC3.2017.8284317

Khasawneh, R. T., Wahsheh, H. A., Arabia, S., Aismadi, I. M., & Arabia, S. (2013). Sentiment Analysis of Arabic Social Media Content : A Comparative Study, 101–106. https://doi.org/10.1109/ICITST.2013.6750171

Khullar, H., & Singh, A. (2019). A Proposed Approach for Sentiment Analysis and Sarcasm Detection on Textual Data, (1), 3387–3391.

Kumar, A., Narapareddy, V. T., Srikanth, V. A., Malapati, A., Bhanu, L., & Neti, M. (2020). Sarcasm Detection Using Multi-Head Attention Based Bidirectional LSTM. *IEEE Access*, *8*, 6388–6397. https://doi.org/10.1109/ACCESS.2019.2963630

Li, L., Zhao, Y., Jiang, D., Zhang, Y., Wang, F., Gonzalez, I., … Sahli, H. (2013). Hybrid Deep Neural Network--Hidden Markov Model (DNN-HMM) Based Speech Emotion Recognition. In *Proceedings of the 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction* (pp. 312–317). https://doi.org/10.1109/ACII.2013.58

Losada, J. C., & Benito, R. M. (2018). Recurrent Patterns of User Behavior in Different Electoral Campaigns : A Twitter Analysis of the Spanish General Elections of 2015 and 2016, *2018*.

Lunando, E., & Purwarianti, A. (2013). Indonesian social media sentiment analysis with sarcasm detection. *2013 International Conference on Advanced Computer Science and Information Systems, ICACSIS 2013*, 195–198. https://doi.org/10.1109/ICACSIS.2013.6761575

Majumder, N., Poria, S., Peng, H., Chhaya, N., Cambria, E., & Gelbukh, A. (2019). Sentiment and Sarcasm Classification with Multitask Learning. *IEEE Intelligent Systems*, *34*(3), 38–43. https://doi.org/10.1109/MIS.2019.2904691

Manohar, M. Y., & Kulkarni, P. (2018). Improvement sarcasm analysis using NLP and corpus based approach. *Proceedings of the 2017 International Conference on Intelligent Computing and Control Systems, ICICCS 2017*, *2018-Janua*, 618–622. https://doi.org/10.1109/ICCONS.2017.8250536

Mensah, S., Keung, J., Bennin, K. E., & Bosu, M. F. (2016). Multi-Objective Optimization for Software Testing Effort Estimation. In *Proceedings of the 28th International Conference on Software Engineering and Knowledge Engineering (SEKE)* (pp. 527–530). San Francisco Bay, California, USA: SEKE. https://doi.org/10.18293/SEKE2016-017

Mensah, S., Keung, J., Svajlenko, J., Ebo, K., & Mi, Q. (2018). The Journal of Systems and Software On the value of a prioritization scheme for resolving Self-admitted technical debt. *The Journal of Systems & Software*, *135*, 37–54. https://doi.org/10.1016/j.jss.2017.09.026

Munandar, D., Arisal, A., Riswantini, D., Rozie, A. F., & Description, A. D. (2018). Text Classification for Sentiment Prediction of Social Media Dataset using Multichannel Convolution Neural Network. *2018 International Conference on Computer, Control, Informatics and Its Applications (IC3INA)*, 104–109.

Paredes-valverde, M. A., Colomo-palacios, R., Salas-zárate, M. P., & Valencia-garcía, R. (2017). Sentiment Analysis in Spanish for Improvement of Products and Services : A Deep Learning Approach, *2017*.

Park, J. H., Sung, Y., Sharma, P. K., Jeong, Y.-S., & Yi, G. (2017). Novel assessment method for accessing private data in social network security services. *The Journal of Supercomputing*, *73*(7), 3307–3325. https://doi.org/10.1007/s11227-017-2018-6

Parmar, K., Limbasiya, N., & Dhamecha, M. (2018). Feature based Composite Approach for Sarcasm Detection using MapReduce. *Proceedings of the 2nd International Conference on Computing Methodologies and Communication, ICCMC 2018*, (Iccmc), 587–591. https://doi.org/10.1109/ICCMC.2018.8488096

Porwal, S., Ostwal, G., Phadtare, A., Pandey, M., & Marathe, M. V. (2019). Sarcasm Detection Using Recurrent Neural Network. *Proceedings of the 2nd International Conference on Intelligent Computing and Control Systems, ICICCS 2018*, (Iciccs), 746–748. https://doi.org/10.1109/ICCONS.2018.8663147

Prasad, A. G., Sanjana, S., Bhat, S. M., & Harish, B. S. (2017). Sentiment analysis for sarcasm detection on streaming short text data. *2017 2nd International Conference on Knowledge Engineering and Applications, ICKEA 2017*, *2017-Janua*(2009), 1–5.

https://doi.org/10.1109/ICKEA.2017.8169892

Raghav, S., & Kumar, E. (2018). Review of automatic sarcasm detection. *2nd International Conference on Telecommunication and Networks, TEL-NET 2017, 2018-Janua*, 1–6. https://doi.org/10.1109/TEL-NET.2017.8343562

Razali, M. S., Halin, A. A., Norowi, N. M., & Doraisamy, S. C. (2018). The importance of multimodality in sarcasm detection for sentiment analysis. *IEEE Student Conference on Research and Development: Inspiring Technology for Humanity, SCOReD 2017 - Proceedings, 2018-Janua*, 56–60. https://doi.org/10.1109/SCORED.2017.8305421

Rendalkar, S., & Chandankhede, C. (2018a). Sarcasm Detection of Online Comments Using Emotion Detection. *Proceedings of the International Conference on Inventive Research in Computing Applications, ICIRCA 2018*, (Icirca), 1244–1249. https://doi.org/10.1109/ICIRCA.2018.8597368

Rendalkar, S., & Chandankhede, C. (2018b). Sarcasm Detection of Online Comments Using Emotion Detection. *Proceedings of the International Conference on Inventive Research in Computing Applications, ICIRCA 2018*, (Icirca), 1244–1249. https://doi.org/10.1109/ICIRCA.2018.8597368

Romanowski, A. (2015). Sentiment Analysis of Twitter Data within Big Data Distributed Environment for Stock Prediction, *5*, 1349–1354. https://doi.org/10.15439/2015F230

Saeed, H. H. (2018). Overlapping Toxic Sentiment Classification using Deep Neural Architectures. *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, 1361–1366. https://doi.org/10.1109/ICDMW.2018.00193

Sanagar, S., & Gupta, D. (2020). Unsupervised Genre-Based Multidomain Sentiment Lexicon Learning Using Corpus-Generated Polarity Seed Words. *IEEE Access*, *8*, 1–1. https://doi.org/10.1109/access.2020.3005242

Serrano-Guerrero, J., Olivas, J. A., Romero, F. P., & Herrera-Viedma, E. (2015). Sentiment analysis: A review and comparative analysis of web services. *Information Sciences*, *311*, 18–38. https://doi.org/10.1016/j.ins.2015.03.040

Son, L. H., Kumar, A., Sangwan, S. R., Arora, A., Nayyar, A., & Abdel-Basset, M. (2019). Sarcasm detection using soft attention-based bidirectional long short-term memory model with convolution network. *IEEE Access*, *7*, 23319–23328. https://doi.org/10.1109/ACCESS.2019.2899260

Srikanth, G. U. (n.d.). Survey of Sentiment Analysis Using Deep Learning Techniques.

Suhaimin, M. S., Hanafi, M., Hijazi, A., Alfred, R., & Coenen, F. (2019). Modified framework for sarcasm detection and classification in sentiment analysis, *13*(3), 1175–1183. https://doi.org/10.11591/ijeecs.v13.i3.pp1175-1183

Teh, P. L., Boon, O. P., Chan, N. N., & Chuah, Y. K. (2018). A comparative study of the effectiveness of sentiment tools and human coding in sarcasm detection, (January 2019), 0–15. https://doi.org/10.1108/JSIT-12-2017-0120

Troussas, C., Krouska, A., & Virvou, M. (n.d.). Evaluation of Ensemble-based Sentiment

Classifiers for Twitter Data. *2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 1–6. https://doi.org/10.1109/IISA.2016.7785380

Wp, R., T, A. N. S., & T, C. S. S. (2017). Sentiment Analysis Using Multinomial Logistic Regression, 46–49.

Appendix A

Table 3 Selected articles using SLR

	Tracking code	Citation	Source
1	LT1	M. Bouazizi and T. Ohtsuki, "Opinion Mining in Twitter How to Make Use of Sarcasm to Enhance Sentiment Analysis," *2015 IEEE/ACM Int. Conf. Adv. Soc. Networks Anal. Min.*, pp. 1594–1597, 2015.	IEEE/ACM
2	LT2	D. K. Bharti, R. Pradhan, K. S. Babu, and S. K. Jena, "Sarcastic Sentiment Detection Based on Types of Sarcasm Occurring in Twitter Data," no. August, 2017.	Scopus
3	LT3	P. L. Teh, O. P. Boon, N. N. Chan, and Y. K. Chuah, "A comparative study of the effectiveness of sentiment tools and human coding in sarcasm detection," no. January 2019, pp. 0–15, 2018.	Scopus
4	LT4	M. Bouazizi and T. Otsuki, "A Pattern-Based Approach for Sarcasm Detection on Twitter," *IEEE*	IEEE

		Access, vol. 4, pp. 5477–5488, 2016.	
5	LT5	S. Porwal, G. Ostwal, A. Phadtare, M. Pandey, and M. V. Marathe, "Sarcasm Detection Using Recurrent Neural Network," *Proc. 2nd Int. Conf. Intell. Comput. Control Syst. ICICCS 2018*, no. Iciccs, pp. 746–748, 2019.	IEEE
6	LT6	S. Rendalkar and C. Chandankhede, "Sarcasm Detection of Online Comments Using Emotion Detection," *Proc. Int. Conf. Inven. Res. Comput. Appl. ICIRCA 2018*, no. Icirca, pp. 1244–1249, 2018.	IEEE
7	LT7	N. Bhan and M. D'Silva, "Sarcasmometer using sentiment analysis and topic modeling," *Int. Conf. Adv. Comput. Commun. Control 2017, ICAC3 2017*, vol. 2018-Janua, pp. 1–6, 2018.	IEEE
8	LT8	H. Khullar and A. Singh, "A Proposed Approach for Sentiment Analysis and Sarcasm Detection on Textual Data," no. 1, pp. 3387–3391, 2019.	Scopus
9	LT9	Y. Wang, K. Kim, B. Lee, and H. Y. Youn, "Word clustering based on	Springer

		POS feature for efficient twitter sentiment analysis," *Human-centric Comput. Inf. Sci.*, 2018.	
10	LT10	S. K. Bharti, K. S. Babu, and S. K. Jena, "Parsing-based sarcasm sentiment recognition in Twitter data," *Proc. 2015 IEEE/ACM Int. Conf. Adv. Soc. Networks Anal. Mining, ASONAM 2015*, pp. 1373–1380, 2015.	IEEE/ACM
11	LT11	A. G. Prasad, S. Sanjana, S. M. Bhat, and B. S. Harish, "Sentiment analysis for sarcasm detection on streaming short text data," *2017 2nd Int. Conf. Knowl. Eng. Appl. ICKEA 2017*, vol. 2017-Janua, no. 2009, pp. 1–5, 2017.	IEEE
12	LT12	A. G. Prasad, S. Sanjana, S. M. Bhat, and B. S. Harish, "Sentiment analysis for sarcasm detection on streaming short text data," *2017 2nd Int. Conf. Knowl. Eng. Appl. ICKEA 2017*, vol. 2017-Janua, no. 2009, pp. 1–5, 2017.	IEEE
13	LT13	M. S. Razali, A. A. Halin, N. M. Norowi, and S. C. Doraisamy, "The importance of multimodality in	IEEE/ACM

		sarcasm detection for sentiment analysis," *IEEE Student Conf. Res. Dev. Inspiring Technol. Humanit. SCOReD 2017 - Proc.*, vol. 2018-Janua, pp. 56–60, 2018.	
14	LT14	B. D. Dharmavarapu and J. Bayana, "Sarcasm Detection in Twitter using Sentiment A Sarcastic sentiment detection based on types of sarcasm occurring in twitter datanalysis," no. 1, pp. 642–644, 2019.	Scopus
15	LT15	M. S. Suhaimin, M. Hanafi, A. Hijazi, R. Alfred, and F. Coenen, "Modified framework for sarcasm detection and classification in sentiment analysis," vol. 13, no. 3, pp. 1175–1183, 2019	Scopus
16	LT16	N. Majumder, S. Poria, H. Peng, N. Chhaya, E. Cambria, and A. Gelbukh, "Sentiment and Sarcasm Classification with Multitask Learning," *IEEE Intell. Syst.*, vol. 34, no. 3, pp. 38–43, 2019.	IEEE

Appendix B: the results obtained using our extensive twitter dataset

Available online on:

https://colab.research.google.com/drive/100Hfb_Igyx_Z8M5Af7gZInPBXFHSpeR0#scrollTo=RrPOFIeJLCCA.

Appendix C: shows the result on our sampling considering the driverless dataset.

Sample input size: torch. Size ([10, 71])
Sample input:

```
tensor ([[   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0, 1673,   52,   23,
            10,   92,    6,   82,   98,   10,    4,  475,  195,    1,  652],
        [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,  535,   30,   61,  929,  218],
        [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,  332,
             1,   90,  471,   83,   14,  801,  259,   52,   84,   35,    9,  802,
           199,   62,  142,  218,  808,  245,   41,   54,  133,   38,  244],
        [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,   42,   16,  133,    1,   27,   44,  285,    5,  309],
        [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,   78,
             0,   21,  111,   19,    3,  188,   61,   12,   19,    8,    2,  188,
            43,    8,  306,  153, 1041,   58,    5,  101,   11,   37,  504],
        [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0, 1296,   83,    4,    2,
```

```
 1297,    5,   94,   50,  197,  642, 1408,  110,  262, 1300,  385],
[    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
     0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
     0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
     0,    0,    0,    0,    0,    0,    0,    0,    0,    0,  266,
     1,   72,  378,   67,   11,  642,  208,   41,   67,   28,    7,  643,
   147,   38,  114,  439,  644,  202,   41,   18,   61,   36,   48],
[    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
     0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
     0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
     0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
     0,    0,    2,   97,    2,   83,   60,  392,   51,    1,   88,   71,
    18,  121,  406,  456,  216,    2, 2433,   24,   44,   21,   69],
[    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
     0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
     0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
     0,    0,    0,    0,    0,    0,    0,    0,    0,    0,  346,
     1,   93,  491,   87,   15,  835,  270,   54,   88,   37,    9,  836,
   205,   60,  147,  226,  841,  249,   38,   49,  114,   38,  213],
[    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
     0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
     0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
     0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
     0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
     0,    0,    0,    0,    0,    0,   38,   24,   18,    7, 1089]])

Sample label size:   torch. Size ([10])
Sample label:
 tensor ([0, 0, 1, 1, 1, 1, 1, 0, 1, 0])
```

Appendix D: shows the result on our sampling considering the extensive or broad-spectrum dataset.

```
Sample input size:   torch.Size ([10, 71])
Sample input:
 tensor ([[    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
               0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
               0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
               0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
               0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
               0,    0,    0,    1,    1,    1,  130, 22819,    9,
            1868, 1675,    7,   23,  108, 2027,   18,  243,   42,   17,
              10],
          [    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
               0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
               0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
               0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
```

```
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,   375,    45, 10147,  2019,    10,   401,
   1298,   331, 10276, 15716, 15717, 15718,  8031, 15719,   592,  1489,
  37558],
[     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,   372,
   9757,   337, 17541, 13512,    87, 11040,  2244, 17543,  3319, 17544,
  17545],
[     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,   261,    19,  6469, 30503,     8,
   6470],
[     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,   955,  9574, 13450,
    426],
[     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,    10,   106,   756,   284,   756,  1170,   548,
     30,   661,  2474, 12438,   500,    52,     8,  1365,    11,    90,
   1002],
[     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     1,  1944,  1945,  1946,   102,  1676,    10,  1622,
      1],
[     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
      0,     6,   233,    18,     4,   331,    73,   331,   630,   498,
    392],
[     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
```

```
        0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
        0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
        0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
        0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
        0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
        0,     0,     0,     0,     0,     0,    60,     3,   736,  5763,
     5763],
[    0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
        0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
        0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
        0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
        0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
        0,     0,     0,     0,     0,     0,     0, 20885, 33180, 33181,
       17,    12,   364,   218,   420,   432,   364,  5773,  1116, 21164,
    26532]])
```

```
Sample label size:   torch.Size([10])
Sample label:
 tensor([0, 0, 1, 0, 0, 0, 1, 1, 0, 1])
```

Appendix E: shows the results obtained using our driverless car dataset

```
Epoch: 1/4... Step: 10... Loss: 0.308221... Val Loss: 0.091225
Epoch: 1/4... Step: 20... Loss: 0.003682... Val Loss: 0.102582
Epoch: 1/4... Step: 30... Loss: 0.027171... Val Loss: 0.090254
Epoch: 1/4... Step: 40... Loss: 0.046117... Val Loss: 0.098415
Epoch: 1/4... Step: 50... Loss: 0.753728... Val Loss: 0.090104
Epoch: 1/4... Step: 60... Loss: 0.351336... Val Loss: 0.088068
Epoch: 1/4... Step: 70... Loss: 0.028270... Val Loss: 0.085113
Epoch: 1/4... Step: 80... Loss: 0.024787... Val Loss: 0.081984
Epoch: 1/4... Step: 90... Loss: 0.016288... Val Loss: 0.081004
Epoch: 1/4... Step: 100... Loss: 0.027513... Val Loss: 0.079952
Epoch: 1/4... Step: 110... Loss: 0.253308... Val Loss: 0.077065
Epoch: 1/4... Step: 120... Loss: 0.329916... Val Loss: 0.076185
Epoch: 1/4... Step: 130... Loss: 0.295668... Val Loss: 0.074330
Epoch: 1/4... Step: 140... Loss: 0.019882... Val Loss: 0.072703
Epoch: 1/4... Step: 150... Loss: 0.016161... Val Loss: 0.067215
Epoch: 1/4... Step: 160... Loss: 0.005318... Val Loss: 0.068186
Epoch: 1/4... Step: 170... Loss: 0.002181... Val Loss: 0.078808
Epoch: 1/4... Step: 180... Loss: 0.034508... Val Loss: 0.080732
Epoch: 1/4... Step: 190... Loss: 0.008713... Val Loss: 0.051247
Epoch: 1/4... Step: 200... Loss: 0.847413... Val Loss: 0.050752
Epoch: 1/4... Step: 210... Loss: 0.165887... Val Loss: 0.042462
Epoch: 1/4... Step: 220... Loss: 0.032503... Val Loss: 0.033708
Epoch: 2/4... Step: 230... Loss: 0.006288... Val Loss: 0.022869
Epoch: 2/4... Step: 240... Loss: 0.212718... Val Loss: 0.018187
```

Epoch: 2/4... Step: 250... Loss: 0.009774... Val Loss: 0.014826
Epoch: 2/4... Step: 260... Loss: 0.002254... Val Loss: 0.007697
Epoch: 2/4... Step: 270... Loss: 0.018461... Val Loss: 0.003845
Epoch: 2/4... Step: 280... Loss: 0.000709... Val Loss: 0.002665
Epoch: 2/4... Step: 290... Loss: 0.000634... Val Loss: 0.002383
Epoch: 2/4... Step: 300... Loss: 0.059355... Val Loss: 0.001749
Epoch: 2/4... Step: 310... Loss: 0.000500... Val Loss: 0.001550
Epoch: 2/4... Step: 320... Loss: 0.001336... Val Loss: 0.002043
Epoch: 2/4... Step: 330... Loss: 0.000684... Val Loss: 0.001425
Epoch: 2/4... Step: 340... Loss: 0.000218... Val Loss: 0.000794
Epoch: 2/4... Step: 350... Loss: 0.000231... Val Loss: 0.000818
Epoch: 2/4... Step: 360... Loss: 0.000346... Val Loss: 0.000770
Epoch: 2/4... Step: 370... Loss: 0.000268... Val Loss: 0.000734
Epoch: 2/4... Step: 380... Loss: 0.000173... Val Loss: 0.000699
Epoch: 2/4... Step: 390... Loss: 0.000275... Val Loss: 0.000666
Epoch: 2/4... Step: 400... Loss: 0.000431... Val Loss: 0.000858
Epoch: 2/4... Step: 410... Loss: 0.000888... Val Loss: 0.001160
Epoch: 2/4... Step: 420... Loss: 0.000726... Val Loss: 0.001105
Epoch: 2/4... Step: 430... Loss: 0.000593... Val Loss: 0.000954
Epoch: 2/4... Step: 440... Loss: 0.001276... Val Loss: 0.000815
Epoch: 3/4... Step: 450... Loss: 0.000467... Val Loss: 0.000716
Epoch: 3/4... Step: 460... Loss: 0.000421... Val Loss: 0.000644
Epoch: 3/4... Step: 470... Loss: 0.000431... Val Loss: 0.000584
Epoch: 3/4... Step: 480... Loss: 0.001586... Val Loss: 0.000535
Epoch: 3/4... Step: 490... Loss: 0.002078... Val Loss: 0.000494
Epoch: 3/4... Step: 500... Loss: 0.000290... Val Loss: 0.000455
Epoch: 3/4... Step: 510... Loss: 0.000451... Val Loss: 0.000428
Epoch: 3/4... Step: 520... Loss: 0.000849... Val Loss: 0.000389
Epoch: 3/4... Step: 530... Loss: 0.000214... Val Loss: 0.000364
Epoch: 3/4... Step: 540... Loss: 0.000246... Val Loss: 0.000342
Epoch: 3/4... Step: 550... Loss: 0.000238... Val Loss: 0.000322
Epoch: 3/4... Step: 560... Loss: 0.000267... Val Loss: 0.000305
Epoch: 3/4... Step: 570... Loss: 0.000278... Val Loss: 0.000292
Epoch: 3/4... Step: 580... Loss: 0.000153... Val Loss: 0.000279
Epoch: 3/4... Step: 590... Loss: 0.000145... Val Loss: 0.000265
Epoch: 3/4... Step: 600... Loss: 0.000663... Val Loss: 0.000255
Epoch: 3/4... Step: 610... Loss: 0.000175... Val Loss: 0.000243
Epoch: 3/4... Step: 620... Loss: 0.000238... Val Loss: 0.000232
Epoch: 3/4... Step: 630... Loss: 0.000354... Val Loss: 0.000224
Epoch: 3/4... Step: 640... Loss: 0.000138... Val Loss: 0.000216
Epoch: 3/4... Step: 650... Loss: 0.000542... Val Loss: 0.000210
Epoch: 3/4... Step: 660... Loss: 0.000121... Val Loss: 0.000204
Epoch: 4/4... Step: 670... Loss: 0.000098... Val Loss: 0.000198
Epoch: 4/4... Step: 680... Loss: 0.000137... Val Loss: 0.000194
Epoch: 4/4... Step: 690... Loss: 0.000134... Val Loss: 0.000191
Epoch: 4/4... Step: 700... Loss: 0.000125... Val Loss: 0.000189

Epoch: 4/4... Step: 710... Loss: 0.000164... Val Loss: 0.000178
Epoch: 4/4... Step: 720... Loss: 0.000126... Val Loss: 0.000173
Epoch: 4/4... Step: 730... Loss: 0.000081... Val Loss: 0.000168
Epoch: 4/4... Step: 740... Loss: 0.000342... Val Loss: 0.000164
Epoch: 4/4... Step: 750... Loss: 0.000103... Val Loss: 0.000160
Epoch: 4/4... Step: 760... Loss: 0.000155... Val Loss: 0.000155
Epoch: 4/4... Step: 770... Loss: 0.000095... Val Loss: 0.000149
Epoch: 4/4... Step: 780... Loss: 0.000104... Val Loss: 0.000145
Epoch: 4/4... Step: 790... Loss: 0.000105... Val Loss: 0.000139
Epoch: 4/4... Step: 800... Loss: 0.000207... Val Loss: 0.000135
Epoch: 4/4... Step: 810... Loss: 0.000096... Val Loss: 0.000131
Epoch: 4/4... Step: 820... Loss: 0.000110... Val Loss: 0.000127
Epoch: 4/4... Step: 830... Loss: 0.000311... Val Loss: 0.000124
Epoch: 4/4... Step: 840... Loss: 0.000286... Val Loss: 0.000120
Epoch: 4/4... Step: 850... Loss: 0.000110... Val Loss: 0.000116
Epoch: 4/4... Step: 860... Loss: 0.000080... Val Loss: 0.000113
Epoch: 4/4... Step: 870... Loss: 0.000149... Val Loss: 0.000110
Epoch: 4/4... Step: 880... Loss: 0.000088... Val Loss: 0.000107
Epoch: 4/4... Step: 890... Loss: 0.000058... Val Loss: 0.000104

Appendix F: General Overview of Sarcasm Detection Operators Flowchart