SARCASM DETECTION USING DEEP LEARNING

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY (M.Tech)
IN
ARTIFICIAL INTELLIGENCE

Submitted by

Kiran Bari (2K21/AFI/27)

Under the supervision of

Mrs. Minni Jain

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi 110042

MAY, 2023

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, **Kiran Bari**, Roll No. 2K21/AFI/27 students of M.Tech (ARTIFICIAL INTELLIGENCE),hereby declare that the project Dissertation titled "**Sarcam Detection Using Deep Learning**" which is submitted by me to the Department of Computer Science & Engineering, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: Delhi Kiran Bari

Date: (2K21/AFI/27)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the Project Dissertation titled "**Sarcam Detection Using Deep Learning**" which is submitted by **Kiran Bari**, Roll No. **2K21/AFI/27**, **Department of Computer Science & Engineering** ,Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the students under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi **Mrs. Minni Jain**

Date: **Assistant Professor**

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

I wish to express our sincerest gratitude to **Mrs. Minni Jain**, Assistant Professor, Department of Computer Science and Engineering, Delhi Technological University, Delhi for his continuous guidance and mentorship that he provided us during the project. She showed us the path to achieve our targets by explaining all the tasks to be done and explained to us the importance of this project as well as its industrial relevance. She was always ready to help us and clear our doubts regarding any hurdles in this project. Without her constant support and motivation, this project would not have been successful.

Place: Delhi — Kiran Bari

Date: — 2K21/AFI/27

Abstract

Even for humans, it can be difficult to recognize sarcasm, which is an important part of communication. To attract readers' attention, sarcasm is frequently used in newspaper headlines. Although headlines usually include irony, readers frequently miss it, misinterpreting the news and spreading misinformation to friends, coworkers, and other people. As a result, it is more important than ever to have a system that can automatically and reliably recognize sarcasm. In order to build sarcasm detectors, we employ neural networks, and we investigate how a computer may learn sarcastic patterns. The sequences that are fed into our project might be ironic or not. From collections of news headlines, these sequences were created. Our classifiers are evaluated for accuracy. Our approach is effective at identifying the difference between remarks that are sarcastic and those that are not.

Sarcasm identification is a method for spotting expressions that mean the exact opposite of what they mean to say. Systems for sentiment analysis that rely on emotion recognition face a substantial barrier due to the metaphorical character of sarcasm. Natural language processing (NLP) is a highly specialised field that focuses on sarcasm identification rather than sentiment analysis across many domains. Sarcasm detection in online forums is one such application.

Though they differ slightly, sarcasm detection and sentiment analysis are closely linked. Sarcasm recognition is a specialised study topic in the science of NLP that aims to ascertain if a particular text or phrase is sardonic. In a recent study, we investigated and contrasted several methods for sarcasm detection.

This study aims to tackle the problem of analysing big datasets while preserving the effectiveness of Deep Learning models. We seek elegant answers that provide efficient computing without compromising effective analysis.

Contents

List of Figures

List of Abbreviation

ML	Machine Learning
DL	Deep Learning
NLP	Natural Language Processing
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
CNN	Convolutional Neural Network
BERT	Bidirectional Encoder Representations from Transformers

Chapter 1

INTRODUCTION

Sarcasm is never used in headlines or comments on websites. Naturally, we're kidding. This kind of sarcasm is typically easy for people to recognize in person, but not all instances of sarcasm in writing are. Computers struggle to discern between the two in any format. We want to know if a computer can recognize sarcasm in a statement based on its language. Does extra context matter or can a machine learn these principles on its own? We'll approach our project in a sophisticated manner. We want to use neural networks and machine learning to build a sarcasm detector.

1.1 Overview

Textual customer reviews are supplied, and a data mining aim is to categorise them according to their emotional content using sentiment analysis. Because it uses positive phrases to express negative feelings, sarcasm is an essential part of natural language processing. It's important to recognise sarcastic remarks in order to avoid them being taken literally. Such sarcastic expressions are difficult for both humans and machines to recognise. Sarcasm significantly reduces the efficacy of sentiment analysis models, especially when addressing the commonly present deceptive attitudes in sarcastic remarks.People use a variety of mediums, including text, emoticons, and images, to convey their ideas and feelings on the vast social networking platform. Organisations frequently use this plethora of data to learn more about how the general public feels about certain goods, films, and political issues. However, effectively classifying utterances into positive, negative, or neutral feelings is a substantial issue for the process of sentiment analysis. Sarcasm makes determining a statement's polarity more difficult and can result in misunderstandings. As a result, the identification of sarcasm has grown to be a challenging problem in the study of sentiment analysis. Understanding and recognising sarcasm depend heavily on context and expression. The difficulty of sarcasm detection rises as social media gets more and more prevalent. Microblogs and social media platforms are rife with sarcasm, and authors there create a tonne of material that can't all be manually checked for sarcasm. As a result, there is a need for the creation of software or algorithms that are particularly

made to recognise and locate instances of sarcasm in such writings. .[1]

1.2 Problem statement

Finding sentences that have a meaning different from their intended one is a method of sarcasm detection. For sentiment analysis algorithms that depend on emotion recognition, the metaphorical character of sarcasm presents a significant hurdle. Sentiment analysis often seeks to identify the speaker's or author's point of view, interpreting it as a unique perspective or tone within a remark.

1.3 Project Objective

People are sociable creatures by nature. Whether pleasant or unpleasant, we often communicate with and interact with one another in a number of ways. One of the numerous elements of human communication that can influence the other person in both favourable and unfavourable ways is sarcasm. The challenge of correctly identifying the communication as sarcastic or "non-sarcastic" by employing irony or mockery to express scorn is known as sarcasm identification. It's a challenging procedure since we can't see or hear the person whose comment our software is evaluating because we can't see their face. Humans, on the other hand, are able to recognize a text's satirical undertone and explain it. In order to prevent incorrectly interpreting sarcasm in literal remarks, it is crucial for Natural Language Processing models, whose accuracy and validity are commonly impacted by false sarcastic sentiments.[2] Therefore, it is essential to filter out erroneous data from data sources while working on this project. For instance, the phrase "So thrilled to have different varieties of eatabe to be on call for work so that we have our result the whole weekend!" conveys excitement. This is because the term is being used sarcastically, which is meant to express a depressing and repetitive mood. On social media, microblogging websites, and e-commerce websites, sarcasm is often utilized.

Chapter 2

LITERATURE REVIEW

The discipline of NLP (natural language processing) has a substantial hurdle when attempting to identify sarcasm. In terms of model accuracy, recent research on sarcasm detection has demonstrated that machine learning and deep learning algorithms perform better than conventional approaches. Deep neural networks have the ability to automatically learn crucial characteristics rather than depending on manually created features. Deep learning models have proven to be quite effective in a variety of NLP tasks, including phrase summarization, machine translation, and improving reading comprehension. These applications demonstrate how the attention mechanism helps deep learning models perform better overall.

In their study, Pawar et al.[3] created a Machine Learning (ML) method for detecting sarcasm using Twitter datasets. On the datasets, they used K-Fold cross-validation to train and assess their models. In the experiment, sarcasm was detected using algorithms like Random Forest, SVM, and KNN.

Sarcasm identification was done by Srivastava et al.[4] as part of their study on well-known social media websites like Facebook and Twitter. They unveiled an approach based on the potent language model Google BERT. This method showed that handling massive amounts of data was possible. The suggested BERT-based model demonstrated improved efficiency and produced more accurate results when compared to existing models like SVM, CNN, and Logistic Regression.

Sarcasm identification was carried out in Kumar et al.'s work [5] on a particular Twitter dataset gathered during the COVID-19 time frame. Since more people were using social media at the time, people were actively commenting on other people's posts with their opinions and feelings. By using sarcasm recognition, the researchers hoped to find any instances of negative emotion in these remarks. The authors suggested the Naive Bayes classifier, Linear SVM classifier, and Decision Tree as techniques. The Decision Tree model outperformed the other methods with a 90% accuracy rate, making it the best at sarcasm detection in the provided Twitter dataset.

Jena et al.'s research [6] established a C-Net model that sequentially extracts contextual information from utterances and categorizes them as sardonic or non-sarcastic.

In the study, the effectiveness of conventional Machine Learning methods was contrasted with that of more modern transformer models. The results revealed that the Contextual Network model outperformed the performance of the conventional strategies in terms of sarcasm detection and exhibited promising outcomes.

Twitter datasets gathered during the COVID-19 era, when a sizable number of people were actively utilising social media platforms, were used in the research by Bhat et al.[7]. The study's main issue was how frequently hateful remarks and ideas appeared on these sites. The primary goal of the study was to create methods and approaches to find hate speech in a efficiently manner finding instances in the gathered of datasets collected from twitter.

To improve the working mechanism of an LSTM model, the author of this study suggested the idea of knowledge distillation.[8] The basis for the proposed method is abot the model of Teacher and Student. The main goal of this strategy is to teach the student model what the teacher's model has learnt. Without any information transfer from the instructor model, the student model's initial accuracy was 75.4%. However, the accuracy of the student model was greatly improved to 82.6% by taking into account the information learned from the instructor model. This enhancement shows how knowledge distillation may improve the functionality of the LSTM model.

Using a variety of approaches, Ahmed et al.[9] carried out sarcasm detection on Twitter datasets.We have seen SVM and Multinomial Logistic Regression and also for Random Forest classifiers were among the suggested methods. A lemmatization method was used to tokenize and preprocess the tweets. These methods were used to perform sentiment analysis on the datasets, which allowed for the categorization of tweet polarity in individual groups of five.

Ahmet et al.[10] conducted a thorough examination of DL techniques for sentiment analysis over a range of assessment domains, such as coarse-grain, cross-domain, and fine-grain domains. The findings showed that CNN, GRU, LSTM, and attention processes are the most often used techniques in sentiment analysis. In connection with this, Pathak et al.[**?**] provide a summary of approaches and technique used for sentiment analysis in the field of language modelling. They compared several text representation methods, datasets, neural network topologies, and basic deep learning algorithms for sentiment analysis in their work. By examining several assessment domains and emphasising crucial methodology and tools, these works expand our understanding of sentiment analysis using deep learning techniques.

Sarcasm identification was carried out on Facebook datasets by Das et al.[11] using a supervised learning model that combines text, picture, and numeric data. The authors discussed the difficulty of sarcasm detection on picture datasets, pointing out the paucity of prior research in this field. The authors suggested taking into account context and polarity as two important factors while analysing textual data. The caption that goes

with each image was taken into consideration as a pertinent component in the case of image analysis [12]. This method sought to efficiently capture sarcasm in the datasets' textual and visual components.

Due to the expanding network of social media platforms that are connected to one another and the resultant increase in big data, the ability to identify sarcasm has become crucial in the digital world. More academics are concentrating on sentiment analysis as Natural Language Processing (NLP) has become a significant area in machine learning. In recent years, a great deal of research has been done on Twitter datasets to investigate different methods for improving classification and opinion mining jobs.Punctuation and stop words play a key role in sarcasm recognition, according to Kreuz and Caucci [13], who investigated the use of lexical characteristics for sarcasm detection. They also used unigram lexical extraction of features, a method for sarcasm detection Carvalho et al. [14] that was later modified. Lukin and Walker [15] introduced a new method that uses pattern-based approach and also n-gram extraction for sarcasm detection.

Sarcasm detection on a dataset collected from Twitter using pattern-based method was proposed by Bouzizi [16]. The categorization of tweets was done using four proposed sets of characteristics. Basak et al.'s [17] main focus was on automatically classifying abusive and humiliating tweets into different categories. They created a block shaming web application to stop attackers and using an SVM classifier as their primary categorization method.

A semi-supervised method to identify sarcasm using two different datasets collectd from Twitter and Amazon was proposed by Davidov et al.[18]. The preprocessing of these datasets included features including pattern extraction, pattern selection, feature selection and pattern matching. Using these processed datasets as a basis, the statements were annotated. The datasets were tested using the KNN cluster classification method and the SASI algorithm, with an F-score of 0.82 and a precision of 91.2%. Various metrics were used to compare and assess the performance of the two datasets.

For the purpose of sarcasm detection using neural networks, many designs have been proposed. The deep CNN model created by dos Santos and Gatti [14][19] is one such architecture. Zhang et al.'s [15][20] neural CRF model is another efficient design. These neural network-based models show how effective neural networks are in detecting sarcasm. These suggested designs perform better than existing models, demonstrating the potency of neural networks in this field.

Chapter 3

Background

The key procedures for doing sarcasm detection will be covered in this section.

3.1 Collecting Dataset

Manual Annotation: One method is to manually annotate a dataset by asking human annotators to categorise samples as sarcastic or non-sarcastic. Although this might be time- and resource-consuming, it offers exact labelling and control over the dataset quality.

Crowdsourcing: Labelled data may be gathered via systems for crowdsourcing such as Amazon Mechanical Turk. Workers might be given detailed instructions on how to spot and classify sarcastic passages in provided text samples. To guarantee correct annotations, this strategy could need stringent screening and quality control techniques.

Existing Datasets: There are publicly accessible datasets that have been produced in the past for studies on sarcasm recognition. Examples include the Reddit Sarcasm Dataset and the SARCASM dataset, which both feature sarcastic tweets. While these datasets can serve as an excellent place to start for training and assessment reasons, it is crucial thing to make sure they are appropriate and in line with the precise specifications of your work.

Social media mining: Because sarcasm is so common there, social media is a great place to look for examples of sarcasm. To retrieve pertinent postings or comments, you may make use of the APIs offered by websites like Twitter or Reddit. When obtaining data from these sites, keep in mind the privacy rules, data usage guidelines, and terms of service.

Domain-Specific Data: According to the application you're developing, you might need to gather information that is specialised to a given domain or subject. For example, you would need to obtain information from pertinent review sites if your research is focused on the use of sarcasm in customer evaluations.

3.2 Data Pre-processing

Following the collection of the datasets, pre-processing is carried out with the intention of removing noise from the raw data received in the preceding stage. Stop-word elimination, tokenization, stemming, and lemmatization are a few of the techniques used in this procedure, which are usually used in jobs of NLP to produce the data tokens.

3.3 Tokenization

Natural language processing (NLP)'s core tokenization technique includes dividing a text into smaller components known as tokens. Depending on the amount of granularity required, these tokens may be words, sentences, or subword units. Many NLP jobs start with tokenization, which makes it possible to perform additional analysis including sentiment analysis, part-of-speech tagging, and machine translation. There are several tokenization methods and libraries available, each with a distinct strategy and set of factors to take into account based on the demands of the work.

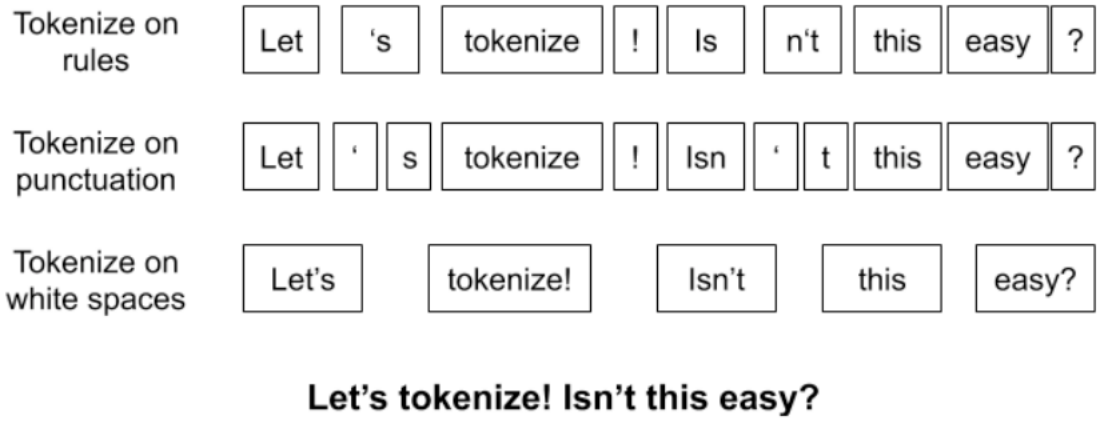

Figure 3.1: Tokenization

3.3.1 Stemming

Stemming is a method that reduces a word to its basic form, which might not be a true word but can still be used to determine the meaning of the word. To get a word's stem or root, one must take out any suffixes or prefixes. Although the resultant stem might not always be a correct word, it is nonetheless helpful in various individual NLP tasks including text mining, information retrieval and also sentiment analysis. Porter, Snowball, and Lancaster stemming are a few well-known stemming algorithms.

3.3.2 Lemmatization

Lemmatization is the process of stripping words of their inflections and returning them to their lemma, or dictionary form, by examining their vocabulary and grammar. The specific meaning of an expression in its context is ascertained using lemmatization algorithms, which then change the word to its root form.

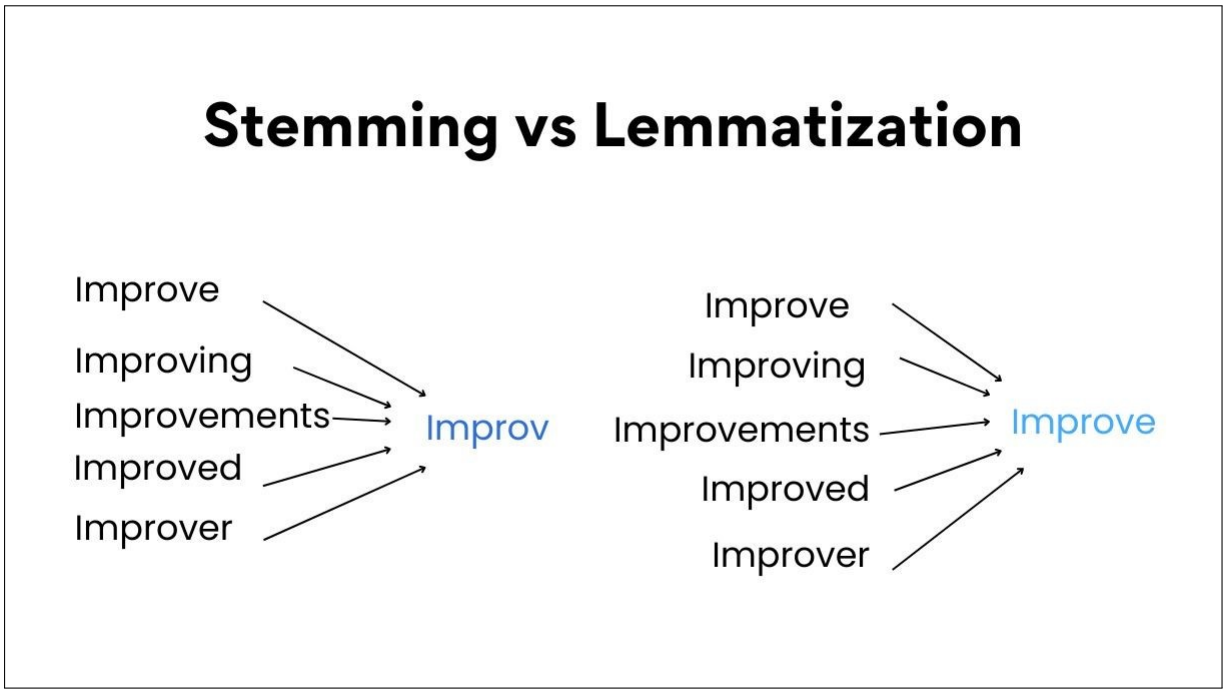

Figure 3.2: Stemming and Lemmatization

3.4 Features Extraction

The process of turning raw data, such as text, photos, or audio, into a collection of characteristic characteristics that capture the crucial details or qualities of the data is known as feature extraction. Feature extraction in the larger context of NLP entails transforming textual input into a numerical representation that machine learning algorithms can handle. In NLP, there are several methods for feature extraction, such as:

3.4.1 Bag-of-Words (BoW)

In this process, each unique word in sentence is represented along with how frequently it occurs in the document. It ignores word order and concentrates on the recurrence of certain words.

3.4.2 Term Frequency-Inverse Document Frequency (TF-IDF)

By using the word's frequency both inside and throughout the whole corpus, TF-IDF determines the significance of the word in a document. It gives the common terms in a document but not common in the corpus greater weights.

3.4.3 Word Embeddings

Word embeddings, like Word2Vec or GloVe, depict words as dense vectors in an ongoing semantic space. Based on their context, these vectors identify semantic connections and similarities between words.

3.4.4 N-grams

In a text, n-grams are a group of consecutive words. They can offer capabilities that are context-aware and they can record local word dependencies.

3.5 Feature Selection

The process of picking a subset of pertinent features from a broader pool of accessible features is known as feature selection. The objective is to decrease the data's dimensionality while preserving or improving the efficacy of the machine learning models. Natural language processing (NLP), among other machine learning tasks, requires careful feature selection. There are several methods for feature selection, such as:

3.5.1 Filter techniques

Independent of the selected learning algorithm, these methods assess the importance of features using statistical metrics like correlation or mutual information.

3.5.2 Wrapper techniques

In wrapper methods, a particular learning algorithm is used to assess subsets of attributes iteratively and choose the set that improves the performance of the model.

3.5.3 Embedded approaches

During training, embedded methods include feature selection in the learning algorithm itself. This includes approaches like regularisation and LASSO (Least Absolute Shrinkage and Selection Operator).

3.6 Classification Approaches

Machine learning techniques called classification approaches are used to categorise or classify input data based on its properties. Classification is frequently used in the context of NLP for tasks including sentiment analysis, text classification, spam detection, and topic classification.Identifying sarcasm in a text corpus requires the analysis of two classification problems. To tackle this classification challenge, previous research have used machine learning, deep learning, and hybrid techniques. Here are a few common NLP categorization techniques:

3.6.1 Naive Bayes

Naive Bayes is an independent between features probabilistic classifier. Based on the probability of a characteristic existing in a class, it determines the likelihood that an input belongs to that class. naïve Bayes frequently works well and is highly computationally effective in spite of its naïve premise. A data item's membership in a given category is determined using the Naive Bayes method. It may be used in text processing to classify words or phrases into predetermined labels. This method, which uses a simple word embedding technique, is based on the Bayes' rule and is regarded as naïve.

Consider a situation where the input is a text review of a movie, and there are two types (good and negative). The objective is to ascertain whether or not the review is favourable. Positive and negative words can be combined to form a bag of words. We may determine if a review is good or bad by counting how often these terms appear.

3.6.2 Support Vector Machines (SVM)

SVM is a popular supervised learning method that use for classification tasks. It locates an ideal hyperplane in a space of high-dimensional features that optimally divides various classes. SVMs are capable of managing both non-linear and linear decision limits and are efficient in tackling complicated classification issues. In higher dimensions, the SVM (Support Vector Machine) model can be seen as a hyperplane that successfully divides class labels. SVM aims to reduce error by repeatedly building the hyperplane. The dataset is segmented into groups in order to find the biggest margin hyperplane. This approach seeks to identify support vectors that separate N-dimensional spaces data points, where N is indicate the features number.

The following are important SVM terms:

Support vectors: These are the data points that are most closely spaced from the hyperplane. Hyperplane: A decision plane or space that divides several classes is referred to as a hyperplane. Margin: This term refers to the separation or space between two different classifications.

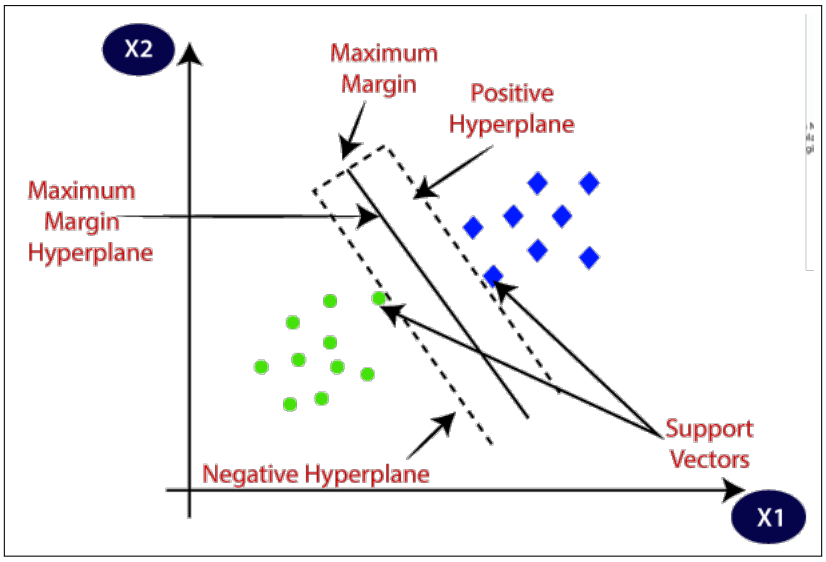

Figure 3.3: SVM

3.6.3 Logistic Regression

Using a logistic function, logistic regression predicts that an input likely to which class. With the use of methods like one-vs-rest or softmax regression, it may be expanded to tackle multi-class classification issues in addition to binary classification jobs.

3.6.4 Decision Trees

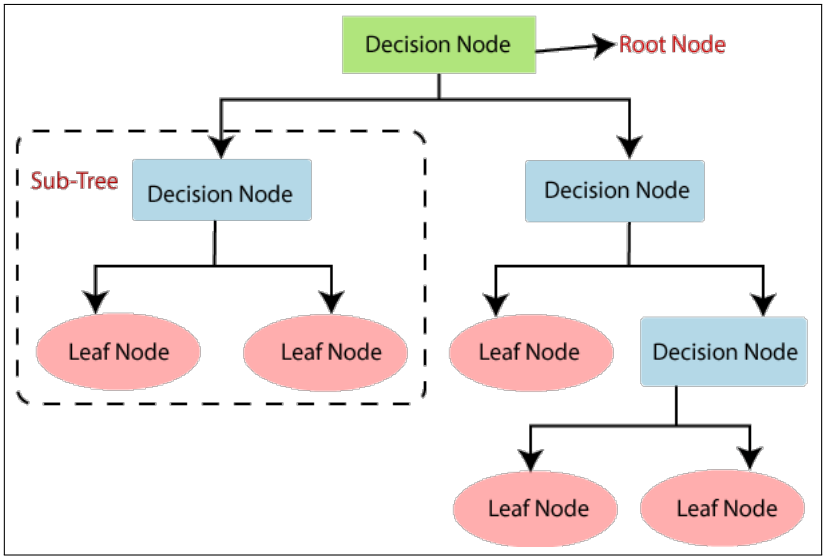

Figure 3.4: Decision Tree

Based on the values of features, decision trees are structures of hierarchy that form judgements. Each node in the leaf represents a class label, whereas every internal node indicates a feature. Both category and numerical characteristics may be handled by decision trees, and they are simple to read. The prediction and classification tasks that

the decision tree method is designed for. It has a tree-like structure, with the root nodes serving as the division of the population. Decision nodes are the nodes that are produced as a result of categorising the root nodes. Leaf nodes are nodes that can no longer be further divided.

3.6.5 Random Forest

Using a combination of several decision trees, Random Forest is an ensemble learning technique that produces predictions. By lowering overfitting and supplying more reliable and accurate classifications, it enhances performance. A machine learning technique called Random Forest may be used to solve a variety of problems, including regression and prediction. It makes use of supervised methods, which are useful for resolving issues involving several classes. A random forest method uses several different decision trees. The method use classifiers or bootstrapping aggregations to train the "forest". An ensemble meta-algorithm called bagging is used to optimise the model. The decision trees' estimates

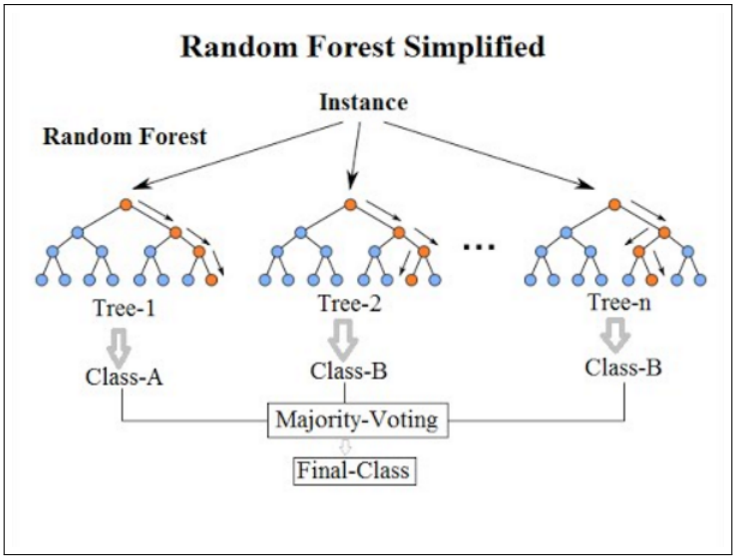

Figure 3.5: Randon Forest

are used by the random forest method to decide the outcome. To produce forecasts, it aggregates the results of several trees. As the number of trees rises, the accuracy of the forecasts rises as well. Some benefits of employing this method include:

High Accuracy: Random Forest has a propensity to produce forecasts with high accuracy. Effective Missing Data Handling: The method successfully manages missing data, minimising the effects on the outcomes. Random Forest aids in preventing overfitting problems, which can arise when a model is too sophisticated and unable to generalise to new data.

3.6.6 Neural Networks

Deep learning neural network models like CNN and RNN have demonstrated outstanding performance in NLP applications. They are able to successfully collect semantic and contextual information and learn complicated representations of text data. We will go through several deep learning techniques used to find both local and sustained positional abnormalities in the comments in the sections below.

Convolutional Neural Network (CNN) :

A well-liked deep learning technique for analysing structured grid-like data, such as photos and movies, is the convolutional neural network (CNN). It has three main layers, thats are convolutional layers and pooling layers and also fully connected layers.

CNNs use the convolutional principle, which involves applying filters to specific small areas of the input data to extract useful characteristics. These filters may detect different patterns and data structures and are taught throughout the training phase. By combining layers, the network becomes more computationally efficient by reducing the overall dimension into the feature maps. Finally, the characteristics of retrieved data are combined and predictions are based on them using fully connected layers.

In recognition of images tasks, object identification, as well as other computer vision applications, CNNs have proved quite effective. They have the ability to naturally learn hierarchy representations of visual input, which gives them the ability to accurately recognise complicated patterns and objects.

A particular kind of deep neural network called a convolutional neural network (CNN) is frequently employed in deep learning for the analysis of visual images. CNNs are renowned for their capacity to detect complex patterns in pictures, but because of their "total connectivity" nature, they can also be vulnerable to overfitting. Different regularisation methods, including penalising particular characteristics during training or removing connections, might be used to overcome this.

CNNs take use of the hierarchical structure seen in visual data, unlike conventional multilayer perceptrons. To catch patterns of increasing complexity, they use filters, which are shorter, easier sequences. By using a hierarchical learning strategy, CNNs may successfully learn and represent visual characteristics while limiting connection and complexity.

CNNs establish a compromise between gathering important data and avoiding overfitting by using the hierarchical structure and applying simpler filters. They are therefore excellent at identifying patterns and objects in visual input, making them suitable for jobs involving picture analysis.

CNN Layers : There are essentially three levels to CNN.

Convolution Layer:

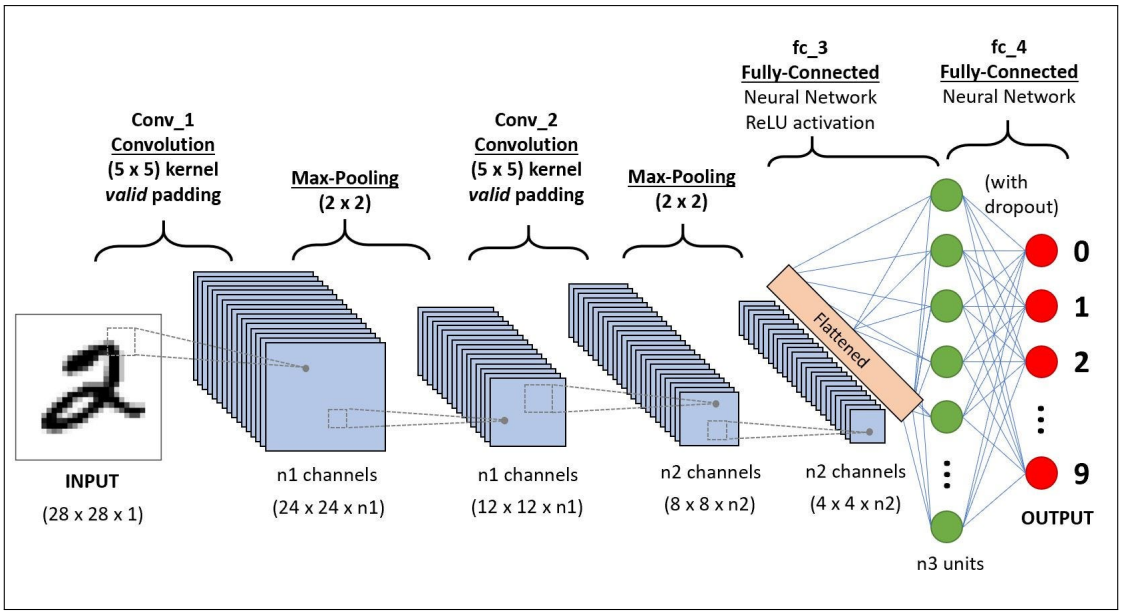

Figure 3.6: CNN

In a convolutional neural network (CNN), convolutional layers are essential for processing data. These layers apply a convolution process to the incoming data, simulating the way that brain neurons react to various inputs. Each convolutional neuron handles a certain chunk of the data processing and transmits the findings to the linked perceptron.

Fully linked feed-forward artificial neural networks are useful for categorizing data and training classifiers, but they could struggle with bigger inputs like high-resolution images. CNN really shines in this area. CNNs can effectively analyze and extract features from complicated visual data by using convolutional layers, which makes them more appropriate for jobs involving pictures or other high-dimensional input.

Pooling Layer:

It is typical for one to include the Pooling Layer after the Convolutional Layer. By lowering the number of dimensions of the feature maps created in the previous step, this layer helps to cut down on computing expenses. It does this by using separate operations on each feature map and fewer connections between layers. According to the mode chosen, many pooling processes occur. The greatest value found in each area of the map of features is chosen in Max Pooling. The mean of the components included within a certain area of the input is calculated using average pooling. Sum Pooling determines the total of the components in a given area. The Pooling Layer often serves as a link to the Convolutional Layer & the fully linked layer, making it easier to move between these two phases of the neural network design.

Fully Connected Layer:

Every neuron in each layer of fully connected layers, sometimes referred to as dense layers or multilayer perceptron neural networks (MLP), is coupled to every neuron in the layer below it. When doing classification or prediction tasks, the input data is generally flattened into a vector and sent through a fully connected layer.

A weight matrix is used to alter the input vector linearly inside every neuron of a fully connected layer. The input vector is multiplied by the weighted matrix in this transformation, and a bias component is also included. The output is then subjected to a non-linear activation function, that introduces non-linearity and allows the network to learn intricate correlations and patterns in the data.

In order for the network to simulate and capture non-linear interactions between the input and output, the non-linear activation function is used to add non-linearity into the system. The hyperbolic tangent function (tanh), rectified linear unit (ReLU) function, and the sigmoid function are often employed activation functions in fully linked layers.

Overall, fully connected layers of a neural network aid in the capture of complex patterns and relationships by performing linear transformations on the input data, followed by non-linear activation functions. As a result, the network may learn the weights of the connections between the neurons and use them to generate predictions.

Recurrent Neural Network (RNN):

An effective neural network type for processing sequential data is called a recurrent neural network (RNN). RNNs may remember and use information from earlier phases in the sequence because they have connections that create a directed cycle, unlike feedforward neural networks.

An RNN's essential characteristic is its capacity to sustain an internal state, or memory, which enables it to recognise relationships and patterns across time. The RNN receives an input at each step in the sequence, changes its hidden state depending on that input, and then receives another input at the next step. As a result, the network may utilise data from earlier actions to improve predictions or judgements about the future.

RNNs are often employed in a variety of sequential data applications, including time series analysis, speech recognition, and natural language processing. When handling activities that need for a grasp of context and temporal interdependence, they are highly successful.

Traditional RNNs, on the other hand, may have the "vanishing gradient" problem, where the gradients are progressively smaller as they spread over time, making it challenging to learn long-term dependencies. RNN variations such Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) have been developed to overcome this problem. These architectures provide unique methods that improve gradient flow and let the network identify longer-term relationships.

RNNs are strong models of neural networks that excel at processing sequential input

because they can access internal memory and recognise temporal relationships. They have made a substantial contribution to the development of several sequential information processing tasks.

RNNs prioritise the entire sequence rather than individual parts, making them particularly well-suited for jobs where the sequence is critical. A fully connected neural network with certain layers reconfigured into a loop can be thought of as an RNN. A non-linear activation function, matrix multiplication, and operations like concatenation or composition of inputs are frequently used in this loop. The important feature of RNNs is its internal memory, which enables prior inputs to affect predictions in the future. By taking into account the context that the words before them give, this memory helps the network to generate predictions in a phrase that are more correct.

The activation value of the previous time step, t-1, is simultaneously provided as an input with the current input at time t in the context for language processing, where each word in a sentence is handled as a separate input at a particular time t. By using the information of previous words to anticipate the meaning of following words, this method enables the RNN to capture relationships and patterns across the sequence.

RNNs may perform well in tasks like language modelling, machine translation, and voice recognition thanks to this configuration since good predictions depend on a grasp of the context and temporal interactions between elements. RNNs have shown to be efficient at capturing long-term dependencies and producing context-aware outputs by making use of their own internal storage and recurrent connections.

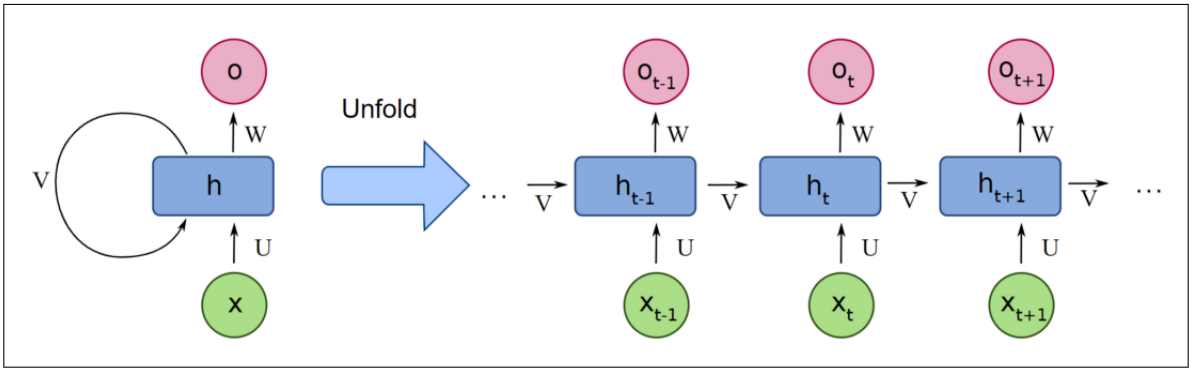

Figure 3.7: RNN

Long-Short Term Memory (LSTM) :

Long-Short Term Memory (LSTM), also referred to as RNN, is a form of RNN that is made to process sequential input and identify long-term dependencies. LSTM networks, as opposed to conventional RNNs, use a memory cell which can store as well as retrieve data over a lengthy period of time. By selectively storing or forgetting information, this memory cell enables LSTMs to interpret and anticipate sequences successfully. Time

series analysis, speech recognition, and natural language processing are just a few of the areas where LSTMs are being successfully used.

Recurrent neural networks of the LSTM variety are distinguished from more conventional recurrent neural networks by their higher memory performance. Similar to other kinds of neural networks, it is excellent at learning certain patterns and selectively keeps significant information while rejecting irrelevant input in each cell as it advances through the layers.

There are three primary gates in an LSTM.

FORGET Gate:- During the calculation of cell state, the Forget gate is essential in deciding whether data should be kept or discarded. The prior hidden state (ht-1) and the present cell state (xt) are inputs that are used. The Forget gate then analyses these inputs using a sigmoid function, which aids in determining which values should be ignored (closer to 0) and which should be utilised for modifying the cell state.

INPUT Gate:- The input gate controls the state of the cell and determines whether incoming data is relevant. It assists in identifying significant information and storing it in the memory, much as the aforementioned gate. The sigmoid and tanh functions are used to process the inputs ht-1 and xt, respectively. The tanh function maintains the network while reducing bias, whereas the sigmoid function aids in controlling the information flow.

OUTPUT Gate:- The next concealed state is decided by the Output gate, which acts as the last gate. The information flow is determined through the application of a sigmoid function to the parameters ht-1 and xt. The data to be kept in the hidden state is determined by the updated cell state after it has been processed through the tanh function, which is subsequently multiplied by the sigmoid function's output.

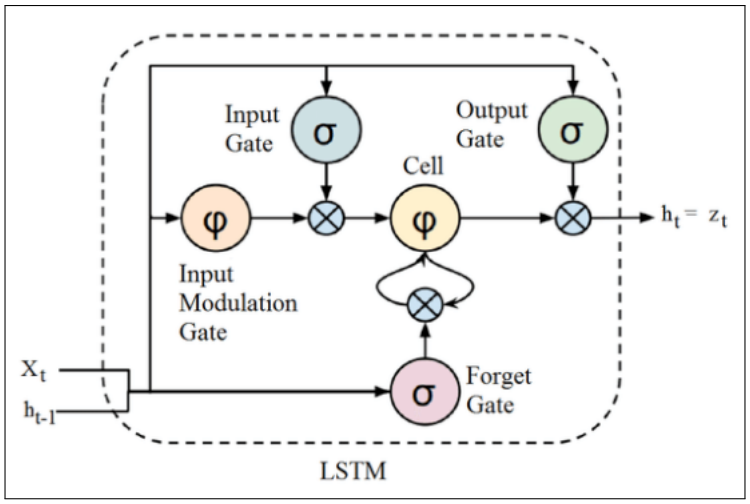

Figure 3.8: LSTM

Chapter 4

Data Analysis

The information is offered in json format, which may be processed to create a pandas data frame. The data set has columns for the piece of writing link, the headline itself, and an indicator indicating whether or not the title is sarcastic. It contains 26,709 headlines. An algorithm based on machine learning will be trained to correctly recognise sarcasm in headlines from newspapers. The organisation column, which denotes the source of the news piece, is used in place of the article link column in order to streamline the analysis. This data can provide light on how sardonic and non-sarcastic headlines are distributed throughout various organisations. Noting that there are no missing values in the dataset ensures that the information is full for analysis.

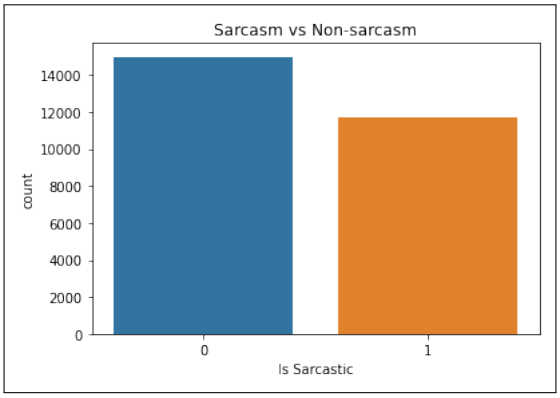

Figure 4.1: Dataset

Sarcastic headlines are designated as 1, while non-sarcastic headlines are designated as 0, respectively, in the dataset. The label column's average value of around 0.44 shows that the data is not strongly biassed towards one category and is appropriate for our research. It's also interesting to observe that none of the dataset's ironic headlines come from HuffPost; rather, they are all from The Onion.

Then, all of the headlines are tokenized using Python's Keras framework, and a dictionary is made. When all the news headlines are taken into account, the resultant dictionary comprises a total of 29,656 distinct terms. But because the maximum number of words in our model is 20,000 (randomly initialised), we exclude the final 9,656 words with low frequency. We proceed with our project after making these preparations, using

a bidirectional LSTM to analyse the embedded text in the news headlines.

Chapter 5

METHODOLOGY

The suggested method for sarcasm detection uses the RNN-LSTM model, which has four key components: A) Dataset B) Pre-Processing Dataset C) Words Embedding D) RNN-LSTM model.

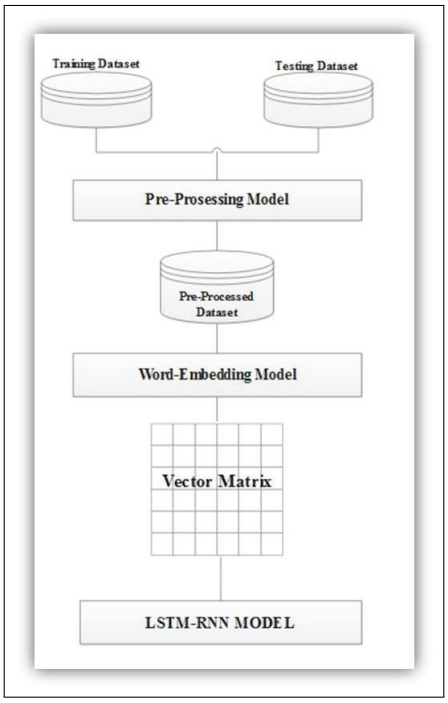

Figure 5.1: Architecture of the proposed system

Let's examine the architecture and key elements in more detail. Two distinct datasets, one to be trained and one for testing, were used to train the model. The training dataset underwent pre-processing operations to get it ready for training. During pre-processing, any newlines, punctuation, special symbols, redirection to links, white spaces and other factors that don't contribute to understanding the text's sarcasm, if any, were eliminated. After the pre-processing, we got a new pre-processed dataset. The dataset that are pre-processed is employed in the following stages. Word Embedding is done on the new clear dataset. Following the completion of label and integer encoding, a neural network (in this example, an RNN) is trained, and then words are converted to vectors. Addition-

ally, a vector matrix is used to store the word-to-vector conversion. The LSTM makes use of a vector matrix, and it stores its weights in a different model. At this moment, categorization is finished. The sentence's accuracy is then evaluated using the testing dataset.

5.1 Dataset

Several sarcastic and non-sarcastic News Headlines were mined to create the dataset, which was then properly trained using the specified model. The collection also offers factual information, true opinions, and conversational news headlines. The same process is used to produce a test dataset.

5.2 Pre-Processing Dataset

At first, we convert the data into a clear and necessary format and then feed it to the model. The raw News Headlines from these datasets are fed into preprocessing for feature extraction and clean-up. Several unnecessary things have been removed from the news headlines. The two main pre-processing elements are Tokenization and padding sequence. These News Headlines are used in the word embedding model when the text is transformed to a sequence. Transformed News Headlines are uniform in length and feature no padding.

5.3 Words Embedding

Word embedding is used for converting the text into numerical data because the neural networks can only assess numerical data and not language. Word embedding excels at labor-intensive tasks like processing and analysing vast quantities of text. Words are translated into real-number vectors using word embedding. A 128-em embedding size was used. The layers of the neural network are then fed the resulting vectors. Additionally, word embedding makes it easier for words to be connected grammatically, suggesting that two words could run into each other. Based on how many words are combined, the model tends to forecast the next word. Sharp is far more likely to be followed by a term like "Knife" or "Mind" for instance.

5.4 RNN-LSTM model

The RNN-LSTM module may move both forward and backward. Data is provided to the following node during forward travel so that it can get familiar with the previous node. For instance, "I am skilled in computer science, thus I'm pursuing a master's degree

in computer science." To ascertain the possibility of a "Computer Science" related word occurrence, the context of the first node of the good at Computer Science network may be passed on to the next node. Sarcasm identification requires an understanding of context, and one of the most crucial tasks—and one that a neural network excels at—is extracting characteristics. Neural networks may implicitly extract characteristics since they operate on an activation function, which causes their intensity to increase with each node. The first stage in creating a strong model is to train on a dataset that is rich in features, and the second is to make the most of each feature. LSTM cells can recall and forget information, hence it is useful to employ them in a model. Figure 2 displays a diagrammatic depiction of RNN and LSTM nodes. RNN uses the 'tanh' activation function to produce (ht-1) state from (Xt-1) state as further data is sent to the following node. An LSTM's gates are comparable to electrical gates in that they both store information and may act as a counter when arranged in a certain way. Similar to this, LSTMs analyse data by storing only the data that is necessary and discarding the rest.

We used the first layer embedding layers, second layer as a spatial dropout layer, and the third layer LSTM layers in our model.[21] As the no of layers affects in a neural network's accuracy, we chose a moderate no of layers to control both accuracy and computation. We used various python libraries to build the model such as TensorFlow, Keras etc.

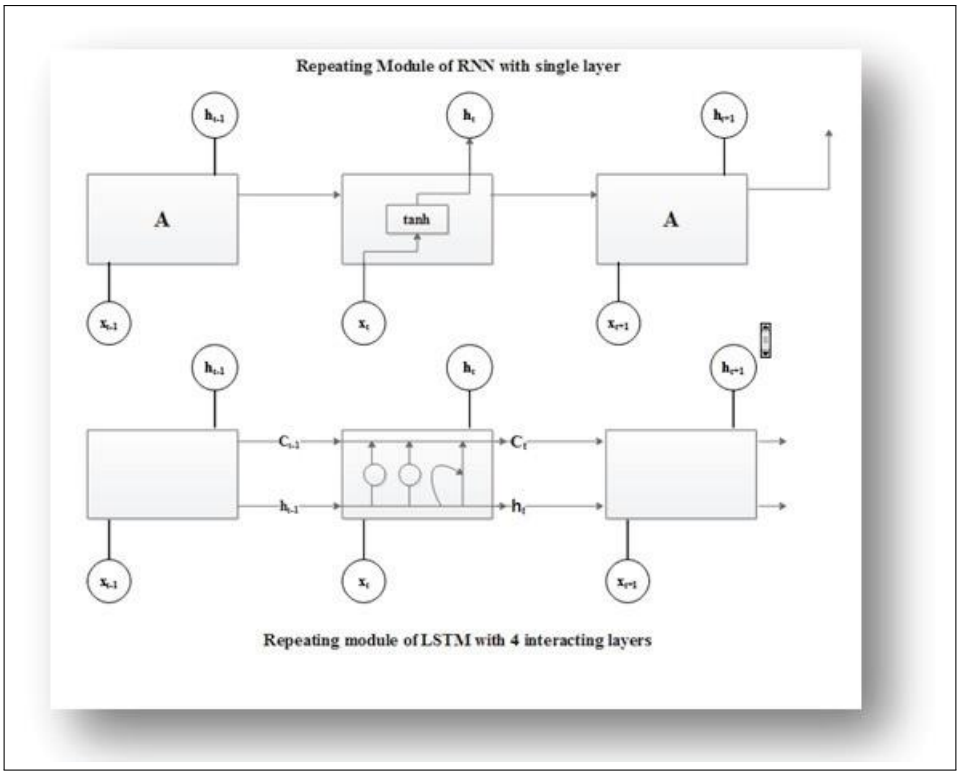

Figure 5.2: LSTM-RNN

Chapter 6

RESULTS

11,724 sarcastic headlines and 14,985 non-sarcastic headlines were used to train our algorithm. Despite the hardware limitations, the model performed well during training. Only after 25 epochs had passed, the model had attained 84 percent accuracy. As well as no

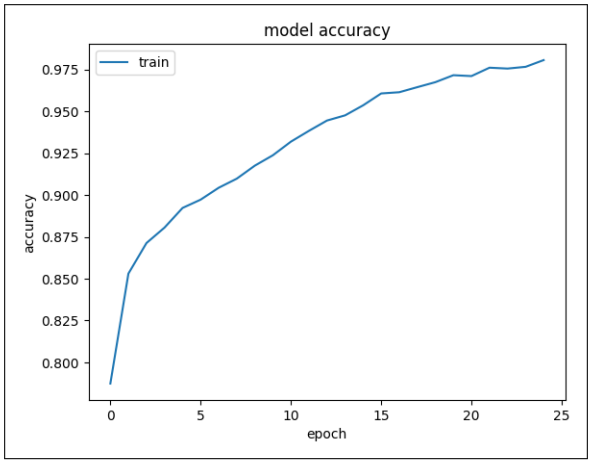

Figure 6.1: Epoch vs Accuracy

of epochs increase the accuracy is increased. Loss of model decreases when no of epochs

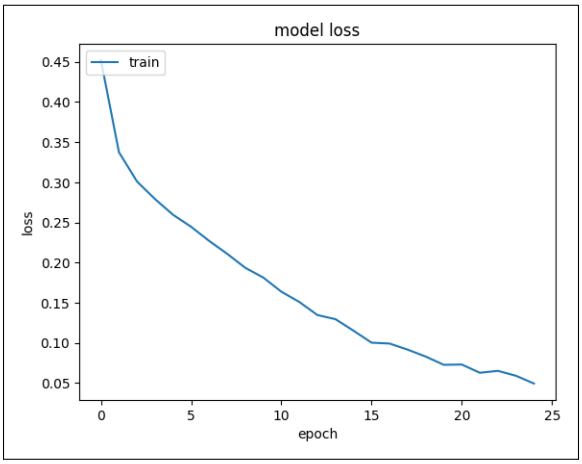

Figure 6.2: Epoch vs Loss

is increased.

On testing dataset of 26,709 News Headlines, Sarcasm accuracy 81.513 % , Non-Sarcastic accuracy 83.842 % It is an effective model that is unquestionably superior to existing algorithmic models for sarcasm detection, with accuracy exceeding 84 percent and fewer epochs.

Chapter 7

CONCLUSION AND FUTURE SCOPE

In our experiment, we were able to recognise sarcasm in a number of headlines from different parts of the world. With an accuracy of 85%, we were able to classify the headlines based on the phrases used. Furthermore, we recognise sarcasm in a specific news title. On our dataset, the RNN and LSTM performed quite well. Our system can learn to which is sarcastic and which non-sarcastic headlines without context, which is a challenge that many individuals find challenging.

Due to its contextual aspect, sarcasm can be difficult to understand and anticipate. Sarcasm is sometimes difficult to understand using traditional sentiment analysis techniques and algorithms, especially when working with big amounts of user-generated data like product evaluations. This may result in false information being reported, possibly harming businesses, and faking review data. The suggested model can act as a filter to solve this problem by separating sarcastic from non-sarcastic phrases. Then, to ensure more accurate findings, only the non-sarcastic phrases are transmitted to sentiment analysis software and instruments.

An easy method to use the sarcasm detecting model is by implementing it as an API. The API accepts statements from users and returns a response reflecting whether the statement qualifies as sarcastic or not. This API implementation makes the model easier to retrieve and more user-friendly for a variety of applications.

References

[1] Y.-D. Zhang, T. Senjyu, S. Chakchai, and A. Joshi, *Smart Trends in Computing and Communications*. Springer, 2020.

[2] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, "Deep learning–based text classification: a comprehensive review," *ACM computing surveys (CSUR)*, vol. 54, no. 3, pp. 1–40, 2021.

[3] N. Pawar and S. Bhingarkar, "Machine learning based sarcasm detection on twitter data," in *2020 5th international conference on communication and electronics systems (ICCES)*. IEEE, 2020, pp. 957–961.

[4] M. Shrivastava and S. Kumar, "A pragmatic and intelligent model for sarcasm detection in social media text," *Technology in Society*, vol. 64, p. 101489, 2021.

[5] R. Kumar and A. Bhat, "An analysis on sarcasm detection over twitter during covid-19," in *2021 2nd International Conference for Emerging Technology (INCET)*. IEEE, 2021, pp. 1–6.

[6] B. Jang, M. Kim, G. Harerimana, S.-u. Kang, and J. W. Kim, "Bi-lstm model to increase accuracy in text classification: Combining word2vec cnn and attention mechanism," *Applied Sciences*, vol. 10, no. 17, p. 5841, 2020.

[7] A. Isaac and A. Bhat, "A conceptual enhancement of lstm using knowledge distillation for hate speech detection," in *Smart Trends in Computing and Communications*, Y.-D. Zhang, T. Senjyu, C. So-In, and A. Joshi, Eds. Singapore: Springer Singapore, 2022, pp. 561–570.

[8] A. Garg, K. Aggarwal, M. Saxena, and A. Bhat, "Classifying medical histology images using computationally efficient cnns through distilling knowledge," in *Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2020, Volume 3*. Springer, 2021, pp. 713–721.

[9] M. Ahmed, M. Goel, R. Kumar, and A. Bhat, "Sentiment analysis on twitter using ordinal regression," in *2021 International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON)*. IEEE, 2021, pp. 1–4.

[10] A. Ahmet and T. Abdullah, "Recent trends and advances in deep learning-based sentiment analysis," *Deep learning-based approaches for sentiment analysis*, pp. 33–56, 2020.

[11] D. Das and A. J. Clark, "Sarcasm detection on facebook: A supervised learning approach," in *Proceedings of the 20th international conference on multimodal interaction: adjunct*, 2018, pp. 1–5.

[12] K. Parmar, N. Limbasiya, and M. Dhamecha, "Feature based composite approach for sarcasm detection using mapreduce," in *2018 second international conference on computing methodologies and communication (ICCMC)*. IEEE, 2018, pp. 587–591.

[13] R. Kreuz and G. Caucci, "Lexical influences on the perception of sarcasm," in *Proceedings of the Workshop on computational approaches to Figurative Language*, 2007, pp. 1–4.

[14] P. Carvalho, L. Sarmento, M. J. Silva, and E. De Oliveira, "Clues for detecting irony in user-generated contents: oh...!! it's" so easy";-," in *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, 2009, pp. 53–56.

[15] S. Lukin and M. Walker, "Really? well. apparently bootstrapping improves the performance of sarcasm and nastiness classifiers for online dialogue," *arXiv preprint arXiv:1708.08572*, 2017.

[16] M. Bouazizi and T. Ohtsuki, "Sarcasm detection in twitter:" all your products are incredibly amazing!!!"-are they really?" in *2015 IEEE global communications conference (GLOBECOM)*. IEEE, 2015, pp. 1–6.

[17] R. Basak, S. Sural, N. Ganguly, and S. K. Ghosh, "Online public shaming on twitter: Detection, analysis, and mitigation," *IEEE Transactions on computational social systems*, vol. 6, no. 2, pp. 208–220, 2019.

[18] D. Davidov, O. Tsur, and A. Rappoport, "Semi-supervised recognition of sarcasm in twitter and amazon," in *Proceedings of the fourteenth conference on computational natural language learning*, 2010, pp. 107–116.

[19] C. Dos Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts," in *Proceedings of COLING 2014, the 25th international conference on computational linguistics: technical papers*, 2014, pp. 69–78.

[20] M. Zhang, Y. Zhang, and D.-T. Vo, "Neural networks for open domain targeted sentiment," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 612–621.

[21] S. S. Salim, A. N. Ghanshyam, D. M. Ashok, D. B. Mazahir, and B. S. Thakare, "Deep lstm-rnn with word embedding for sarcasm detection on twitter," in *2020 international conference for emerging technology (INCET)*. IEEE, 2020, pp. 1–4.

PAPER NAME

kiran.pdf

WORD COUNT

7043 Words

CHARACTER COUNT

38471 Characters

PAGE COUNT

27 Pages

FILE SIZE

797.9KB

SUBMISSION DATE

May 26, 2023 11:33 PM GMT+5:30

REPORT DATE

May 26, 2023 11:33 PM GMT+5:30

12% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 3% Internet database
- 4% Publications database
- Crossref database
- Crossref Posted Content database
- 8% Submitted Works database