



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI

CHENNAI– 600119

www.sathyabama.ac.in



**Submitted in partial fulfillment of the requirements for the award of Bachelor
of Engineering Degree in Computer Science and Engineering**

By

D.Sowmith

REG. NO: 38110511

Sura Karthik

REG. NO: 38110239



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU, March
2022.



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI

CHENNAI- 600119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **D.Sowmith (38110511)** , **Sura Karthik (38110239)** who carried out the project entitled “**SARCASM DETECTION IN NEWS HEADLINES**” under my supervision from December 2021 to March 2022.

Internal Guide

Dr.A.Mary Posonia .M.E.,Ph.D.,

Head of the Department

Dr. L. Lakshmanan, M.E., Ph.D.

Dr. S. Vigneshwari, M.E., Ph.D.

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **D.Sowmith and Sura Karthik** hereby declare that the project report entitled **SARCASM DETECTION IN NEWS HEADLINES** done by me under the guidance of **Dr.A.Mary Posonia .M.E.,Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D, Dean, School of Computing, Dr. S. Vigneshwari, M.E., Ph.D. and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department of Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.A.Mary Posonia .M.E.,Ph.D.,,** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Sarcasm is an important part of communication. Sarcasm is expressed in words, facial expressions and can be noticed in the intonation of voice. So in this digital age , sarcastic comment are passed everyday via tweets, comment sections of different social media outlets , and even news headlines. Newspapers often seem to use sarcasm in their headlines to grab the reader's attention. Some of these headlines can be misunderstood and taken to mean something different than the original intentions. However, more often than , not the readers find it difficult to detect the irony within the headlines thus, getting the incorrect idea about the actual news and further passing on their understanding to their friends and colleagues .This leads to a need to detect sarcasm especially in the news and on social media. The sarcasm detection of text has its challenges because text lacks intonations and deflection of voices that occur when a sarcastic statement is formed vocally by a person's.

This project focuses on the effect of the different encoding methods used in the text to extract the feature of machine learning models. A deep learning model is used and the results are compared. Prior to the feature extraction, pre-processing data techniques such as tokenization, training set and testing set, and embedding concepts and punctuation are used by researchers in any work that involves textual analysis. These pre-processing methods are widely used and accepted and used in this project. Different methods of extracting features such as Count Vectorizer and word embedding were used in this project. Random Forest , and Logistic Regression were the machine learning algorithms used in this project.

Keywords: Sarcasm Detection, Convolutional Neural Networks (CNN), Random forest, Logistic regression, deep learning.

CONTENTS

<u>DESCRIPTION</u>	<u>PAGE NO.</u>
DECLARATION	2
CERTIFICATE	3
ACKNOWLEDGEMENT	4
ABSTRACT	5
CONTENTS	6
LIST OF FIGURES	7
LIST OF TABLES	8
1.INTRODUCTION	10
1.1 Objectives and Importance of the Project	11
1.2 Scope	12
1.3 Motivation	13
2. LITERATURE REVIEW	
2.1 Previous Work	15
2.2 Summary	16
3. PRE-PROCESSING DATA	18
3.1 Data Source	19
3.2 Tokenizing the data	21
3.3 Training and testing sets	22
3.4 Word Embeddings	23
4.CLASSIFICATION OF ALGORITHMS	
Random Forest Classifier	26
Logistic Regression	26
Convolutional Neural Networks (CNN)	27
5. EXPERIMENTAL RESULTS	30
Validating the models	31
data visualization for accuracy and loss	33
6. CONCLUSION AND FUTURE WORK	36
REFERNECE	37

LIST OF FIGURES

Figure 1: The distribution of headlines in the data set	20
Figure 2: Importing necessary modules	20
Figure 3: Loading data into array	20
Figure 4: Tokenizing the data	21
Figure 5: Padding the data	22
Figure 6: Building the model	22
Figure 7: Splitting the data into training and testing sets	23
Figure 8: Tokenizing and padding the splitted data	23
Figure 9: Embedding the data	24
Figure 10: Random forest classifier based accuracy	30
Figure 12: Logistic regression based accuracy	31
Figure 13: Using convolutional neural networks classifier based accuracy 1	32
Figure 14: Using convolutional neural networks classifier based accuracy 2	33
Figure 15: Data visualization for accuracy and loss	33
Figure16 : Training and testing accuracy model with each epoch	34
Figure 17: Training and testing loss model with each epoch	34

LIST OF TABLES

Table 1: Few headlines in the data set	19
Table 2: Model Summary Table 1	31
Table 3: Model Summary Table 2	32

CHAPTER 1

INTRODUCTION

Over the past decade, there has been an increase in the use of internet services disseminate information. These texts have distinctive features that are explored through the use of Natural Language Processing techniques in machine learning for knowledge and understanding. One of the most important things features the presence of sarcasm in the text. Sarcasm is a complex communication process that allows speakers to express rich ideas in an ambiguous way. There is a lot of ridicule in texts that are distributed online and on social media [1]. This includes news headlines and other media. Mocking is traditionally defined as the process of intentional misuse of words in order to convey a specific meaning (often the opposite of what is being said) [1]. For example, if someone says ‘How lucky I am? I caught a corona virus! "It is obvious that although the words used are constructive, the particular meaning is wrong, making the speech sarcastic. Sarcasm is characterised by the utilization of irony that reflects a negative meaning.

No one has ever used sarcasm in online comments or in the headlines. People can usually see this kind of sarcasm in person, but not all sarcastic words are so clear in the text. Computers strive to tell the difference in any way. We ask ourselves, can the language of the sarcastic language itself tell the computer that the word is sarcastic? Can a computer read these rules, or is the context involved? In our project, we will be taking an advanced route. Our goal is to build a sarcasm detector using machine learning through neural networks. The entries in our project contain a series of articles (newspaper articles) labeled as sarcastic or non-sarcastic. The Onion aims to produce current sarcastic versions of events, and we collected all the news headlines from the News Articles (funny). We collected real and non-sarcastic news headlines in HuffPost. This dataset is provided in Kaggle.

This project addresses the most problem of NLP known for Sarcasm Detection employing a combination of models supported Convolutional Neural Networks (CNNs). The discovery of humor is vital in other areas like affective computing and Sentimental analysis because such an expressions can change sentence variability.

OBJECTIVES AND IMPORTANCE OF SARCASM DETECTION IN NEWS HEADLINES

Although sarcasm is well-known and widely used, it is a challenge not only for computers but also for people can detect them quickly. Some people find it difficult to identify and understand the use of sarcasm [2]. Because of this fact, the presence of sarcasm, if not detected and calculated can interfere with machine learning activities such as sentimental analysis and opinion mining. This makes the sarcasm detection is an important task. Receiving automatic jokes can be seen as text classification problem. Text data is one of the easiest forms of data. The machine learning algorithms are unable to process non-numerical data and as a result the need for the feature extraction arises. It is important to take useful information from any type of data, especially the data which is in unstructured forms such as text data. Feature extract and appropriate representation of the text to achieve the purposes of classification is an important factor that affects the accuracy of the classification.

The Project consists of mainly three parts. The first task is pre processing data and second one is classification models and the third one is model elevation.

The first is pre-processing the dataset which contains of a data set, tokenizer and sequence, training set and testing set, word emdeddings , tokenizing and padding the splitted data.

The second part is classifiers used in this project are random forest, and Logistic regression. The main classifier is Convolutional Neural Networks (CNNs) which is used in the deep learning algorithm. By using these models ,we can get the accuracies for the sarcasm detection appropriately.

The third part consists of the performance elevation and results of the projects that involves combining all the models and provides the best result.

The importance of sarcasm detection can be in understanding the real feelings or sentiments and opinions of people. The use of sarcasm detection can benefit many areas of NLP

applications, including marketing research, opinion mining and information classification. Acquiring complex ideas and detection of ironic opinions can help companies and governments develop products and services. Reliable identification of sarcastic and sarcasm in the text can improve the effectiveness of natural language processing techniques.

SCOPE

We have successfully implemented and completed all three objectives in sarcasm detection where we have imported the necessary modules in the implementation , loading data into arrays and tokenizing the data and padding the data etc., and using the machine learning and deep learning techniques ,we have checked the accuracies using the random forest , nd logistic regression and finally we have used the Convolutional Neural Networks (CNNs) classifier and found out that it is the best classifier to detect sarcasm in our project. The results can be seen in the last section.

This report will explain the details of the literature used in this project (the list of references used and the review of some key reference journals/papers) and how the proposed model is different and better from the pre-existing ones, the limitations of the pre-existing and the proposed model and the future scope of how the proposed model can be improved.

We will explain the implementation of this project with diagrammatic explanation, the results derived after the implementation as well as the performance evaluation of the sarcasm detection in news headlines.

The scope of this project is to detect sarcasm in news headlines , where people like all of us sometimes doesn't understand the sarcastic meaning in the news paper article. So this leads to misunderstandings between us. So by this sarcasm detection , opinions of the people can help companies and governments develop products and services in their domain. When the newspaper headlines containing sarcasm sometime leads to conflicts between Politian's and the government . By using this detection , there will be less chances of conflicts and the government can also understand the peoples opinion on their political parties , so that they will know where to improve themselves.

MOTIVATION

Why sarcasm detection? Currently , the need to understand the sarcastic meaning in text is very less and people don't usually understand the sarcasm in text very quickly. There are so many ways of understanding sarcasm very easily such as like in pictures and pitch of voice by a human or vocally by a person can be understood very easily. The sarcasm is the use of words that can be a meaning opposite to the one you actually intend to pass on. It has the ability to flip the sentiment of the sentence this makes sarcasm detection an important part of the sentiment analysis. This detection is used to detect whether the newspaper headline is sarcastic or non sarcastic. Passing on sarcastic comments effects the persons feeling or sentimental emotions of the humans.

We are motivated to take this project because there are so many projects like Twitter sarcasm detection , social media sarcasm detection and images sarcasm detections and emoji's sarcasm detection and etc., these are done by many people. So we wanted to take this Sarcasm detection in news paper headlines which are done by very few people , so we wanted to experiment this project.

CHAPTER 2

LITERATURE REVIEW

The literature review on this project is done by gathering the work done by the researchers and authors.

PREVIOUS WORK

The detection of sarcasm is very important in the context of sentimental analysis and also in opinion mining. Different machine learning algorithms were used in this problem. Some researchers have already used the Naive Bayes Classifier and Support Vector machines for data analysis in social media in Indonesia [5]. Classifiers used in the work [5] described and did well in the job and appreciated it the decision to use Naive Bayes and SVM in this analysis [7]. They used vector support machines with Tf-IDF and Bag-Of-Words. The work of Davidov et.al [6] provides details on how to detect sarcasm using semi supervised methods on two different data sets, one containing a review of Amazon products and one with tweets collected on Twitter. These researchers focuses on features such as punctuation, syntax of sentences, hashtags used etc. The proposed system has more than 75% accuracy [6]. The work performed [14] by the reasearcher shows the use of Support Vector Machines and in order to detect ridicule from the media news headlines. The proper method of extracting the element used in this work results in points of accuracy are about 80%. Some researchers have used deep learning to the problem strategies [2]. Collecting data for the purpose of identifying sarcasm is even more challenging to read. This is especially so social media data case. This study uses some of the methods described in previous works with better accuracy results and other steps such as well.

Previous work has focused on almost Twitter data and sentimental analysis, but such data sets are noisy in terms of labels and language. In addition, many tweets are a response to some tweets and finding sarcasm and this need to be found contextual tweets. Therefore, we try to differentiate and find sarcasm in different ways. To overcome the noise-related restrictions on Twitter data sets and taking into account the fact that headline data has a similar structure compared to tweets, this discrimination News Database collected on two news websites.

Therefore, it will be interesting to see how the models interact with off-site data and how the head-trained model compares to existing tweet-based models.

SUMMARY

Text extraction describes the process of taking words from text data and converting them into a set of useful numeric features in a machine learning classifier [13]. Extracting features are useful for all analyzes that include text data for all domains. In the work detailed in [8] the impact of pre-processing is discussed and reviewed. Researchers determine the value of 'slang' and the correct spelling. The SVM classifier is used in their experiments. One new researcher uses vector presentations to provide a solution to the problem of emotional analysis [9], obtaining a high (86%) accuracy. Other studies look at the use of four data sets as well as the use of Word Bag, Dictionary and section-based features of speech [10]. An integrated SVM, Logistic Regression and Naive Bayes model was used for analysis. The authors in [11] used three levels of feature removal techniques. They also use three- dimensional dividers in their analysis. Another group of researchers in the work described

[12] used ten different methods for extracting features in the problem of analyzing emotions. They concluded that the problem-solving techniques used in a problem have the potential to improve the performance of a segment separator from any particular problem. Page [7] compares the results of 6 algorithms in which features are extracted using Bag-Of-Words and TF-IDF. They conclude that the TF-IDF provides the best results with a difference of up to 4%. Most of the research studies were conducted with SVM, Logistic Regression and Naive Bayes al gorithms for standard monitoring software applications. This led to the use of these algorithms in this analysis. The CNN model is also run at the end of the analysis using the Count Vectorizer vector as features and using word2vec-generated features.

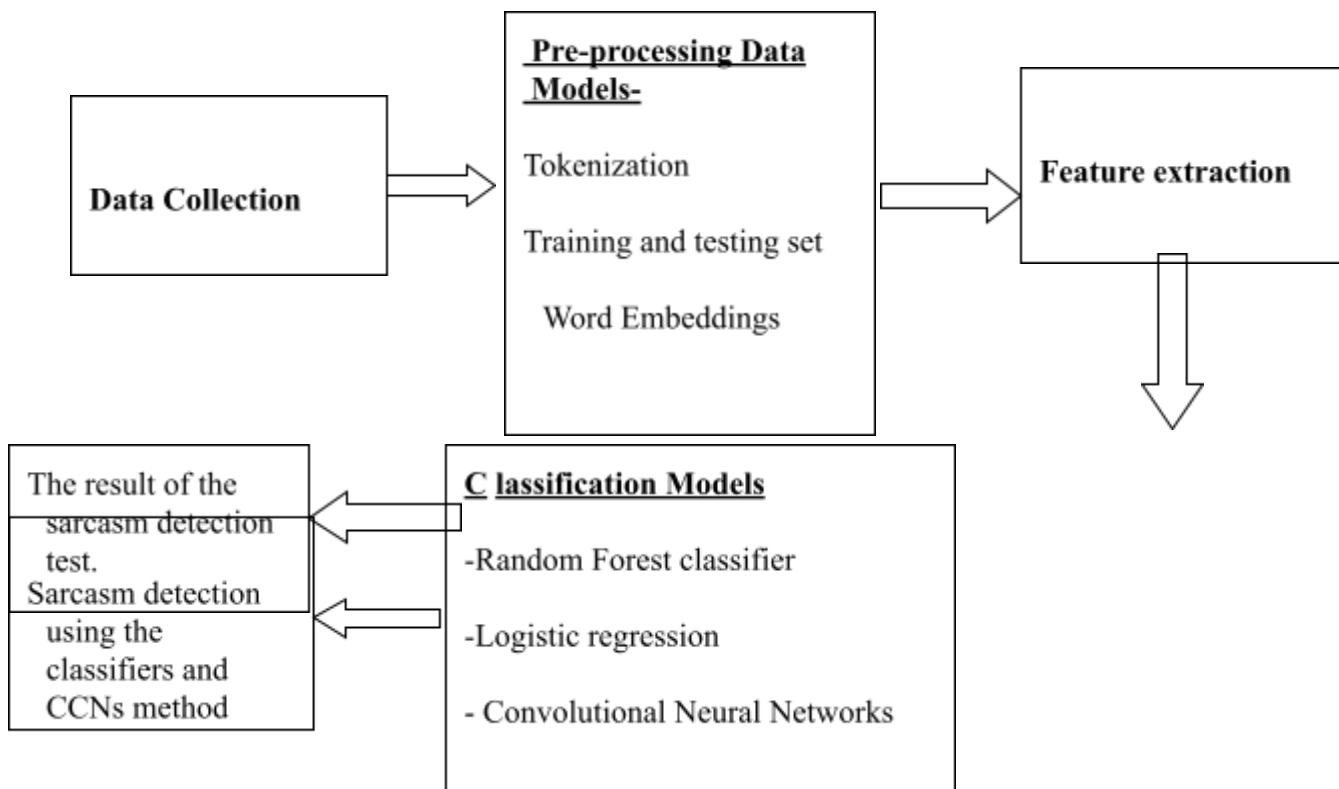
CHAPTER 3

PRE-PROCESSING DATA

The pre-processing data techniques such as tokenization, training set and testing set, and embedding concepts.

The second part is classifiers used in this project are random forest, and Logistic regression. The main classifier is Convolutional Neural Networks (CNNs) which is used in the deep learning algorithm. By using these models ,we can get the accuracies for the sarcasm detection appropriately.

The third part consists of the performance elevation and results of the projects that involves combining all the models and provides the best result. The flowchart of the sarcasm detection can been seen below.



Flow chart of sarcasm detection process

DATA SOURCE

Data set-

Data used in this analysis were taken from Kaggle. News headlines were collected into two separate tournament files in Kaggle. Each file was a JSON file, containing news headlines and a label called 'sarcastic' indicating the presence or absence of sarcasm in that title. The two files are grouped together to form a large data set for analysis. The total data used in the analysis contains more than 26000 news headlines. The data set is not very similar but there have been a few non-destructive news headlines rather than a sarcastic headline. Table 1 below shows a few lines of data and labels for those data points.

⇒ The is_sarcastic one is 1 if the record is sarcastic otherwise 0.

⇒ The headline is the headline of the news article

⇒ The article link is a URL to the text of the article

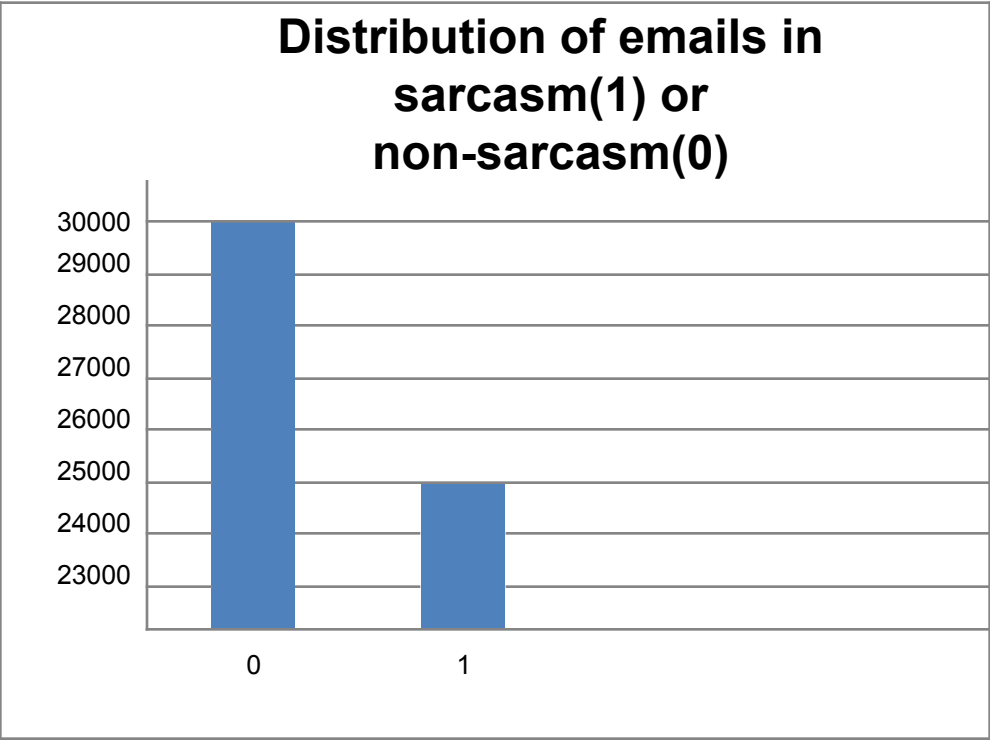
Table 1: Few headlines in the data set

	Headline	Is.sarcastic
1	former versace store clerk sues over secret 'black code' for minority shoppers	0
2	why writers must plan to be surprised	0
3	boehner just wants wife to listen, not come up with alternative debt-reduction ideas	1
4	remembrance is the beginning of the task	0
5	4 lessons prison taught me about power and control	0
6	top snake handler leaves sinking huckabee campaign	1
7	courtroom sketch artist has clear manga influences	1
8	stock analysts confused, frightened by boar market	1
9	gillian jacobs on what it's like to kiss adam brody	0
10	diy: sports equipment closet	0

⇒ First ,will import Jason so that we can use the Json parsers in Python. Then will open

the sarcasm Json file.

⇒ Using `Jsonload()` ,we can load and parse the entire thing



⇒ Then initializing arrays for the sentences, labels and URLs. Then for each item we will append it's a headline, the sarcasm level, and the URL to the appropriate array.

31000

22000

Figure 1: The distribution of headlines in the data set.

IMPORTING NECESSARY MODULES

Figure 2: Importing necessary modules

LOADING DATA INTO ARRAYS

```
[ ]
datastore = [json.loads(line) for line in open('sarcasm.json', 'r')]

sentences = []
labels = []
urls = []
for item in datastore:
    sentences.append(item['headline'])
    labels.append(item['is_sarcastic'])
    urls.append(item['article_link'])

print(sentences)
print(labels)
```

Figure 3: Loading data into array

TOKENIZING THE DATA

We created a tokenizer and fitted it on the sentence, here , the sentences are the large array of 25000 plus sentences that are read from the sarcasm data set.

We are using tokenizer to show us the word- index ,so we can see what words it learned from the data set.

We turned on the sentences into sequences where instead of words we have the tokens representing those words. Then the padding post which means that all of the sentences which will be the length of whichever is the longest word. And anything than that will be padded with zeros at the end of the sentences. In order to keep them all the same length.

If we want to inspect them, we can the print put one of them and we can print out the shape of the entire padded data.

There are 26709 padded sentences, and each of those is 40 value long.

There are about 26,000 words and 29657 unique independent words are there in this word index.

We have 26,709 headlines and the maximum length of each one is 40.

TOKENIZING THE DATA

```
[ ] tokenizer = Tokenizer(oov_token="<OOV>")
tokenizer.fit_on_texts(sentences)

word_index = tokenizer.word_index
print(len(word_index))
print(word_index)

29657
{'<OOV>': 1, 'to': 2, 'of': 3, 'the': 4, 'in': 5, 'for': 6, 'a': 7, 'on': 8, 'and': 9, 'with': 10, 'is': 11, 'new': 12, 'trump': 13, 'man': 14, 'from': 15, 'at': 16, ' '
```

Figure 4: Tokenizing the data

Padding the data:

PADDING THE DATA

```
[ ] sequences = tokenizer.texts_to_sequences(sentences)
    padded = pad_sequences(sequences, padding='post')
    print(padded[0])
    print(padded.shape)

[  308 15115   679  3337  2298    48   382  2576 15116     6  2577  8434
    0      0      0      0      0      0      0      0      0      0      0
    0      0      0      0      0      0      0      0      0      0      0
    0      0      0      0]
(26709, 40)
```

Figure 5: Padding the data

TRAINING SET AND TESTING SET

- Building a model that understands this text enough to classify future sentences as sarcastic or not sarcastic.
- We took the training size as 20,000 and testing size as 6000.
- We have a corpus of many thousands of sentences and labels.
- So, the training sentences will be the complete corpus from zero to the training size.
- And testing sentences will be the from training size to the end of the testing size.
- So, the training labels will be the complete corpus from zero to the training size
- And testing labels will be the from training size to the end of the testing size.

BUILDING THE MODEL

```
[ ] vocab_size = 10000
    embedding_dim = 16
    max_length = 100
    trunc_type='post'
    padding_type='post'
    oov_tok = "<OOV>"
    training_size = 20000
```

Figure 6: Building the model

SPLITTING THE DATA INTO TRAINING AND TEST SETS

```
[ ] training_sentences = sentences[0:training_size]
    testing_sentences = sentences[training_size:]
    training_labels = labels[0:training_size]
    testing_labels = labels[training_size:]
```

Figure 7: Splitting the data into training and testing sets

TOKENIZING AND PADDING THE SPLITTED DATA

```
[ ] tokenizer = Tokenizer(num_words=vocab_size, oov_token=oov_tok)
    tokenizer.fit_on_texts(training_sentences)

    word_index = tokenizer.word_index

    training_sequences = tokenizer.texts_to_sequences(training_sentences)
    training_padded = pad_sequences(training_sequences, maxlen=max_length, padding=padding_type, truncating=trunc_type)

    testing_sequences = tokenizer.texts_to_sequences(testing_sentences)
    testing_padded = pad_sequences(testing_sequences, maxlen=max_length, padding=padding_type, truncating=trunc_type)
```

Figure 8: Tokenizing and padding the splitted data

WORD EMBEDDINGS

The concept of embedding which helps us turn the sentiment of a word into a number in much the same way as tokenized words.

⇒ In this case, an embedding is a vector pointing in a direction, and by using those directions to establish meanings in words.

⇒ The coordinates of words are seen to as embedding for the sentiment of those words.

There is no absolute meaning but, relatively to each other we can then establish sentiment.

⇒ We used be used keras layer called embedding. Our embedding should be defined as

a vector for every word.

- ⇒ We took vocab size words and then specified how many dimensions we want the arrow direction to use.
- ⇒ In this case, it's 16 that we created earlier.
- ⇒ So, the embedding layer will learn 10,016 dimension vectors, way the direction of the vector establish is the sentiment of the word.
- ⇒ By matching the words to the labels, it will have a direction that it can Start learning from.

The model is an embedding, followed by a global average pooling, followed by a dense layer, which has 24 neurons in it , followed by a dense layer for the output with a single neuron. And this is the binary classifier.

EMBEDDING THE DATA

```
[ ] model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),
    tf.keras.layers.GlobalAveragePooling1D(),
    tf.keras.layers.Dense(24, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
```

```
[ ] model2 = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),
    tf.keras.layers.GlobalAveragePooling1D(),
    tf.keras.layers.Dense(30, activation='relu'),
    tf.keras.layers.Dense(10, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
model2.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
```

Figure 9: Embedding the data

CHAPTER 4

CLASSIFICATION OF ALGORITHMS

A couple of classification algorithms were chosen for this project. These algorithms are briefly described below:

RANDOM FOREST CLASSIFIER

Random Forest Classifier uses the Scikit-Learn library of Python programming language and to do this, we used the IRIS database which is a common and popular dataset. Random Forest or Random Decision Forest is a supervised machine learning algorithm used for classification, regression, and other activities using decision trees.

A random forest section creates a set of decision trees from a randomly selected small set of training set. It is basically a set of decision trees (DT) from a randomly selected subset set of training set and Collect votes on different decision trees to determine the final prediction.

LOGISTIC REGRESSION

Logistic Regression is a popular classification algorithm. It belongs to the class of Generalized Linear Models. Its loss function is the sigmoid function which minimizes the results to be a value between 0 and 1. The Logistic Regression Model was implemented for this analysis using the function available in the scikit-learn package.

Logistic regression is a statistical analysis method used to predict the amount of data based on the preview of a data set. Logistic regression has become an important tool in training machine learning. This method allows an algorithm used in machine learning applications to separate incoming data based on historical data. As more relevant data enters, the algorithm should be better at predicting divisions between data sets. Processing may also contribute to data processing activities by allowing data sets to be placed in pre-defined buckets during extraction, conversion, loading (ETL) to process information for analysis.

In this algorithm, we used the training and testing padded and training and testing lables.

CONVOLUTIONAL NEURAL NETWORKS (CNN)

In deep learning, the convolutional neural network is one of the most important algorithm and it is a phase of the neural network of artificial, which is widely used to analyze visual images[13]. They are also known as space variant artificial networks, based on the distributed weight structure of convolution characters or slider filters near input features and provide equivalent translation responses known as feature maps. Contrary to intuitively, most convolutional neural networks are only equal, as opposed to consistent, and interpretive. They contain applications for image and video recognition, complimentary programs, image classification, image classification, medical image analysis, natural language processing, brain-computer communication, and a series of financial periods.

CNNs are standard versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in a single layer is connected to all the neurons in the next layer. The "full connection" of these networks makes them prone to data overload[13]. Typical ways to do so, or to prevent excessive immersion, include: punitive restrictions during training or interconnection (skipping links, stopping out, etc.) . and incorporates complex growth patterns using small and simple patterns labeled in their filters. Therefore, in terms of connectivity and complexity, CNNs are at a very low level.

Transformation networks are promoted by biological processes in that the pattern of communication between neurons resembles the left visual association of an animal. Each cortical neurons respond to stimuli only to a limited area of the visual field known as the receptor field. The receiving fields of different neurons are so intertwined that they close the entire field of view.

CNN uses less advanced processing compared to other image classification algorithms. This means that the network learns to optimize filters (or kernels) by automatic reading, and in traditional algorithms these filters are manually made. This independence from previous knowledge and human intervention in the removal of the element is a great advantage.

This is most important classifier in our project ,which will detect the most accuracy than other classifiers which were used to detect.

CHAPTER 5

EXPERIMENTAL RESULTS

Here, we are going to see the results of the implementation performed.

Random forest classifier

Based on random forest classifier, we got the accuracy 0.6683 %. The figure below shows the accuracy.

RANDOM FOREST CLASSIFIER

```
[ ] from sklearn.ensemble import RandomForestClassifier
    m=RandomForestClassifier(n_estimators=400,max_depth=25)
    m.fit(training_padded,training_labels)
    m.score(testing_padded,testing_labels)
```

0.6683559397823818

```
[ ]
```

Figure 10: Random forest classifier based accuracy Logistic regression

Based on Logistic regression, we got the accuracy 0.5732 %. The figure below shows the accuracy.

▼ LOGISTIC REGRESSION

```
[ ] from sklearn.linear_model import LogisticRegression
logreg=LogisticRegression()
logreg.fit(training_padded,training_labels)
logreg.score(testing_padded,testing_labels)
```

C:\Users\Sravanth\anaconda3\lib\site-packages\sklearn\linear_model_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

n_iter_i = _check_optimize_result(
0.5732598002682964

Figure 12: Logistic regression based accuracy

VALIDATING THE MODELS

The following table gives the information about the number of hidden layers present in the models and the number of each layer nodes. The direction of each (embeddings).

Table 2:Model Summary

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	(None, 100, 16)	160000
global_average_pooling1d_2 (GlobalAveragePooling1D)	(None, 16)	0
dense_5 (Dense)	(None, 30)	510
dense_6 (Dense)	(None, 10)	310
dense_7 (Dense)	(None, 1)	11
Total params: 160,831		
Trainable params: 160,831		
Non-trainable params: 0		

After training the model and then testing it on our test dataset, the accuracies gradually increases , we see that we achieved 0.9937% accuracy, with loss of around 1.265.

```

625/625 - 2s - loss: 0.0446 - accuracy: 0.9862 - val_loss: 0.8419 - val_accuracy: 0.8193 - 2s/epoch - 3ms/step
Epoch 18/30
625/625 - 2s - loss: 0.0431 - accuracy: 0.9855 - val_loss: 0.8627 - val_accuracy: 0.8177 - 2s/epoch - 3ms/step
Epoch 19/30
625/625 - 2s - loss: 0.0373 - accuracy: 0.9877 - val_loss: 0.8996 - val_accuracy: 0.8167 - 2s/epoch - 3ms/step
Epoch 20/30
625/625 - 2s - loss: 0.0324 - accuracy: 0.9904 - val_loss: 1.0214 - val_accuracy: 0.8116 - 2s/epoch - 3ms/step
Epoch 21/30
625/625 - 2s - loss: 0.0292 - accuracy: 0.9912 - val_loss: 1.0106 - val_accuracy: 0.8146 - 2s/epoch - 3ms/step
Epoch 22/30
625/625 - 2s - loss: 0.0293 - accuracy: 0.9912 - val_loss: 1.0207 - val_accuracy: 0.8168 - 2s/epoch - 3ms/step
Epoch 23/30
625/625 - 2s - loss: 0.0250 - accuracy: 0.9923 - val_loss: 1.1114 - val_accuracy: 0.8137 - 2s/epoch - 3ms/step
Epoch 24/30
625/625 - 2s - loss: 0.0238 - accuracy: 0.9927 - val_loss: 1.1184 - val_accuracy: 0.8122 - 2s/epoch - 3ms/step
Epoch 25/30
625/625 - 2s - loss: 0.0239 - accuracy: 0.9921 - val_loss: 1.1968 - val_accuracy: 0.8141 - 2s/epoch - 3ms/step
Epoch 26/30
625/625 - 2s - loss: 0.0234 - accuracy: 0.9923 - val_loss: 1.1697 - val_accuracy: 0.8113 - 2s/epoch - 3ms/step
Epoch 27/30
625/625 - 2s - loss: 0.0197 - accuracy: 0.9939 - val_loss: 1.1942 - val_accuracy: 0.8074 - 2s/epoch - 4ms/step
Epoch 28/30
625/625 - 2s - loss: 0.0209 - accuracy: 0.9938 - val_loss: 1.2271 - val_accuracy: 0.8089 - 2s/epoch - 3ms/step
Epoch 29/30
625/625 - 2s - loss: 0.0174 - accuracy: 0.9945 - val_loss: 1.2488 - val_accuracy: 0.8109 - 2s/epoch - 3ms/step
Epoch 30/30
625/625 - 2s - loss: 0.0196 - accuracy: 0.9937 - val_loss: 1.2657 - val_accuracy: 0.8038 - 2s/epoch - 3ms/step

```

Figure 13: Using convolutional neural networks classifier based accuracy 1

Table 3:Model Summary

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 100, 16)	160000
global_average_pooling1d (GlobalAveragePooling1D)	(None, 16)	0
dense (Dense)	(None, 24)	408
dense_1 (Dense)	(None, 1)	25
Total params: 160,433		
Trainable params: 160,433		
Non-trainable params: 0		

After training the model and then testing it on our test dataset, we see that we achieved 0.9944% accuracy, and val_accuracy of 0.8113 with loss of around 1.2101.

Here , the val_accuracy is a bit more compared to before predictions .so, what happens here

is it gets overfitted in it, when we use more than 3 layers. The best is to use upto 2 layers which gives us the satisfying accuracy.

```
Epoch 17/30
625/625 - 2s - loss: 0.0619 - accuracy: 0.9804 - val_loss: 0.7127 - val_accuracy: 0.8225 - 2s/epoch - 3ms/step
Epoch 18/30
625/625 - 2s - loss: 0.0567 - accuracy: 0.9827 - val_loss: 0.7068 - val_accuracy: 0.8258 - 2s/epoch - 3ms/step
Epoch 19/30
625/625 - 2s - loss: 0.0521 - accuracy: 0.9832 - val_loss: 0.7844 - val_accuracy: 0.8225 - 2s/epoch - 3ms/step
Epoch 20/30
625/625 - 2s - loss: 0.0493 - accuracy: 0.9847 - val_loss: 0.8172 - val_accuracy: 0.8213 - 2s/epoch - 3ms/step
Epoch 21/30
625/625 - 2s - loss: 0.0445 - accuracy: 0.9858 - val_loss: 0.8137 - val_accuracy: 0.8204 - 2s/epoch - 3ms/step
Epoch 22/30
625/625 - 2s - loss: 0.0449 - accuracy: 0.9847 - val_loss: 0.8445 - val_accuracy: 0.8232 - 2s/epoch - 3ms/step
Epoch 23/30
625/625 - 2s - loss: 0.0369 - accuracy: 0.9890 - val_loss: 0.8873 - val_accuracy: 0.8208 - 2s/epoch - 3ms/step
Epoch 24/30
625/625 - 2s - loss: 0.0348 - accuracy: 0.9898 - val_loss: 0.9299 - val_accuracy: 0.8195 - 2s/epoch - 3ms/step
Epoch 25/30
625/625 - 2s - loss: 0.0311 - accuracy: 0.9912 - val_loss: 0.9712 - val_accuracy: 0.8195 - 2s/epoch - 3ms/step
Epoch 26/30
625/625 - 2s - loss: 0.0287 - accuracy: 0.9920 - val_loss: 1.0016 - val_accuracy: 0.8167 - 2s/epoch - 3ms/step
Epoch 27/30
625/625 - 2s - loss: 0.0265 - accuracy: 0.9927 - val_loss: 1.0564 - val_accuracy: 0.8152 - 2s/epoch - 3ms/step
Epoch 28/30
625/625 - 2s - loss: 0.0249 - accuracy: 0.9926 - val_loss: 1.1360 - val_accuracy: 0.8146 - 2s/epoch - 3ms/step
Epoch 29/30
625/625 - 2s - loss: 0.0234 - accuracy: 0.9933 - val_loss: 1.2515 - val_accuracy: 0.8083 - 2s/epoch - 3ms/step
Epoch 30/30
625/625 - 2s - loss: 0.0212 - accuracy: 0.9944 - val_loss: 1.2101 - val_accuracy: 0.8113 - 2s/epoch - 3ms/step
```

Figure 14: Using convolutional neural networks classifier based accuracy 2

DATA VISUALIZATION FOR ACCURACY AND LOSS

```
import matplotlib.pyplot as plt

def plot_graphs(history, string):
    plt.plot(history.history[string])
    plt.plot(history.history['val_'+string])
    plt.xlabel("Epochs")
    plt.ylabel(string)
    plt.legend([string, 'val_'+string])
    plt.show()

plot_graphs(history, "accuracy")
plot_graphs(history, "loss")
```

Figure 15: Data visualization for accuracy and loss

On plotting the graphs for Model Accuracy and Model Loss, we get the following results. Figure 16 shows the model training and testing accuracy with each epoch of both the datasets and figure 17 shows the model training and testing loss with each epoch for the datasets.

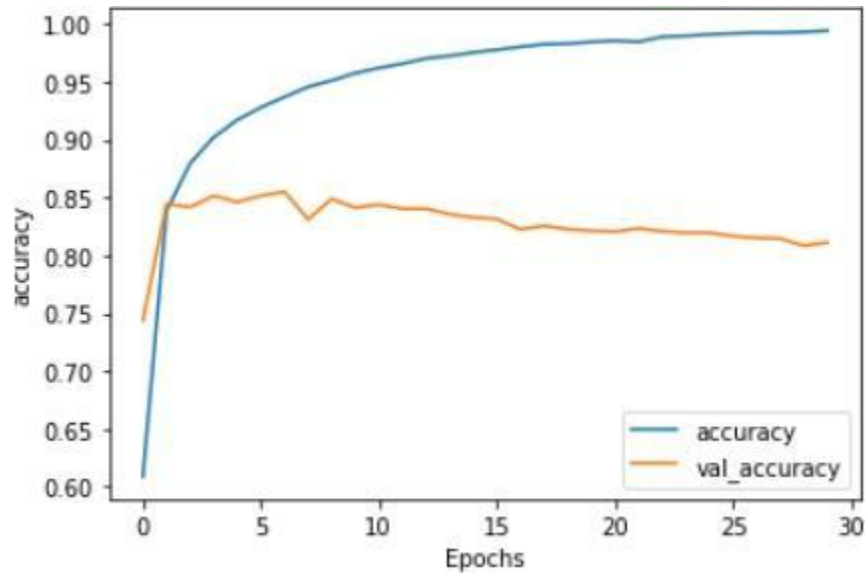


Figure16 : Training and testing accuracy model with each epoch

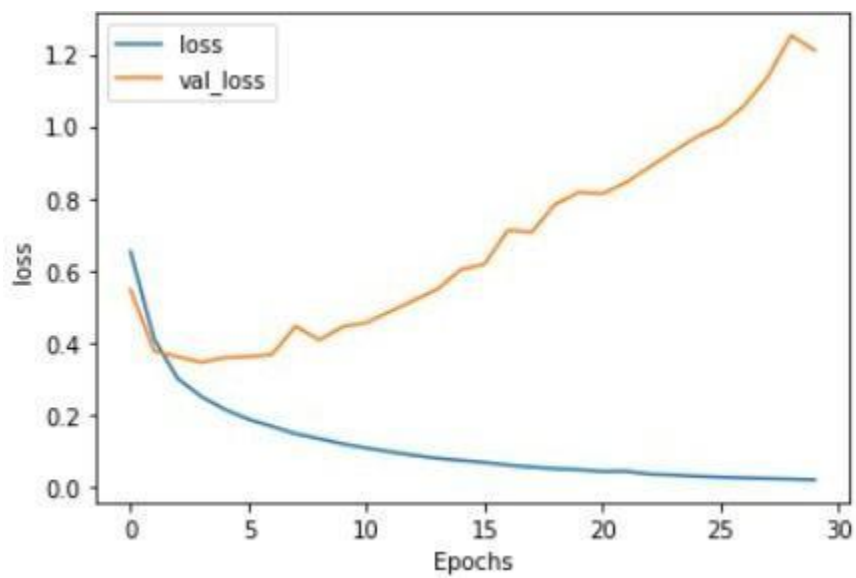


Figure 17: Training and testing loss model with each epoch

CHAPTER 6

CONCLUSION

Sarcasm can be found in many places. So that you can see accurately sarcasm from different aspects and topics, will need a large enough set of data. The accuracy of the prediction as shown in this project depends largely on the CNNs classifier. The convolution neural network classifier which is a deep learning technique gave the most satisfactory performance result. The results of this study favor the methods of the supervised machine over deep learning methods. Although this may be due to the limited value of this available data. This will show that the use of simple methods can produce good results, especially in a situation where available data is also limited is not suitable for deep learning methods. The use of word embedding in this work has shown great efficiency with precision, both in training sets and testing set. Total performance in both CNN and supervised learning methods suggests that the use of CNN is well suited for this task. The use of machine learning algorithms that we used in the project did not show good result as compared to deep learning techniques like CNN showed the best result accuracy as mentioned above in the result section.

REFERENCES

- [1] Liu, P., Chen, W., Ou, G., Wang, T., Yang, D., Lei, K.: Sarcasm Detection in Social Media Based on Imbalanced Classification In: Web-Age Information Management, pp. 459-471. Cham, (2014).
- [2] Mehndiratta, P., Sachdeva, S., Soni, D. :Detection of Sarcasm in Text Data using Deep Convolutional Neural Networks In: Scientific International Journal for Parallel and Distributed Computing, vol. 18, September (2017).
- [3] Zhibo, W. , Ma, L., and Zhang, Y. : A Hybrid Document Feature Extraction Method Using Latent Dirichlet Allocation and Word2Vec. In : IEEE First International Conference on Data Science in Cyberspace, June (2016).
- [4] Zhang, W., Xu, W., Chen G., Guo, J. : A Feature Extraction Method Based on Word Embedding for Word Similarity Computing. In: Communications in Computer and Information Science, vol. 496, pp. 160-167 January, (2014).

- [5] Lunando, E., Purwarianti, A. : Indonesian Social Media Sentiment Analysis With Sarcasm Detection. In: Int. Conf. Adv. Comput. Sci. Inf. Syst. ICACSIS, pp. 195-198. September (2013).
- [6] Davidov, D., Tsur, O., Rappoport, A. : Semi-Supervised Recognition of Sarcasm in Twitter and Amazon. In: Proceedings of the Fourteenth Conference on Computational Natural Language Learning, pp.107–116, July (2010)
- [7] Ahuja, R., Chug, A., Kohli, S., Gupta, S., Ahuja, P. : The Impact of Features Extraction on the Sentiment Analysis In: Procedia Computer Science, vol. 152, pp. 341-348. Elsevier, January (2019).
- [8] Agarwal, A., Xie, B., Vovsha, I., Rambow, O., and Passonneau R.:Sentiment Analysis of Twitter Data In: Proceedings of the Workshop on Language in Social Media, pp. 30-38, June (2011).
- [9] D. Das and A. Clark, “Sarcasm detection on flickr using a cnn”, in International Conference on Computing and Big Data, (2018).
- [10] Angulakshmi, G., Manicka Chezian, R.: Three level feature extraction for sentiment classification. In: International Journal of Innovative Research in Computer and Communication Engineering 2, vol. 8,pp. 5501-5507 (2014).
- [11] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “Pre- training of deep bidirectional transformers for language understanding”, (2018).
- [12] Waykole, R.N, Thakare, A.D: A Review Of Feature Extraction Methods For Text Classification. In: International Journal of Advance Engineering and Research Development, vol. 5, April (2018).
- [13] Detecting Sarcasm in News Headlines, Onyinye Chudi-Iwueze 1 and Haithem Afli 2 ADAPT Centre, Cork Institute of Technology, Cork, Ireland.(2019).
- [14] Mandal, Paul & Mahto, Rakeshkumar. Deep CNN-LSTM with Word Embeddings for News Headline Sarcasm Detection (2019).