TABLE OF CONTENTS

ABSTRACT

Sarcasm is a form of language in which individuals convey their message in an implicit way i.e. the opposite of what is implied. Sarcasm detection is the task of predicting sarcasm in text. This is the crucial step in sentiment analysis due to the inherently ambiguous nature of sarcasm. With this ambiguity, sarcasm detection has always been a difficult task, even for humans. Therefore sarcasm detection has gained importance in many Natural Language Processing applications.The main challenge in Sentimental Analysis is the presence of sarcasm. Sarcasm is a sophisticated form of language use that acknowledges a gap between the intended meaning and the literal meaning of the words.With this ambiguity, sarcasm detection is a difficult task even for humans. People regularly express it verbally using heavy tonal stress and certain gestural intimations like rolling of the eyes. This tonal and gestural information is clearly not available for communicating sarcasm in content, making its detection dependent upon different variables. We aim at using a recurrent neural network model for sarcasm detection because it automatically extracts features required for machine learning approaches.Along with the recurrent neural network,this model also uses long short-term memory(LSTM) cells on tensorflow to capture syntactic and semantic information over Twitter tweets to detect sarcasm .Finally we present the result of this model and a statistical overview of the dataset.

LIST OF FIGURES

LIST OF TABLES

Chapter 1
INTRODUCTION

1.1. Features

Sarcasm is defined as a cutting, often ironic remark intended to express contempt or ridicule. Sarcasm detection is a challenging task owing to the lack of intonation and facial expressions in text. Nonetheless humans can still spot a sarcastic sentiment in the text and reason about what makes it so.

Recognizing sarcasm in text is an important task for Natural Language processing to avoid misinterpretation of sarcastic statements as literal statements. Accuracy and robustness of NLP models are often affected by untruthful sentiments that are often of sarcastic nature. Thus, it is important to filter out noisy data from the training data inputs for various NLP related tasks. For example, a sentence like" So thrilled to be on call for work the entire weekend!" could be naively classified as a sentence with a high positive sentiment. However, it's actually the negative sentiment that is cleverly implied through sarcasm.

The use of sarcasm is prevalent across all social media, micro-blogging and e-commerce platforms. Sarcasm detection is imperative for accurate sentiment analysis and opinion mining. It could contribute to enhanced automated feedback systems in the context of customer-based sites. Twitter is a micro-blogging platform extensively used by people to express thoughts, reviews, discussions on current events and convey information in the form of short texts. The relevant context of the tweets are often specified with the use of #(hash-tag). Twitter data provides a diverse corpus for sentences which implicitly contain sarcasm.

We present an RNN-LSTM based approach to detect sarcasm in day to day sentences. This implements a Neural network based approach and its variants that take different features as input for learning. The use of Long-short Term Memory (LSTM) network is able to handle sequences of any length and capture long term dependencies. We detect whether the sentence is 'Sarcastic' or 'Non-Sarcastic'.

1.2. Objectives

1. To identify various techniques for processing of semantics.
2. To study the various techniques and identify their limitations that may help to suggest an approach which may overcome the drawbacks of existing methods.
3. To study various methods available for data pre-processing i.e tokenization, stemming, Noise removal etc.
4. To understand methods of feature extraction for the specified sentence that will help the model for decision making.
5. To help identify various machine learning and neural network techniques available for text classification.
6. To study RNN-LSTM for sentiment analysis and text classification.

1.3. Scope

Computational detection of sarcasm has seen attention from the sentiment analysis community in the past few years. Sarcasm is an interesting problem for sentiment analysis because surface sentiment of words in a sarcastic text may be different from the implied sentiment. For example, ‘Being stranded in traffic is the best way to start a week’ is a sarcastic sentence because the surface sentiment of the word ‘best’ (positive) is different from the implied sentiment of the sentence (negative), considering remaining portions of the text.

We have proposed an algorithm to understand if a sentence or tweet is sarcastic or not. From the dataset we have acquired we are trying to train our model so that we test tweets from Twitter API to detect whether they are sarcastic or not. It is a culmination of recurrent neural networks for sarcasm detection using Keras. We implement the neural network model that contains recurrent neural networks and LSTM (Long Short Term Recollection) which automatically extracts the features evading the overhead of exclusively extracting the features. This will result in obtaining precise sentiment analysis and opinion mining. It could contribute to enhanced automated feedback systems in the context of customer predicated sites.

1.4. Organization of the report

The introduction is given in Chapter 1. It describes the fundamental terms used in this project. It motivates to study and understand the different techniques used in this work. This chapter also presents the outline of the objective of the report. The Chapter 2 describes the review of the relevant various techniques in the literature systems. It describes the pros and cons of each technique. The Chapter 3 presents the Theory and proposed work. It describes the major approaches used in this work. The societal and technical applications are mentioned in Chapter 4. The summary of the report is presented in Chapter 5.

Chapter 2
LITERATURE SURVEY

2.1. Introduction

Sarcasm is a form of expressing negative feelings using positive words. Sarcasm is also when people mean something else from what they speak. Sarcasm is used not only to make fun but also for criticizing other people, views, ideas etc. due to which sarcasm is very much used on twitter. Sarcasm can be conveyed in various ways like a direct conversation, speech, text etc. It can be reflected using the rating of stars by providing less number of stars. There are many applications for detecting sarcasm. It is used to let the analyst know the intent of the user and the situation in which it is said. Sarcasm is more prevalent in the places where there are capital letters, emoticons, and exclamation marks etc. Sarcasm detection is one of the prominent tasks in sentiment analysis. On Amazon and shopping websites, it helps to understand the review of the product. The consumer's preferences and opinions can be analyzed in order to understand the market behavior for better consumer experience.

2.2. Literature review

Liu et. al, in their work have used the multitask learning framework to jointly learn across multiple related tasks. Based on recurrent neural networks, we propose three different mechanisms of sharing information to model text with task-specific and shared layers. The entire network is trained jointly on all these tasks. Experiments on four benchmark text classification tasks show that our proposed models can improve the performance of a task with the help of other related tasks.In this paper, they've introduced three RNN based architectures to model text sequence with multi-task learning. The differences among them are the mechanisms of sharing information among the several tasks. Experimental results show that their models can improve the performances of a group of related tasks by exploring common features [3].

Yao et. al, in their work constructs an improved NLP method based on long short-term memory (LSTM) structure, in which randomly discard some parameter values when the parameters are passed backwards in the Recurrent Projection Layer. Compared with baseline and other LSTM, the improved method has better F1 Score results on Wall Street Journal dataset both word2vec word vector and one-hot word vector, which implied their method is more suitable for NLP in the limited computing resources and high input data volume[4].

Joshi et. al, in their article mentions a compilation of past work in automatic sarcasm detection. They observed three milestones in the research so far: semi-supervised pattern extraction to identify implicit sentiment, use of hashtag-based supervision, and incorporation of context beyond target text. In this article, they have described datasets, approaches, trends, and issues in sarcasm detection. They have also discussed representative performance values, describe shared tasks, and provide pointers to future work, as given in prior works. In terms of resources to understand the state-of-the-art, the survey presents several useful illustrations—most prominently, a table that summarizes past papers along different dimensions such as the types of features, annotation techniques, and datasets used[5].

Ghosh et. al, in their paper, have proposed a neural network semantic model for the task of sarcasm detection. We also review semantic modelling using Support Vector Machine (SVM) that employs constituency parse trees fed and labeled with syntactic and semantic information. The proposed neural network model is composed of Convolution Neural Network(CNN) and followed by a Long short term memory (LSTM) network and finally a Deep neural network(DNN). The proposed model outperforms state-of-the-art text based methods for sarcasm detection[6].

Ostwal et. al, in their work, have exploited a deep neural network for sarcasm detection using Tensorflow. They built the neural network model which contains recurrent neural networks and LSTM which automatically extracts the features avoiding the overhead of exclusively extracting the feature. They are training and testing the model on their self-designed twitter dataset. Results on a tweet dataset show that the neural model achieves significantly better accuracies compared to the previous studies done on sarcasm detection using linguistic approach[7].

Manohar et. al, [8] the authors proposed a new approach for sarcasm detection as NLP and Corpus-based approach. The objective was to identify the intention to use the sarcastic statement in the tweets by individuals. The authors collected the tweets from the Twitter website and NLP techniques like tokenization, parts of speech (PoS), and lemmatization are performed.NLP techniques on tweets are applied to fetch action words. Once the action words are found from the tweets, these are matched with the corpus of sarcasm data using semantic matching and graph-based matching which gives a score of sarcasm for the given tweet. By this score, the level of sarcasm in the given tweet is detected.

2.3. Summary of literature review

Authors	**Paper**	**Methods used**
Liu et. al [3]	Recurrent Neural Network for Text Classification with Multi-Task Learning	In this paper, they've proposed three different mechanisms of sharing information to model text with task-specific and shared layers.
Yao et. al [4]	An Improved LSTM Structure for Natural Language Processing	In their work, they have constructed an improved NLP method based on long short-term memory (LSTM) structure
Joshi et. al [5]	Automatic Sarcasm Detection: A Survey	This article mentions a compilation of past work in automatic sarcasm detection.

Ghosh et. al [6]	Fracking Sarcasm using Neural Network	In their paper, they have proposed a neural network semantic model for the task of sarcasm detection, composed of Convolution Neural Network(CNN) and followed by a Long short term memory (LSTM) network.
Ostwal et. al [7]	Sarcasm detection using Recurrent Neural Network	In their work, they have exploited a deep neural network for sarcasm detection using Tensorflow. They built the neural network model which contains recurrent neural networks and LSTM .
Manohar et. al [8]	Improvement Sarcasm Analysis using NLP and Corpus based Approach	The authors have proposed a new approach for sarcasm detection as NLP and Corpus-based approach.

Table. 2.1 Summary of literature review

Chapter 3
PROJECT IMPLEMENTATION

3.1. Overview

3.1.1. Existing systems

In the system proposed by Dharwal et. al, Logistic regression technique is used to detect these sarcastic tweets, it has a drawback as it cannot predict for continuous variables[9]. As noted in the survey, past work does not report which of the types of sarcasm are correctly handled by existing systems. Sarcasm and irony are closely related and most work so far considers them to be the same. However, some recent work has dealt with understanding the differences between the two. [10] Present findings of a data analysis to understand differences between sarcasm and irony. According to them, aggressiveness is the distinguishing factor between the two. [11] Presents a set of classifiers that distinguish between sarcasm and irony. They describe an analysis of structural and adjective features in tweets. An important observation that they make is the peculiarity of the hashtag '#not' as a negation marker for sarcasm.

3.1.2. Proposed system

Our algorithm uses RNN-LSTM model for classifying and detecting text as 'Sarcastic' or 'Non-Sarcastic'.

RNN is a sequence of neural network blocks that are linked to each other like a chain. This chain-like nature reveals that recurrent neural networks are intimately related to sequences and lists. They're the natural architecture of neural network to use for such data.

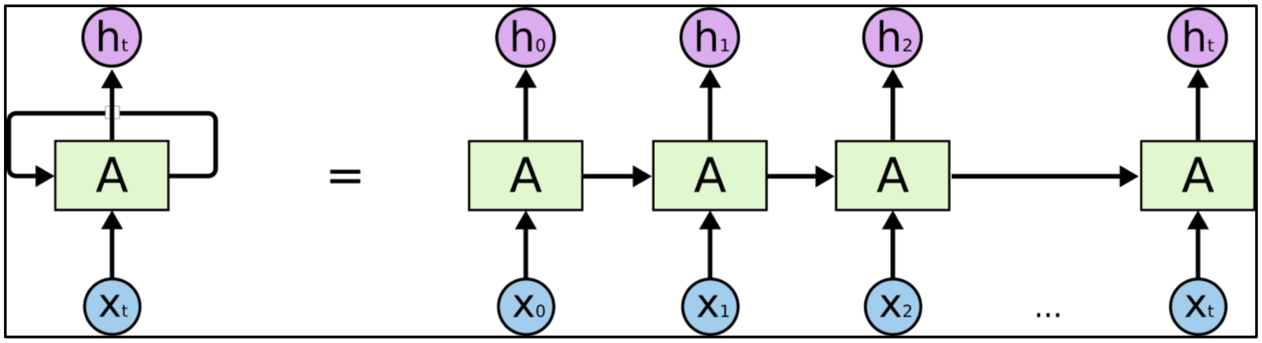


Figure 3.1. RNN Structure

Each one is passing a message to a successor. Text data is processed as a sequence type. The order of words is very important to the meaning. RNNs take care of this and can capture long-term dependencies.

The challenging problem faced by researchers is the long-term dependencies that one can find in text. For example, if someone feeds a sequence like I used to live in France and I learned how to speak... the next obvious word in the sequence is the word French. Recent information suggests that the next word is probably the name of a language, but if we want to narrow down which language, we need the context of France, from further back. It's entirely possible for the gap between the relevant information and the point where it is needed to become very large. Unfortunately, as that gap grows, RNNs become unable to learn to connect the information.

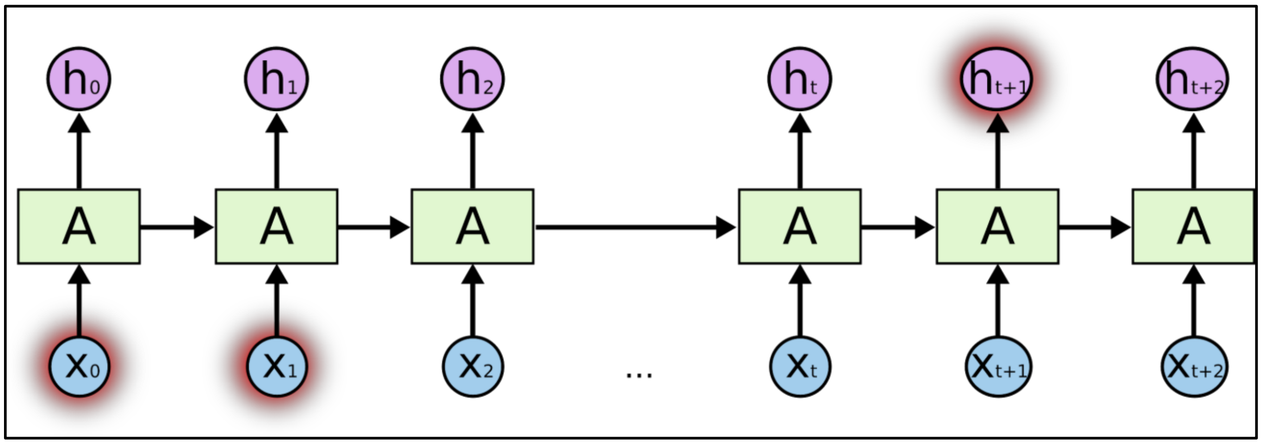


Figure 3.2. Long-term dependency issue in RNNs

Essential to these successes is the use of "LSTMs," a very special kind of recurrent neural network which works, for many tasks, much much better than the standard version.

Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

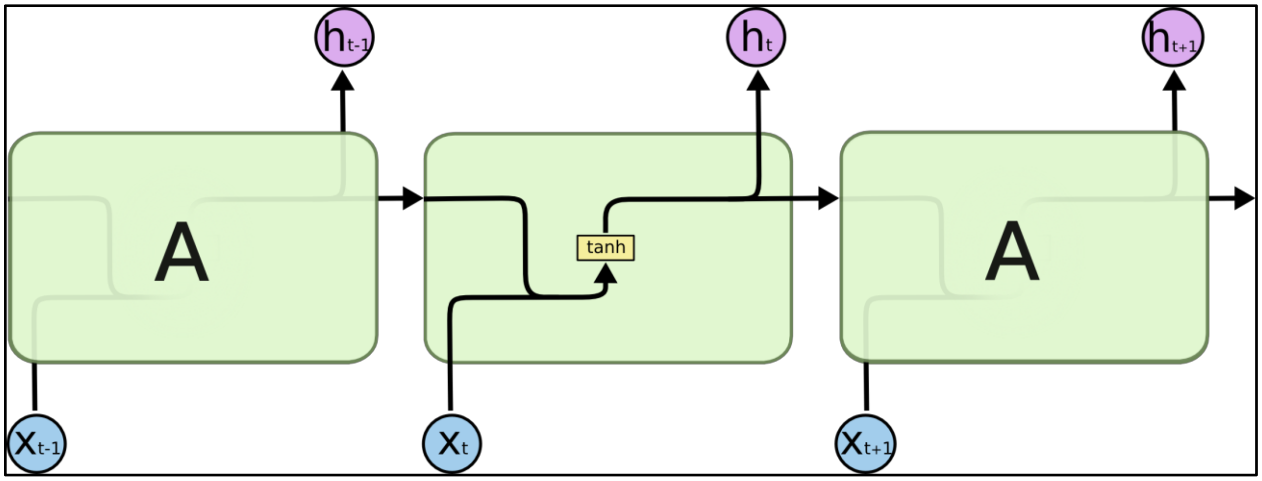


Figure 3.3. The repeating module in a standard RNN contains a single layer.

LSTMs also have this chain-like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

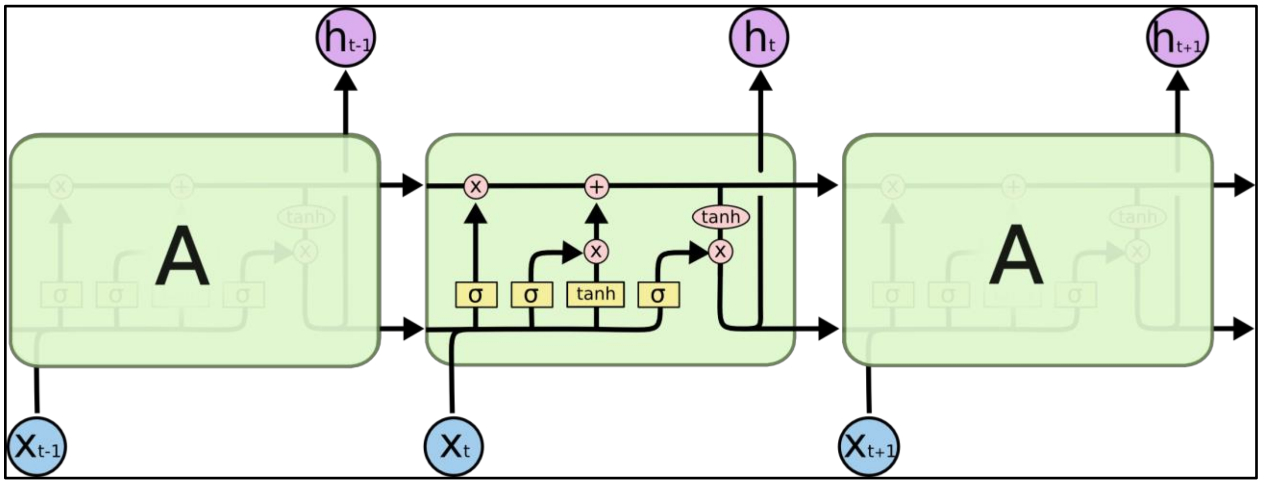


Figure 3.4. The repeating module in an LSTM contains four interacting layers.

The Long short-term memory (LSTM) is made up of a memory cell, an input gate, an output gate and a forget gate. The memory cell is responsible for remembering the previous state while the gates are responsible for controlling the amount of memory to be exposed. The memory cell is responsible for keeping track of the dependencies between the elements in the input sequence. The present input and the previous is passed to the forget gate and the output of this forget gate is fed to the previous cell state. After that the output from the input gate is also fed to the previous cell state. By using this the output gate operates and will generate the output. The mathematics of the LSTM cell looks like this: Input First, the input is squashed between -1 and 1 using a tanh activation function. This can be expressed by:

$$g = tanh(b^g + x_t U^g + h_{t-1} V^g)$$

Where U^g and V^g are the weights for the input and previous cell output, respectively, and b^g is the input bias. Note that the exponents g are not a raised power, but rather signify that these are the input weights and bias values (as opposed to the input gate, forget gate, output gate etc.). This squashed input is then multiplied element-wise by the output of the input gate, which, as discussed above, is a series of sigmoid activated nodes:

$$i = \sigma(b^i + x_t U^i + h_{t-1} V^i)$$

The output of the input section of the LSTM cell is then given by:

$$g \circ i$$

Where the ∘ operator expresses element-wise multiplication. Forget gate and state loop The forget gate output is expressed as:

$$f = \sigma(b^f + x_t U^f + h_{t-1} V^f)$$

The output of the element-wise product of the previous state and the forget gate is expressed as st−1∘ f. The output from the forget gate / state loop stage is:

$$s_t = s_{t-1} \circ f + g \circ i$$

Output gate The output gate is expressed as:

$$o = \sigma(b^o + x_t U^o + h_{t-1} V^o)$$

So the final output of the cell , with the tanh squashing, can be shown as:

$$h_t = tan\,h(s_t) \circ o$$

3.2. Implementation Details:

3.2.1. Methodology:

The architecture of proposed methodology is shown in Figure 5 as follows and mainly consists of three modules such as data processing, data modeling, and classification modules.

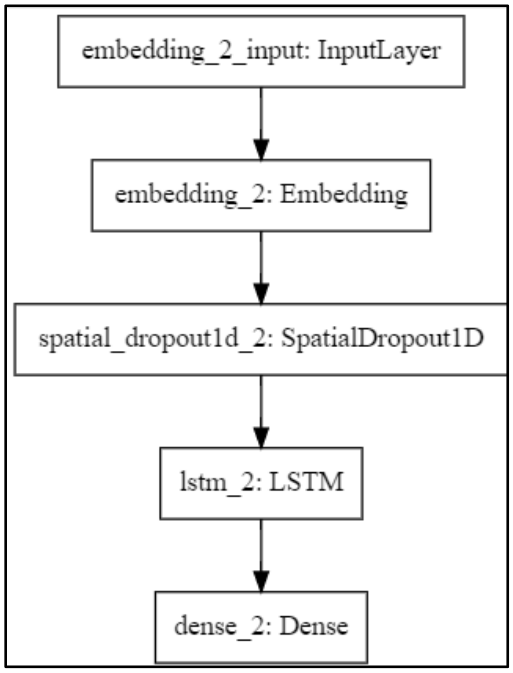

embedding_2_input: InputLayer
embedding_2: Embedding
spatial_dropout1d_2: SpatialDropout1D
lstm_2: LSTM
dense_2: Dense

Figure 3.5. Model Structure.

- Word Embedding Layer:

  It is the collective name for a set of language modeling techniques where words or phrases from the vocabulary are mapped to vectors of real numbers. This feature learning technique in natural language processing (NLP) is called word embedding. We have used word embedding and created a vocabulary of more than 3,00,000 unique words. The collection of word embedding is used to convert the text into vectors of 300 size also called “embed size”. These vectors are then fed to the neural network model, as the neural network can only process numeric information and not natural languages.

- Spatial Dropout Layer:

  The SpatialDropout layer is added to prevent overfitting. The reason behind this is it drops the entire embedding channels while the normal Keras embedding dropout drops all channels for entire words, and sometimes losing one or more words can alter the meaning completely.

- LSTM Layer:

  The LSTM layer takes in input as a 3D array and returns a 3D array. Here, the actual meaning of the output is generated before it is further processed.

- Dense Layer:

  Dense layer provides shape to the output received from the LSTM layer.

3.2.2. Details about the algorithm and packages:

The dataset derived from Kaggle is a structured dataset which has 40k tweets. The first step involves creating a Keras model with the Sequential() constructor. The first layer in the network, as per the architecture diagram shown previously, is a word embedding layer. This will convert our words (referenced by integers in the data) into meaningful embedding vectors. This Embedding() layer takes the size of the vocabulary as its first argument, then the size of the resultant embedding vector that you want as the next argument.

The next layer in our Keras LSTM network is a spatial dropout layer to prevent over fitting. The next layer is the LSTM layer. To specify an LSTM layer, first you have to provide the number of nodes in the hidden layers within the LSTM cell, e.g. the number of cells in the forget gate layer, the tanh squashing input layer and so on. After that, there is a Dense layer for use in recurrent neural networks to produce a standard output. This function adds an independent layer for each time step in the recurrent model. The activation for these dense layers is set to be softmax in the final layer of our Keras LSTM model as shown in Figure 3.6.

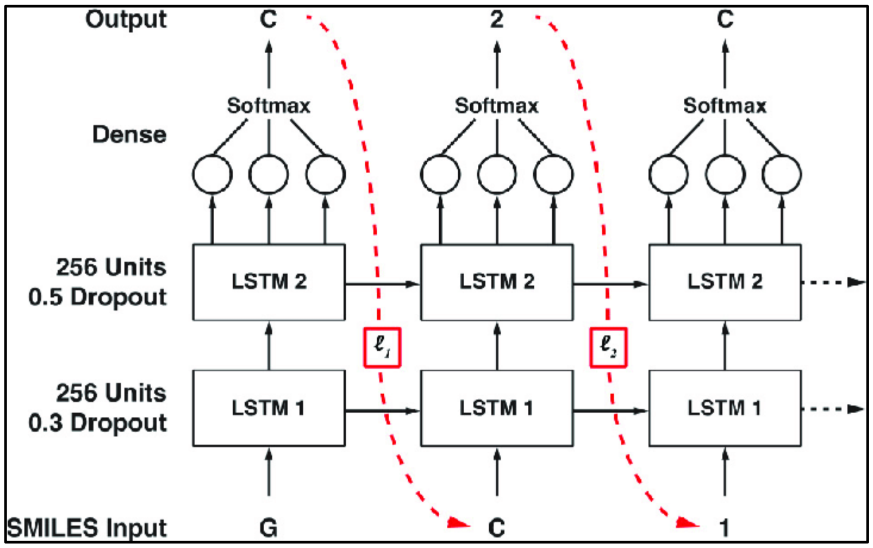
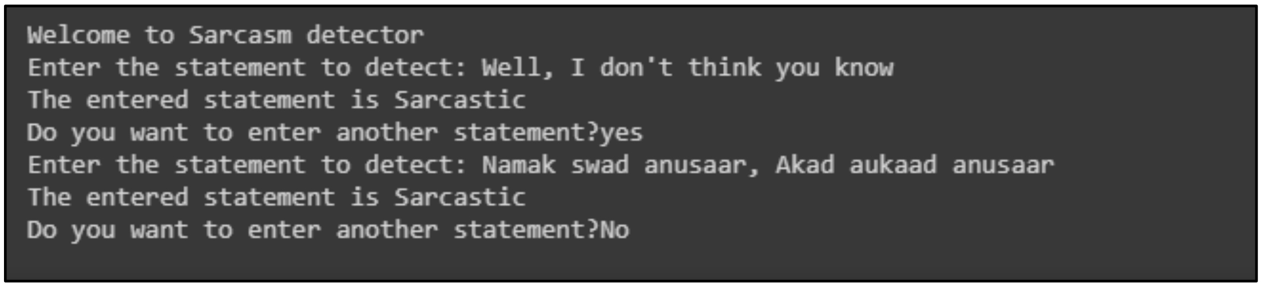


Figure 3.6. Detailed LSTM structure

Chapter 4
PROJECT IMPLEMENTATION DETAILS

4.1. Screenshots:



```
Welcome to Sarcasm detector
Enter the statement to detect: Well, I don't think you know
The entered statement is Sarcastic
Do you want to enter another statement?yes
Enter the statement to detect: Namak swad anusaar, Akad aukaad anusaar
The entered statement is Sarcastic
Do you want to enter another statement?No
```

Figure 4.1. Implementation screenshots

Figure 4.1 demonstrates the use of the sarcasm detector on the given statements. The statement "Well, I don't think you know" here has been correctly classified as "Sarcastic". Further, a regional hinglish statement is given as an input since the dataset included few regional statements included and the model has been trained according to it. This gave the ability to correctly classify the statement as "Sarcastic".

4.2. Evaluation Parameters Details

Using such a huge amount of data we trained our model on 40k total sarcastic and non-sarcastic tweets put together, the model performed really well. 55% of those were identified as Sarcastic and 45% were identified as Non-Sarcastic tweets as shown in Figure 4.2.

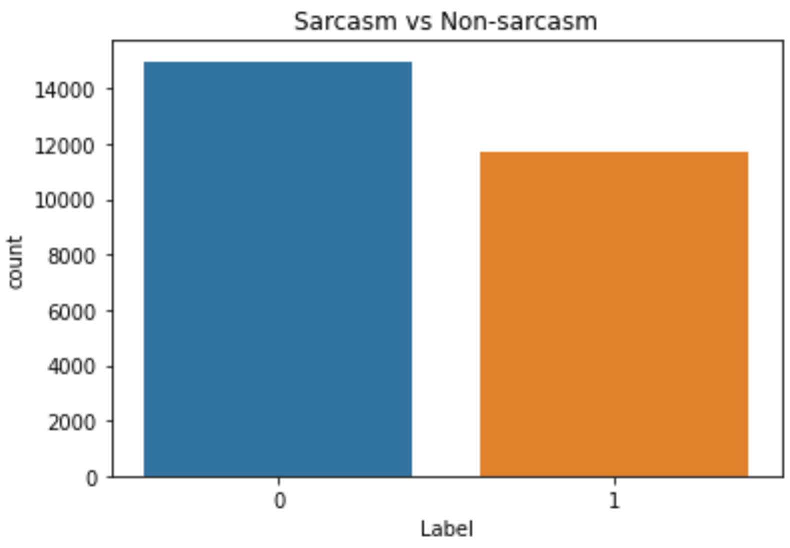


Figure 4.2. Sarcastic vs Non-Sarcastic tweets

During testing the model with 25 epochs gave 95% accuracy. The model was then validated on a group of 1500 tweets and the score and accuracy obtained was 0.98 and 88% respectively.

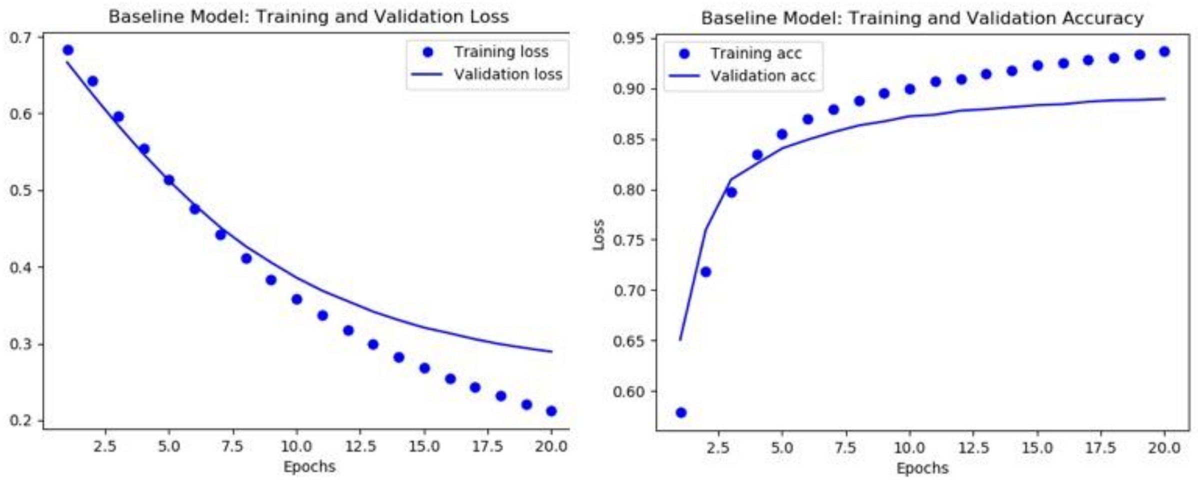


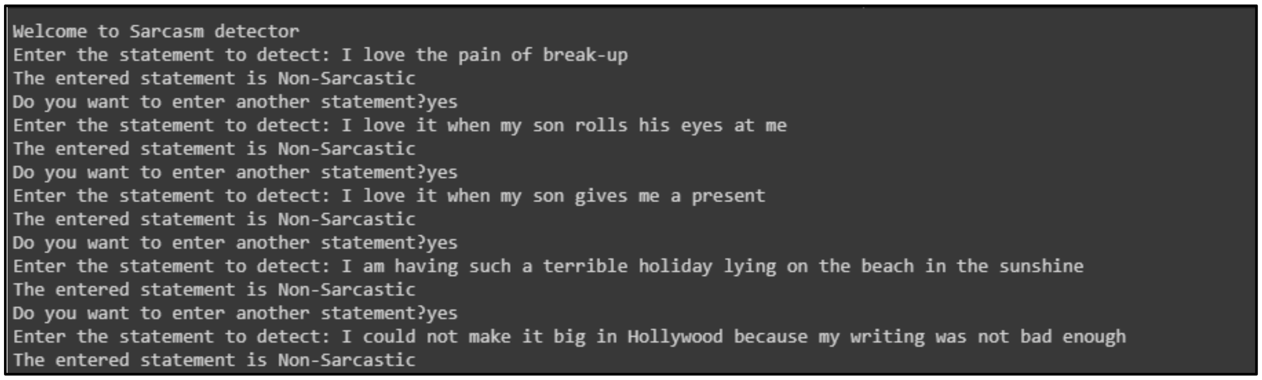
Figure 4.3. Model loss and accuracy

According to Figure 4.3 it is clear that the model gives 88% accuracy on validation, and that is better than all the discrete model algorithms used for sarcasm detection. As we train this model on huge datasets more accurate the predictions will be, as the accuracy increases.

4.3. Limitations

To understand and detect sarcasm it is important to understand the facts related to an event. This allows for detection of contradiction between the objective polarity (usually negative) and the sarcastic characteristics conveyed by the author (usually positive).Consider the example, “I love the pain of breakup”, it is difficult to extract the knowledge needed to detect if there is sarcasm in this statement. In the example, “I love the pain” provides knowledge of the sentiment expressed by the author (in this case positive), and “breakup” describes a contradicting sentiment (that of negative).

Although hashtag-based labeling can provide large-scale supervision, the quality of the dataset may be dubious. This is particularly true in the case of using #not to indicate insincere sentiment. show that #not is the one used to express sarcasm - while the rest of the sentence is not sufficient for identifying the sarcasm. For example, ‘Looking forward to going back to school tomorrow #not’. The speaker expresses sarcasm through #not. In most reported works that use hashtag-based supervision, the hashtag is removed in the pre-processing step. This reduces the sentence above to ‘Looking forward to going back to school tomorrow’ - which may not have a sarcastic interpretation, unless the author’s context is incorporated. Thus, hashtag-based supervision may cause ambiguities(or be incorrect) in some cases. To mitigate this problem, a new trend is to validate on multiple datasets - some annotated manually while others annotated through hashtags train their deep learning-based model using a large dataset of hashtag-annotated tweets, but use a test set of manually annotated tweets.



```
Welcome to Sarcasm detector
Enter the statement to detect: I love the pain of break-up
The entered statement is Non-Sarcastic
Do you want to enter another statement?yes
Enter the statement to detect: I love it when my son rolls his eyes at me
The entered statement is Non-Sarcastic
Do you want to enter another statement?yes
Enter the statement to detect: I love it when my son gives me a present
The entered statement is Non-Sarcastic
Do you want to enter another statement?yes
Enter the statement to detect: I am having such a terrible holiday lying on the beach in the sunshine
The entered statement is Non-Sarcastic
Do you want to enter another statement?yes
Enter the statement to detect: I could not make it big in Hollywood because my writing was not bad enough
The entered statement is Non-Sarcastic
```

Figure 4.4. Example of misclassification

Chapter 5
CONCLUSION AND FUTURE SCOPE

5.1. Conclusion

Sarcasm detection is a really fascinating subject. It evaluates diverse feature types for sentiment extraction including sentiments, words, patterns and n-grams, confirming that each feature type contributes to the sentiment classification framework. As we have seen that it is feasible to do sarcasm detection using NLP tools, one quick and easy way to improve this detector is to use a spell corrector along with, for the tweets. This would help in minimizing the order of dimensions of the dictionary for the n-gram features and will improve the sentiment analysis operation as well. In the future, these methods can be applied for automated clustering of sentiment types and sentiment dependency rules and can be expanded to detect some other non-literal form of sentiments like humor.

We have successfully implemented a sarcasm detector with a 40k dataset using RNN-LSTM with an accuracy of 95%. This shows that the sets are well annotated and reasonably representative for sarcasm detection in tweets.

5.2. Future Scope

a. Emoticon-based features : These features includes positive emoticon, negative emoticon, contrast between word, i.e, a boolean feature that will be one if both positive and negative words are present in the tweet, contrast between emoji, i.e, it will take the value as one when either positive word and negative emoji is present or negative word and positive emoji is present in the tweet.
b. Punctuation-based features : These features include number of exclamation marks, number of dots, number of question marks, number of capital letter words, number of single quotations.

REFERENCES

[1] O. Tsur, D. Davidov, and A. Rappoport. Icwsm-a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews. In ICWSM, 2010.

[2] R. Gonza´lez-Iba´n˜ez, S. Muresan, and N. Wacholder. Identifying sarcasm in twitter: A closer look. In ACL (Short Papers), pages 581–586. Citeseer, 2011.

[3] Liu, Pengfei, Xipeng Qiu, and Xuanjing Huang. "Recurrent neural network for text classification with multi-task learning." *arXiv preprint arXiv:1605.05101* (2016).

[4] Yao, Lirong, and Yazhuo Guan. "An Improved LSTM Structure for Natural Language Processing." *2018 IEEE International Conference of Safety Produce Informatization (IICSPI)*. IEEE, 2018.

[5] Joshi, Aditya, Pushpak Bhattacharyya, and Mark J. Carman. "Automatic sarcasm detection: A survey." ACM Computing Surveys (CSUR) 50.5 (2017): 1-22.

[6]Ghosh, Aniruddha, and Tony Veale. "Fracking sarcasm using neural network." *Proceedings of the 7th workshop on computational approaches to subjectivity, sentiment and social media analysis*. 2016.

[7] Porwal, Saurabh, et al. "Sarcasm Detection Using Recurrent Neural Network." *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*. IEEE, 2018.

[8] Manoj Y. Manohar, PallaviKulkarni, "Improvement Sarcasm Analysis using NLP and Corpus based Approach", International Conference on Intelligence Computing and Control Systems (ICICCS), IEEE, 2017.

[9] Paras Dharwal, Tanupriya Choudhury, Rajat Mittal, Praveen Kumar, “Automatic Sarcasm Detection using Feature Selection”, International Conference on Applied and Theoretical Computing and Communication Technology, IEEE, 2017.

[10] Po-Ya Angela Wang. 2013. #Irony or #Sarcasm A quantitative and qualitative Study Based on Twitter. In 27th Pacific Asia Conference on Language, Information, and Computation. 349–256.

[11] Emilio Sulis, Delia Far´ıas, Delia Irazu Hernandez, Paolo Rosso, Viviana Patti, and Giancarlo Ruffo. 2016. Figurative ´ messages and affect in Twitter: Differences between #irony, #sarcasm and #not. Knowledge-Based Systems 108 (2016), 132 – 143. New Avenues in Knowledge Bases for Natural Language Processing.