To laugh or not to laugh – LSTM based humor detection approach

Krupa Patel
Department of Information Technology
Thadomal Shahani Engineering College
Mumbai, India
krupap1602@gmail.com

Manasi Mathkar
Department of Information Technology
Thadomal Shahani Engineering College
Mumbai, India
manasigo2000@gmail.com

Sarjak Maniar
Department of Information Technology
Thadomal Shahani Engineering College
Mumbai, India
maniarsarjak@gmail.com

Avi Mehta
Department of Information Technology
Thadomal Shahani Engineering College
Mumbai, India
avi.mehta90.am@gmail.com

Prof. Shachi Natu
Department of Information Technology
Thadomal Shahani Engineering College
Mumbai, India
Shachi_natu@yahoo.com

2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT) | 978-1-7281-8595-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCCNT51525.2021.9580124

***Abstract*—Humor holds the power to turn any mundane conversation into something more enthralling. It is an important feature of personal communication. Sentiment Analysis helps people as well as corporations to comprehend the attitudes of people towards certain words based on how they are strung together. Particularly, Humor Detection facilitates the understanding of underlying relations of words and how various combinations of strings can invoke laughter. In this paper, an approach to identify the presence of humor in sentences, using a model based on Long Short-Term Memory (LSTM), is adopted. LSTM is a specialized version of Recurrent Neural Network (RNN). While conventional RNNs involve cyclic connections for modeling sequenced data, LSTM uses additional units for retaining information for comparatively long periods. A wide-ranging, evenly distributed dataset is used in the implementation of the model. After preprocessing, the data progresses along the embedding layer, LSTM layer and the dense layer and the outcome is finally compiled. Ultimately, the model states whether a given statement is humorous or not. The proposed LSTM model gives an accuracy of 94.62%. As new sources of obtaining humorous content keep emerging through multimedia, new patterns can be discovered. Understanding such patterns through such existing Humor Detection models can promote the development of Automated Humor Generation Systems.**

Keywords— Humor Detection, Sentiment Analysis, Long Short-Term Memory, Recurrent Neural Network

I. Introduction

Humor is the quality of being amusing or hilarious in a way that invokes laughter. It is inarguably vital in communication as it adds a part of friendliness to daily interactions. It helps people to connect with each other. Furthermore, scientifically important information related to linguistic, psychological, neurological, and sociological phenomena of beings can also be studied based on how different individuals perceive various forms of humor. Recognizing humor can help in understanding how the human mind cogitates.

The general structure of a humorous joke comprises a punchline and contextual information that leads up to that punchline. Though the pith of a joke lies within the punchline, it would be difficult to interpret a joke with insufficient context. Recognizing the presence of a punchline and distinguishing it from the remaining context is a complicated process. Hence, owing to its complex nature, understanding humor can be quite perplexing and challenging. Humor being subjective is one of the greatest challenges of humor recognition. Thus, challenges arise as the rules or reasons that can help in determining whether something should be regarded as humorous or not, are obscure.

Nevertheless, when this part of daily communication is introduced to machines, humor undoubtedly becomes less intimidating to understand. With the advent of deep learning and sentiment analysis, one can rigorously train and deploy various models that allow feature selection and enhance the process of humor recognition. This paper proposes a Recurrent Neural Networks (RNN) based model - Long Short-Term Memory (LSTM). RNNs prevail in the field of Natural Language Processing (NLP) as they allow previous outputs to be used again as inputs. They involve cyclic connections, making them more effective in modelling sequence data in comparison to feedforward neural networks [1]. The model predicts if a given statement is humorous or Nonhumorous utilizing LSTM Networks, a special variation of RNN. In addition to the standard units found in RNNs, LSTM uses special units for maintaining information for longer periods. The LSTM RNN model has shown considerable advancements over conventional RNN Language Modelling (LM) [2], particularly for learning context-free as well as context-sensitive languages [3].

This paper presents various sections that help in understanding the development of the proposed LSTM based Humor Detection model. Section II, Literature Review, unfolds the literature that was studied to gain a deeper understanding on relevant topics. The literature review is followed by Section III, Data, that expounds on the utilized dataset and its parts. Furthermore, Section IV, Proposed Approach, explains the proposed approach involving two subsections that present the methodology and models used. The successive section, Section V, Experimental Setup and Result Analysis, shows the experiments carried out to understand the model further and the analysis of the model. Finally, the Conclusion encloses the essence of the paper and presents the future scope.

II. Literature Review

In the related work of humor identification, there has been an infinite number of methods studied over time: LSTM and TF-IDF (term frequency-inverse document frequency) Neural Network System [4]. A two-constituent neural

network system was built. The first stage consisted of combinations of bidirectional LSTM and TF-IDF and the variance difference between the actual headline target words and the subtraction of two vectors. In this approach, a model was built using a blend of LSTM and TF-IDF and after that a feed-forward based neural network was used. The second stage was a two-layer feed forward network. A comparison between TF-IDF system with and TF-IDF with subtraction was done and the system with subtraction generated better results.

In the research of Determining sarcasm using sentiwordnet [5], the scores were determined which was based upon the polarities of sentimental words using Sentiwordnet which was imported from nltk. The polarity and subjectivity of the sentences were created using the Textblob and the scores were created from the tool which was sarcasm detector. Luke and Alfredo [6] applied RNN on the reviews from Yelp dataset to detect humor. They applied Convolutional Neural Networks (CNNs) to train a model and to conclude their work, the model had more accurate humor recognition.

Ashwin Rajadesingan [7] applied three machine learning classifiers – Decision tree, Support Vector Machine (SVM), Logistic Regression, by considering the behavioral modelling-based approach. In the work by Ellen Riloff [8] humor was determining positive sentiment phrases and negative sentiment phrases. In their research paper, they enforced a Bootstrapping algorithm to recognize the different phrases. The phrases considered were limited to specific syntactic structures. In the work by Engelthaler and Hills [9], in the word level, there was a dataset of 4997 words. Humor norms and rating were analyzed based on several factors that influence the judgments. In 2019, built on this theory, Westbury, and Hollis [10] examined the semantic, orthography, and frequency of words to predict the humor-rating originally and the result of previous words which were unsettled. They illustrated those texts which are less common and have dubious orthographical structure are judged funnier.

Conventional RNNs only possess hidden states that provide the memory for RNNs. On the other hand, the LSTM model comprises cell states in addition to hidden states. The cell state, regulated by "gates", is capable and efficient of removing or adding information to the cell. As a result, a LSTM model is proposed that is efficient and capable of handling long-term dependencies, in contrast to traditional RNNs.

III. Data

Most pre-existing datasets amalgamate informal and formal jokes with incompatible parameters, which can make it easy for simpler models to detect humor without really comprehending untapped relations. Another notable problem in such datasets is their relatively small text length, which leads to overfitting. Consequently, a dataset that makes it difficult for very basic models to give seemingly perfect accuracy was needed. A number of datasets were available for the purpose of humor detection and their sizes are compared as shown in Table I [11]. Nevertheless, the ColBERT dataset [11], available on Kaggle was incorporated as it is uniformly distributed and encompasses a broad range of content. It consists of about 200k short texts combined from various sources. It comprises humorous (positive) and Nonhumorous (negative) samples in equal numbers, i.e., 100k samples of each as shown in Fig. 1.

TABLE I. Datasets for humor detection

Dataset	Division	
	#Positive Samples	***#Negative Samples***
16000 One-Liners [12]	16,000	16,002
PTT Jokes [13]	1,425	2,551
Pun of the Day [14]	2,423	2,403
English-Hindi [15]	1,755	1,698
Used Dataset	100,000	100,000

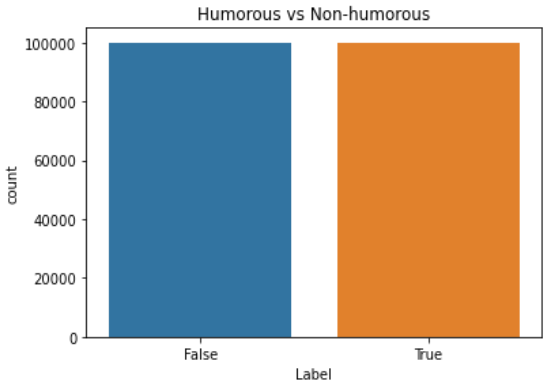

Fig. 1. Data Classification

A. Composition of Data

The composition of the data set can be attributed to two datasets containing formal texts. Out of them one contained humorous texts and one did not contain humorous texts. They were processed into syntactically similar datasets to meet the criteria as mentioned previously. A News Dataset was provided by Huffington Post which had 200,853 headlines (including categories, stories, and links) from 2012 to 2018 [11]. Headlines are jumbled in a variety of news categories counting politics, health, parenthood, and entertainment. The Jokes dataset is made up of 231,657 humorous short jokes that were gathered from Reddit groups [11].

B. Processed Data

The duplicate texts in both cases were dropped. Consequently, 1369 and 1558 rows were removed from the Jokes dataset and the News dataset, respectively. Additional cuts were made to keep the range of characters between 30 and 100, and the range of words between 10 and 18 [11]. Moreover, as the data in the News dataset used Title Case formatting, on contrary to the Sentence Case formatting of the Jokes dataset, the data was made uniform by converting the Title Case of the news dataset to Sentence Case by retaining the foremost character of the sentences as a capital letter and setting the rest to lower case. After processing, the

creators of the dataset merged 100k columns of each dataset randomly, resulting in a statistically similar and evenly distributed dataset.

C. Data Summary

In the resulting dataset, texts are uniformly distributed between humorous and Nonhumorous texts. The dataset comprises 200k labeled short texts with similar syntactic features in total. It is comparatively wider ranging than the previous datasets shown in Table I. Table II gives a detailed summary of the basic statistics of the dataset [11].

TABLE II. STATISTICAL SUMMARY OF DATA

	#Chars	#Words	#Unique words	#Punct-uation	#Duplicate words	#Senten-ces
mean	71.561	12.811	12.371	2.378	0.440	1.180
std	12.305	2.307	2.134	1.941	0.794	0.448
min	36	10	3	0	0	1
median	71	12	12	2	0	1
max	99	22	22	37	13	2

IV. PROPOSED APPROACH

This section scrutinizes the proposed method for humor recognition. From a technical perspective, this paper proposes an LSTM RNN model that supersedes traditional RNNs in classifying texts as humorous and Nonhumorous. Fig. 2 depicts the workflow of this approach. Besides data preprocessing i.e., to convert the words into lower case and removing special characters, the main significance lies in feeding the padded input into the LSTM model. Detailed descriptions about the LSTM model and various layers in keras which are used in this model are explained in the following subsections.

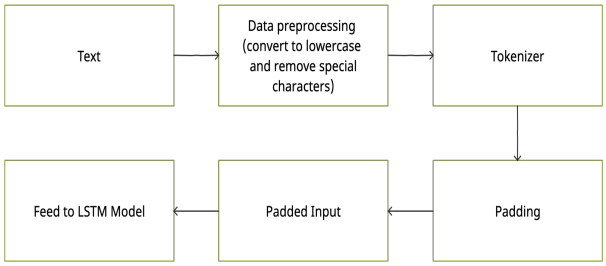

Fig.2. Input Flow Diagram

A. Methodology

Long Short-Term Memory, referred to as "LSTM" was first introduced by Hochreiter and Schmidhuber. The modified version of Recurrent Neural Network is LSTM. Instead of having a standard feed forward network, LSTM consists of feedback connections. LSTM can not only process single data points but additionally process entire sequences of information. While the functionality of LSTM is analogous to RNN, the distinction lies in its gating mechanism. The drawback of short-term memory in RNN is solved by LSTM. The gradients of the loss function approaches to zero when more layers with specific activation functions are added to neural networks, making the network difficult to train. LSTM was developed to resolve this vanishing gradient problem that can be faced while working with RNN.

Fig. 3 to 6 represents the components of LSTM Cell.
C_t represents memory cell state at time t,
C_{t-1} is the previous state output,
f_t represents equation of forget gate at time t,
h_t represents equation of next memory cell at time t,
h_{t-1} indicates previous output,
x_t is input at time t ,
w and b represent the equivalent weights and bias of respective gates,
σ is the sigmoid activation function.

A LSTM unit consists of a memory cell state, forget gate, an input gate, and an output gate.

- Memory Cell State: In Fig.3 the memory cell is used for recalling and forgetting the information over a period. The information which the cell must remember is based on the context of the input. As the context changes it remembers some of the previous information and adds the next information to the cell.

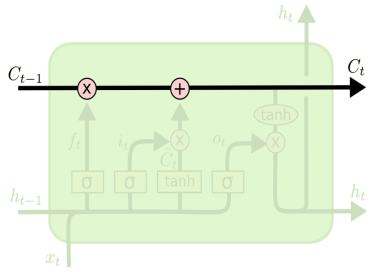

Fig.3. Memory cell state

The equation of memory cell state is:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (1)$$

- Input Gate: In Fig.4 the input gate adds new information to the cell state. The new information which is additionally aggregated is done in three major steps:
1) To understand what values, need to be added to the cell state which requires a sigmoid function. The sigmoid function restricts the output data from 0 to 1. The value 0 resembles the information that does not get passed through the next cell while 1 indicates information will pass through the next cell.
2) Generate a vector which consists of all possible values that can be added to the cell state. Using the tanh function to determine which values need to get aggregated as it ranges the value of output from -1 to +1.
3) Multiplying the value of the regulatory filter to the created vector and then combining this useful information to the cell state by making use of additional operation.

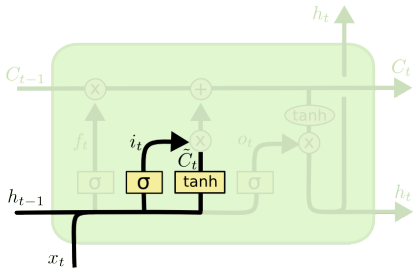

Fig.4. Input gate

The equation of input gate is given by:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (2)$$

$$\tilde{C}_t = \tanh(W_c.[h_{t-1}, x_t] + b_c) \quad (3)$$

- Output Gate: In Fig.5 selecting helpful and useful information from the current cell state and exhibiting it out as an output is done with the help of output gate.

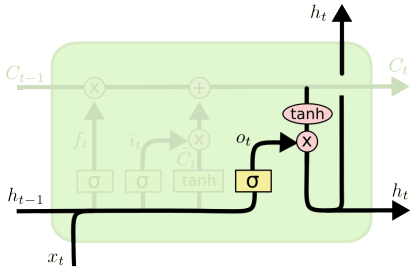

Fig.5. Output gate

The equation of output layer is given by:

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \quad (4)$$

$$h_t = o_t * \tanh(C_t) \quad (5)$$

- Forget Gate: In Fig.6 a forget gate is accountable for removing information from the cell state, which is not needed, or which is not necessary for LSTM. This is done by multiplication of a filter. This in turn optimizes the overall performance of the LSTM network.

Here the concatenation is given by W_f multiplied by h_{t-1} and x_t. The entire function passes through a sigmoid activation function which transforms the input between 0 to 1.

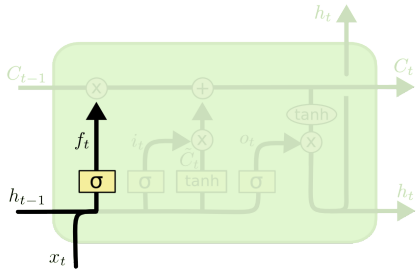

Fig.6. Forget gate

The equation of forget gate is given by:

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \quad (6)$$

B. Models

Sequential

A Sequential model means a stack of layers into the keras model. This model provides training and inference features on the keras model. A Sequential model is only valid for a clear stack of layers where each layer has one input tensor and one output tensor.

Embedding Layer

The Embedding layer in keras is used to process texts in neural networks. It is necessary that the input provided to embedding layer must be an integer hence, each word in the training set is represented by a unique integer. This layer is initialized with weights and will learn an embedding for all the words in the training set. The word embeddings are utilized in another model later. The sentence input is converted to a matrix to pass it to the next layer. Parameters used in the model for embedding:

1. input_dim (input dimensions): It is the size of the vocabulary, i.e., maximum integer index (here 10000).
2. output_dim (output dimensions): Dimension of the dense embedding.

LSTM Layer

There are a few parameters in the LSTM layer provided by keras:

- embed_dim: In the LSTM model, the embedding layer converts the input sequence into a sequence of vectors of dimension embed_dim (here 10).
- lstm_out: The LSTM converts the vector sequence into a single vector with the size of lstm_out, containing the entire sequence information (here 64).

Dense Layer

Dense Layer is regarded as the regular deeply connected neural network layer. It is the most often utilized and, in fact, is one of the most common layers. The following operation is performed on the input by the dense layer, and the output is returned. The output provided by the LSTM layer is the input for the Dense layer in the model which is (None, 64) the output will also be in the form (None, 64) but since we have mentioned the number of neurons in the dense layer which is 2 the output will be changed to (None, 2) as there are only 2 classes in the output set. All the layers in the proposed LSTM model are shown in Fig.7. Activation function used inside the Dense layer is SoftMax which converts a vector of values, here (None, 64), to a probability distribution. The components of the output vector are in range (0, 1) and they sum up to 1. As the result is interpreted as a probability distribution, this function is usually applied in the activation of the last layer of a classification network.

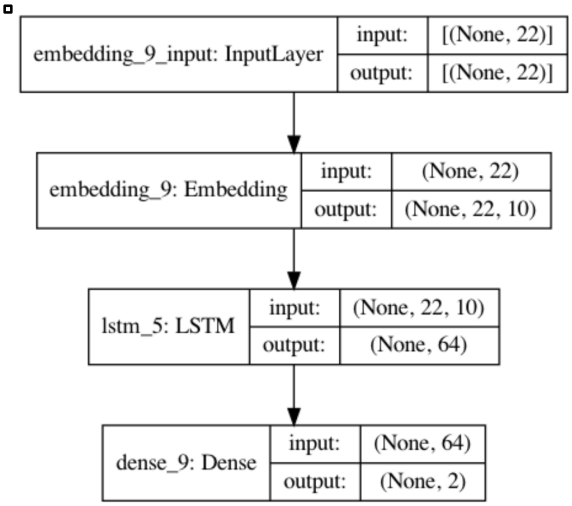

Fig.7. Different Layers in LSTM model

Compilation

In deep learning, Loss function is used to find error while training. Optimization is an important process which helps is increasing the accuracy of the model so various types of optimizers are used, and Metrics is employed to gauge the performance of the model. In the proposed model, "categorical_crossentropy" loss function is used with "adam" optimizer and "accuracy" metrics.

V. Experimental Setup and Result Analysis

In this case, a few experimental changes were performed with respect to the values of the layer/input parameters to obtain best possible accuracy and the time taken to execute was also less. Hence, before performing tokenization, the vocab size i.e., the max features was initially set to 2000. Moreover, while considering factors pertaining to selecting the number of embedded dimensions, activation function and LSTM outputs, we selected some arbitrary values and got a decent accuracy. We noticed that the best accuracy is obtained when the SoftMax activation function is used with 10 embedded dimensions and 64 LSTM outputs, when a vocab size of 10000 is considered.

To get a better sense of the strength of the proposed approach, the model was manually inspected by assembling a text in a list which is in string format taken as an input. Before feeding this text as an input to the model, tokenization is performed on this input text. The purpose of tokenization is to convert the text sentences into separate words which will in turn convert word tokens into numerical format. After obtaining the numerical values (vectors) of the text sequence, padding is initiated, and the max length of each input text sentence is kept 29. The need for padding is to encode sequence data into contiguous batches to make all sequences in a batch fit a given standard length. So, it is essential to pad or truncate some sequences. Therefore value 0 will be added before the number values and this technique is called pre-padding. Now that the input text is all set to go into the model for predicting the output, the result of model prediction is stored in a variable humorDetector.

Let us consider the example of a text sentence: text = ["Empty mind is devil's workshop"]. After model prediction and storing the value in a variable humorDetector, the output will be displayed as [96.215266 3.784736]. There are two numeric values and the type of this humorDetector variable is 'numpy.ndarray'. The value at index 0 is the percentage of Nonhumor in the input text that our model finds, and the value at index 1 is the percentage of Humor in the input text that our model finds. Here, in our example of a text sentence which is "Empty mind is devil's workshop", in the output of the humorDetector variable, the value at index 0 is 96.215266 and the value at index 1 is 3.784736. This means that the generated model finds the given input text 96.215266% Nonhumorous and only 3.784736% Humorous which is a pretty good identification as the input text is just a quote and there is no humor involved in this statement.

The last final step is condition checking. In this step, the NumPy package's argmax method is used to determine the max value among the two values which are obtained from the humorDetector variable which is the result of the model prediction. Further, it is checked if the max value belongs to index 0 or 1 and with that the result is printed as Humorous or Nonhumorous. In the example taken, max value 96.215266 is at index location 0, so according to the condition it matches with 0 and with that, the desired output is obtained as Nonhumorous which is correct for the input text sentence example. After having tested with numerous humorous and Nonhumorous texts, the accuracy of the generated model is exemplary. The model provides accuracy for the train set and validation set which is shown in Fig.8 and Fig.9, respectively.

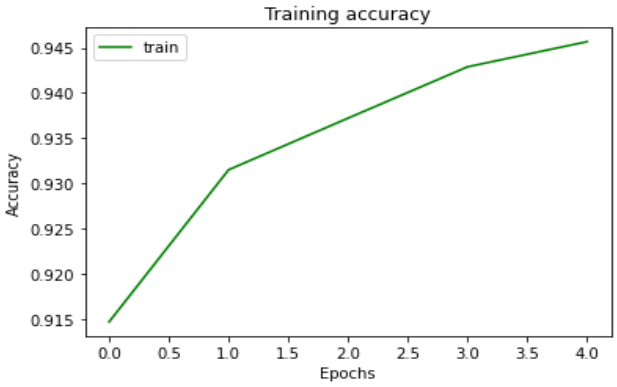

Fig.8. Model Accuracy (Train set)

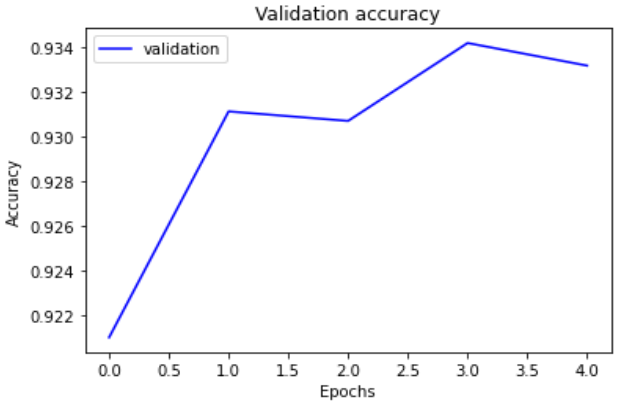

Fig. 9. Model Accuracy (Validation set)

```
text = ["Honesty is the best policy"]
text = tokenizer.texts_to_sequences(text)
text = pad_sequences(text, maxlen=29, dtype='int32', value=0)
sentiment = model.predict(text,batch_size=1,verbose = 2)[0]
print(sentiment*100)
if(np.argmax(sentiment) == 0):
    print("Non-Humorous")
elif (np.argmax(sentiment) == 1):
    print("Humorous")
print(np.argmax(sentiment))
```

```
1/1 - 0s
[79.05673 20.94327]
Non-Humorous
0
```

Fig. 10. Model Outputs – Non-Humorous

```
text = ["""Good girls are found on every corner of the earth,
        but unfortunately the earth is round"""]
text = tokenizer.texts_to_sequences(text)
text = pad_sequences(text, maxlen=29, dtype='int32', value=0)
sentiment = model.predict(text,batch_size=1,verbose = 2)[0]
print(sentiment*100)
if(np.argmax(sentiment) == 0):
    print("Non-Humorous")
elif (np.argmax(sentiment) == 1):
    print("Humorous")
print(np.argmax(sentiment))
```

```
1/1 - 0s
[2.6812317e-02 9.9973183e+01]
Humorous
1
```

Fig. 11. Model Outputs - Humorous

Table III shows some of the example text input sentences and its output. The model performs well and detects correctly whether the given input text is Humorous or Non-Humorous.

TABLE III. INPUT TEXT AND OUTPUT

Input Text	Output
“Empty mind is devil's workshop”	Nonhumorous
“Good girls are found on every corner of the earth, but unfortunately the earth is round”	Humorous
“If we shouldn’t eat at night, why do they put a light in the fridge?”	Humorous
“Honesty is the best policy”	Nonhumorous
“Coffee spelled backwards is eeffoc which is as meaningless as you are”	Humorous

Table IV represents how the same dataset when applied to other traditional models such as decision tree, Support Vector Machine (SVM) and Multinomial Naive Bayes gave accuracy less than 90%. It also helps in depicting how the LSTM model performed better than XGBoost and XLNet [11].

TABLE IV. COMPARISON WITH OTHER MODELS

Dataset	Configuration	Accuracy
Decision Tree		78.6%
SVM	sigmoid, gamma=1.0	87.2%
Multinomial NB	alpha=0.2	87.6%
XGBoost		72%
XLNet	XLNet-Large-Cased	91.6%
Proposed Model		94.62%

VI. CONCLUSION

Humor is the quality of evoking laughter by means of amusing linguistic devices. In this paper, a model for humor recognition by virtue of a deep learning LSTM RNN architecture is proposed. The proposed architecture enables us to discern humorous and nonhumorous texts with the help of positive and negative samples stemming from a wide-ranging dataset. The LSTM RNN Model outperforms traditional RNN models as it could process entire sequences of information in addition to single data points. This model gives an accuracy of 94.62%. In spite of its exemplary accuracy, one limitation of the LSTM model could be that it has a lesser accuracy compared to the ColBERT model, having an accuracy of 98.2%. However, the Colbert model is relatively more time consuming as it takes approximately 2 hours in average for the training of a single epoch. Another limitation could be that the proposed model can only identify humor for English texts as input. Ultimately, the deep learning model enables the selection of semantic attributes for humor recognition, superseding the need for human intervention. With the rapid emergence of new multimedia content, new patterns can be explored. Coupling such new-found patterns with advancing humor detection models, it could be possible to create models for the automatic generation of humorous texts in the future.

REFERENCES

[1] Sundermeyer, Martin, Ralf Schlüter, and Hermann Ney. "LSTM neural networks for language modeling." *Thirteenth annual conference of the international speech communication association*. 2012.

[2] M. Sundermeyer, R. Schluter, and H. Ney, “Lstm neural networks ¨ for language modeling.” in INTERSPEECH, 2012, pp. 194–197.

[3] F. A. Gers and J. Schmidhuber, “LSTM recurrent networks learn simple context free and context sensitive languages,” IEEE Transactions on Neural Networks, vol. 12, no. 6, pp. 1333–1340, 2001.

[4] Xuefeng Luo, Kuan Tang,"Humor Detection of Edited Headlines with LSTM and TFIDF Neural Network System",Linguistics Department, University of Tuebingen,Germany,Proceedings of the Fourteenth Workshop on Semantic Evaluation, December 2020

[5] Adarsh M J, Dr. Pushpa Ravikumar,"An Effective Approach for Sarcasm Detection in Text Data for Sentiment Analysis",International Journal of Engineering & Technology, Vol 7, No 4.39 (2018)

[6] Luke de Oliveira Alfredo Lainez Rodrigo "Humor Detection in Yelp reviews"

[7] Ashwin Rajadesingan, “Detecting Sarcasm on Twitter: A Behavior Modeling Approach”,ARIZONA STATE UNIVERSITY, December 2014

[8] Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, Ruihong Huang,"Sarcasm as Contrast between a Positive Sentiment and Negative Situation",in proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 704-714, 2013

[9] Tomas Engelthaler and Thomas T. Hills "Humor norms for 4,997 English words", 2017

[10] Geoff Hollis and Chris Westbury,“The principles of meaning: Extracting semantic dimensions from co-occurrence models of semantics”,Department of Psychology, University of Alberta

[11] Issa Annamoradnejad* and Gohar Zoghi,"ColBERT: Using BERT Sentence Embedding for Humor Detection"

[12] R. Mihalcea and C. Strapparava, “Making computers laugh: Investigations in automatic humor recognition,” in Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2005, pp. 531–538.

[13] P.-Y. Chen and V.-W. Soo, “Humor recognition using deep learning,” in Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), 2018, pp. 113–117.

[14] D. Yang, A. Lavie, C. Dyer, and E. Hovy, “Humor recognition and humor anchor extraction,” in Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, pp. 2367– 2376.

[15] J. Howard and S. Ruder, “Universal language model fine-tuning for text classification,” arXiv preprint arXiv:1801.06146, 2018.

[16] Annamoradnejad, Issa, and Gohar Zoghi. "Colbert: Using bert sentence embedding for humor detection." *arXiv preprint arXiv:2004.12765* (2020).