International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 09 | Sept - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Consumer Complaint Classification using Neural Networks

Praveen Kumar[1], Vidya Lalitha Rishitha [2], Naga Bindhu[3]

[1,3]Department of Information Technology,
Vignan's Foundation for Science, Technology & Research
(Deemed to be University), Guntur, Andhra Pradesh, India
[1]praveenkumarkazipeta@gmail.com
[2]vidyavadrevu28@gmail.com
[3]nagabindhukanadam@gmail.com

Abstract. Businesses are increasingly facing complaints from customers online, leading to the need for proper classification of these messages to improve transaction processing efficiency. In spite of the importance that has to be given to the issue only a little research was conducted in the categorization of complaint texts, particularly in regard to the utilization of negative emotions and character details. In addition, clients' erratic and aggressive behaviours present further grammatical and semantic difficulties. To tackle these issues, a complaint categorization model has been developed based on Word Embedding and GRU & LSTM, which creates word-level text vectors and extracts features from complaint messages. The relative importance of each word feature is then determined using an LSTM and GRU-based approach, allowing the model to prioritize the most important features for categorization. Experimental results have shown that this model outperforms several text classifications baselines.

Keywords: Recurrent Neural Network ,LSTM and GRU,Skip-Gram,Continous Bag of Words

1 Introduction

For machine learning algorithms to work effectively, it's necessary to identify the appropriate features in the data. This is a complex task that requires a deep understanding of the domain-specific data. The process of turning unprocessed data into features that may be used in machine learning is known as feature engineering. However, this can last for a lot of time. Deep learning exploits neural networks are capable of automatically extracting features from raw data without the need for feature engineering. For our study, we utilized Recurrent Neural Networks (RNN), which can store information over time. Backpropagation algorithm was used to train the network by computing full or partial derivatives of a function. RNNs can also be used to compute future values for time periods in addition to recurrent values. However, the presence of numerous hidden layers in deep neural networks can be a weakness, as it requires a significant The revised weights were computed using a certain amount of multiplications.

2. Related Work

Oguzhan Gencoglu proposed a system for automating the categorization of posts to minimise the manual burden, the largest virtual medical forum in Finland involved in handling forum messages. Naïve Bayes classifier was used by him to group the text messages into a group of 16 categories. Prasenjit Mitra, Cliff Brunk, and Sumit Bhatia introduced a classification method for web search queries based on analysis of the queries, achieving excellent classification results. Lorenxo Rossi and Omprakash Gnawali examined the user interactions in discussion foroums on coursera discussions, categorizing them based on a number of linguistically independent criteria, including the threads' structure, popularity, and temporal dynamics. MR. Bernard J. Jansen and the other researcher Danielle Booth have produced a method for by themselves categorizing web search enquiry on subject and user intent in real-time. D. Iraz Hernandez, Jansen Parth, Paolo Rosso, and Martha Rochagy developed a method for automatically extracting features from corpora, using Naive Bayes and SVM to categorize them. Kristof Coussement and Dirk A mechanism for automatically categorising emails was suggested by Van den Poel. to improve complaint-handling tactics, identifying grammatical disparities between complaint emails and others using Naive Bayes and SVM categorization.

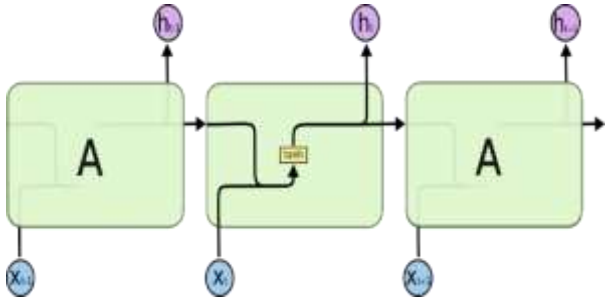
3. Framework for LSTM Model

The details of the many models and assumptions utilized in the construction and experimentation of the protocol are presented here. This section, will introduce our novel LSTM and GRU models. Firstly, we give a quick rundown of the character and sentiment embedding techniques. Next, we describe the mechanism of our LSTM and GRU models.

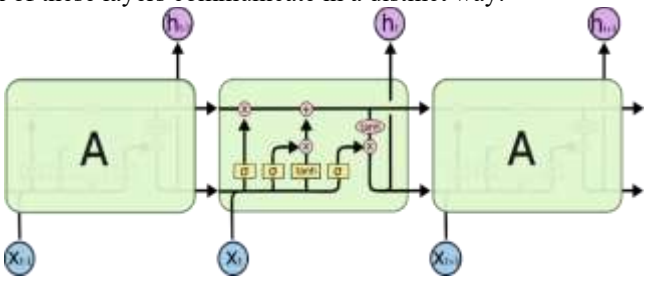
3.1 LSTM Networks

Hochreiter & Schmidhuber introduced LSTM networks. in 1997 as an exclusive class of RNN capable of managing dependency chains over time. Later, many other researchers have built upon and utilized these networks in their works, proving their effectiveness in solving various problems.

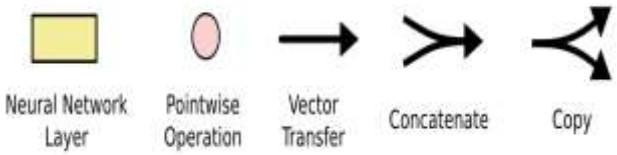
Continuous reliance is a problem that LSTMs are intended to alleviate., and they can retain information for a prolonged period without significant effort. Each RNN consists of a set of recurrent neural network modules that are arranged in a repeating pattern. Unlike the typical RNN, the recurrent module in LSTMs includes multiple layers, such as a tanh layer.



In a typical RNN, the repeating module just has one layer.
LSTMs have a topology similar to a chain, but their repeating module is constructed differently. The neural network has four levels rather than simply a single, and each of these layers communicate in a distinct way.



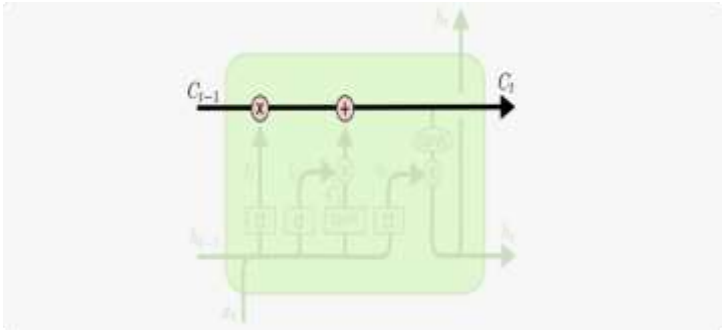
Don't be concerned with the information about whatever is occurring. Later on, we'll take you sequentially via the LSTM algorithm chart. Let's focus on getting familiar with the syntax that we're utilising for the time being.



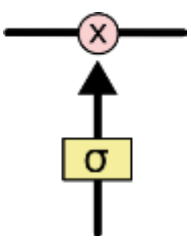
The given chart depicts the transmission of vectors between nodes in a neural network. The yellow boxes denote layers of learning neural networks, while The pink coloured dots represent operations that use points, such as adding vectors.. The merging lines indicate concatenation, whereas the forked lines demonstrate the copying and distribution of data to different locations.

3.1.1 Model Framework of LSTMs

The cell state, which is essential to the operation of LSTMs, is represented by the input line at the top of the picture. The cell stem can be thought of as a conveyor belt that passes information through the entire chain with minimal modifications, allowing information to flow unaltered.



With the use of gates, LSTMs may regulate the information flow inside the cell state. A layer of sigmoid neural networks plus an order of points multiplication mechanism make up these gates. The gates are optional and allow for the insertion or deletion of data from the cell's initial state.

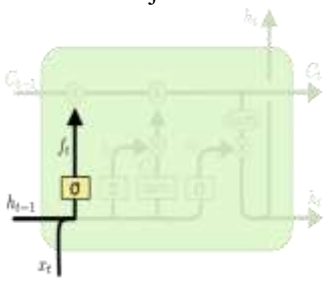


The sigmoid layer in an LSTM produces values between zero and one, which determine the amount of information to be let through. If the value is zero, no information passes through, while a value of one means that all information is allowed to pass through. The three gates are input, output and forget gates, which are used to control and manage the cell stem in an LSTM.

3.1.2Step-by-Step LSTM Walk Through

The next stage involves selecting the recent data that will be stored in the cell state. This is divided into two parts. The so-called sigmoid layer, or "input gate layer", decides which values will be updated first. A vector, Ct, produced by a tanh layer and containing potential new values is then used to update the state. To update the state, these two will be joined in the next phase.

We would want to add the male or female identification of the new subject to the cell state in order to replace the previous topic,

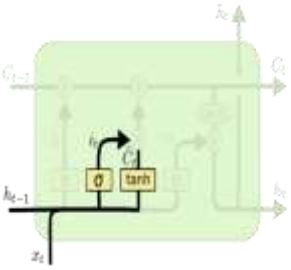


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

whose gender we have forgotten, in our language model.

The following LSTM stage selects the new data to be stored in the cell state. This process involves two steps. First, the "input gate layer," which determines which values should be changed. This is a sigmoid layer as well. A tanh layer is then used to update the state, producing a Ct vector of potential new values.

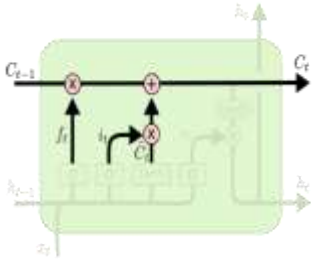
These two are joined in the last stage to provide the state an update. When it comes to our linguistic model, By adding the gender of the new subject to the cell state, we would replace the previous subject whose gender we have forgotten.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

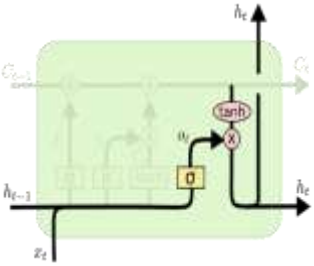
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Time has come to transition from cell state Ct1, the previous cell state, to cell state Ct. by implementing the decisions made in the earlier steps. The previous data that needs to be forgotten is retained, and the earlier state is updated by a factor of two. Then, the new data is added by multiplying it with Ct. The resulting new candidate values are sorted based on the degree of change for every state's value. When it comes to a linguistic model, the previous topic's gender information will be removed, and the new topic's gender information will be added, as per the earlier decisions.



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

The final step involves determining the output determined by the condition of the cell, regardless of what was filtered out. To identify which cell states should be output, a sigmoid layer is used first. After passing Sigmoid and tanh waves reveal the status of the cell, we multiply it is the result to output only the parts we chose (and to push the values between -1 and 1). For example, in a language model, if a subject is encountered, it may be useful to include information related to the corresponding verb. This could include whether the subject is singular or plural, which would help determine the appropriate form of the verb to use.



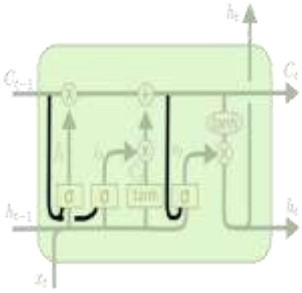
$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

3.1.3. Variants on Long Short Term Memory:

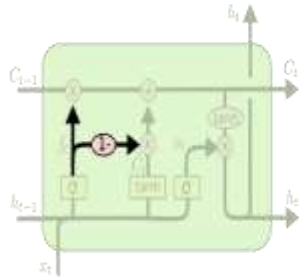
Long short term memory, often known as LSTM, is a kind of artificial neural networking created to address the vanishing gradient issue in recurrent neural networks (RNNs). LSTMs have a horizontal line at the top, representing the cell state, which can be modified by gated structures that add or remove information from the state. These gates are optional and consist of sigmoid layers and pointwise multiplication mechanisms. The sigmoid level outputs values in the range of 0 and 1, which concludes the passage of each component through the gate.

The forget gate layer, the initial stage of an LSTM, is in charge of selecting the data from the cell stem to be omitted. The input gate level follows, where a tanh level generates a vector of calculated new values and the sigmoid layer chooses which values will be changed. The decision on the output, which is dependent on the status of the cell after filtering, is made as the last step.
While there are variations in the way LSTMs are implemented, some of them include peephole connections, which allow the gate layers to monitor the cell state. Gers and Schmid Huber introduced this version in 2000. Other variations include coupled forget and input gates, which enable the network to learn when to forget and when to remember, and the use of multiple memory cells.



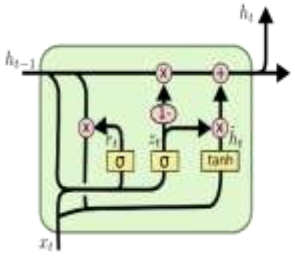
$$f_t = \sigma\left(W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [C_t, h_{t-1}, x_t] + b_o\right)$$

The design above includes peepholes for all the gates, although many documents only have certain peepholes.
Utilizing paired forget and input gates is another alternative. We make such selections together rather than independently determining what to leave out and what we should update. Only when we're about to insert anything in its place do we forget. Only when we forget a previous value do we add new values to the state.



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

Compared to the LSTM, Cho et alGated .'s Recurrent Unit, or GRU, is a little more striking (2014). A single "update gate" is created by combining the memory and input gates." It also adjusts different things and combines the cell state and concealed state. The resultant model, which is more straightforward than traditional LSTM models, has been gaining popularity.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Only a handful of LSTM variations have been discussed here, but there are many others, includes RNNs that handle long-term dependencies in various ways. It is unclear which variant is superior, and whether the differences between them are significant.
In a study by Greff et al. (2015), many popular LSTM variations were compared and found to be highly similar. Jozefowicz et al. (2015) tested over 10,000 RNN designs and found several that performed certain tasks better than LSTMs.

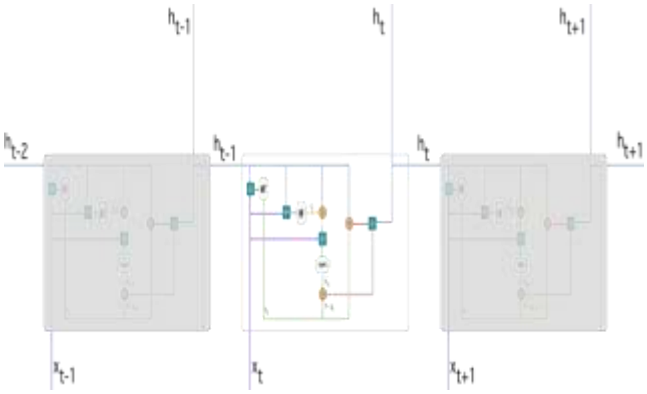
3.2. GRU Networks

Recurrent neural networks have a variation called gated recurrent units (GRUs), which have fewer parameters than LSTMs since they lack output gates. Studies have shown that GRUs and LSTMs have similar performance on several tasks like Speech signal modelling, polyphonic music modelling, and NLP. However, GRUs did give better results and performed well on some tiny, uncommon datamodules. To represent text in a system-readable format, text classification is performed using Word2Vec for both LSTM and GRU models. In 2014, Cho et al. introduced the Gated Recurrent Unit (GRU) as a remedy for the gradient that is fading problem in recurrent neural networks. In this essay, we aim to explain GRU in a clear and straightforward way.

3.2.1. Working of GRU Network

As previously stated, GRUs are an improved type of RNN. What sets them apart and makes them powerful is their ability to address the vanishing gradient problem that plagues traditional RNNs by utilizing update and reset gates.
These two vectors are responsible for determining wherein information may be taught to retain prior knowledge and exclude unnecessary information and should be forwarded to the output.

To further understand the mathematics driving that process, let's take a closer look at one recurrent neural network component below:



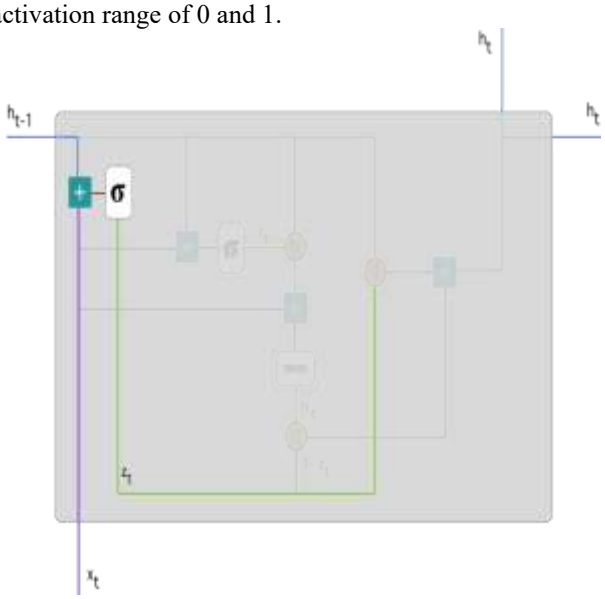
.

3.2.1.1. Update gate:

To start, we use the method below to obtain the modification gate for the following time step(t):

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1})$$

Weight W of X_t doubles when it is linked to a networking device (z). The identical thing is true for the value of h_(t-1),increased by the weight of itself U(z) and contains information for the t-1 units before it. The both values are added together, outcome will be compressed with a function called the sigmoid activation range of 0 and 1.



According to given architecture:

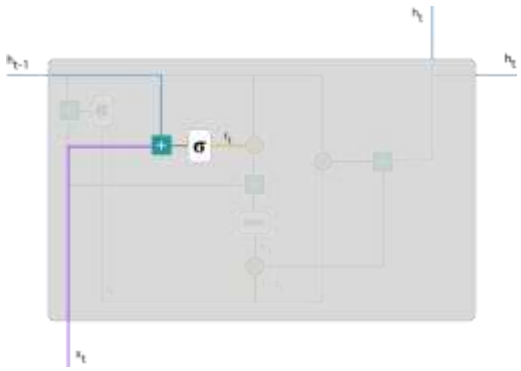
An update gate aids the model decide what amount of historical data of already executed conditions should be transferred to the future. technique is quite effective since the algorithm may decide to replicate all of the historical data, entirely removing the possibility of the fading gradients problem. We'll examine the update gate's functionality later. For now, keep in mind the z_t formula.

3.2.1.2. RESET GATE

This algorithm basically utilises this gate to decide which portions in the previous info must someway be lost. It will be decided by the use of:

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$$

The formula for the update gate is the same as this one. Soon, we'll observe how the gate and weights are used to correct for the disparity. The reset gate is indicated by the following schematic:



The same steps are taken: we add the h (t-1)—blue line and the X_t—purple line, increase it with some appropriate loads, The sigmoid function is then used to smooth the curves after adding the findings.

3.2.2. Information in the memory now:

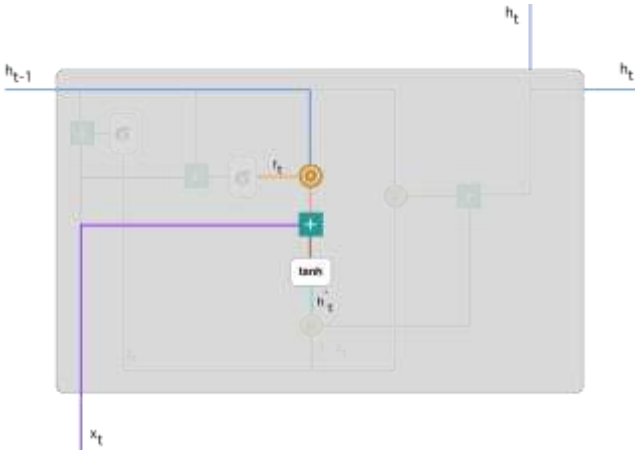
Let's look at how the gates specifically affect the output.We start by activating the reset gate. We offer new memory content that stores the required historical data in the reset gate. The equation reads as follows:

$$h_t' = \tanh(Wx_t + r_t \odot Uh_{t-1})$$

1. Activate the reset gate by providing new memory content that stores the required historical data in the reset gate.
2. Add weights W and U to the input x_t and h_t-1 respectively.
3. 3. By using Handamard formula of product to determine which data from past durations periods should be kept..
4. Gather the results of steps 2 and 3.

5. Use the tanh non-linear activation method.

Here are the steps in clear view:



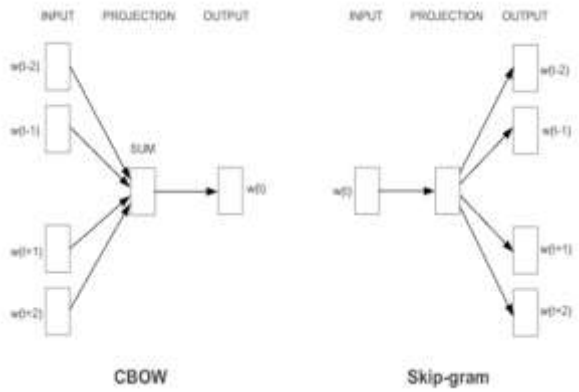
The lines h_(t-1) and r t are multiplied element-wise, and the output, the pink line, and the input, the purple line, x_t are then added. Finally, h'_t , or a bright green line is created using tanh.

The network computes h t, a vector that holds data for the current unit and sends it throughout the network, as the final step. The update gate must be used to do it. It chooses what to take from the information in the present memory, h' t, and from the previous phases, h_ (t-1).

4. Proposed Model

4.1. WORD2VEC:

Word2vec is an example mashup used to distributely describe keywords in a corpus C. A text corpus is used as the input for the Word2Vec (W2V) algorithm, which creates a vector representation for each word.This algorithm comes in two different forms, CBOW and Skip-Gram.



The word2vec technique is a well-liked method for creating dense vectors called word embeddings, which represent words in a high-dimensional environment. An extensive corpus of text, such as a group of documents or web pages, is used to train a neural network in the technique. Given a collection of paragraphs (also known as a corpus), the model turns on each sentence's words and tries to use the current word (w) to predict its neighbors (i.e., its context), in which scenario the approach is referred to as "Skip-Gram," or it applies each of these scenarios in order to foresee the up-to-date word (w), in which case it is referred to as "Continuous Bag Of Words" (CBOW). The term "window size" is a setting that is utilized to limit the number phrases used in each context. This window size determines the number of words that are used as context for each central word. The word matrices that were identified to be connected by the context in which they appear are pushed farther apart by altering the vector's numbers. In this session, we'll primarily focus on the Skip-Gram approach, that in comparison with CBOW, forecasts context words using the primary word as an input.
The Skip-gram model extracts a hot-vector for each word from a corpus of text. A hot vector is a word represented as a vector in which the vector's length corresponds to the vocabulary's size (total unique words). Except for the dimension corresponding to the word that is now being used as an input, all dimensions are set to 0. The length of this vector is usually smaller than the vocabulary size, typically between 50 and 500 dimensions, based on the quantity of data set and the problem's complexity.

In CBOW context is used to predict word and in Skip-gram per word is predicted. In our model, we considered 5 words, and the weights are calculated for those words, and those weights are called as word embeddings. Using tokenizer we convert text to sequences, which are lists of integers representing the words in the text. We then make all the sequences to equal size by padding zeros or truncating extra words. The padded data is considered as training data which will be given as input gate to both the LSTM and GRU models. These models can then learn to extract the most important features from the text and use them to make predictions on new data.

5. Experimental Analysis

On complaint datasets, we evaluate how well each categorization model performs.The total classification accuracy of several models on each dataset is displayed in the table below.

Table 1: Accuracy given by Machine Learning Models

Model	Methodology	Accuracy
Linear SVM	Tf-idf	85
Random Forest Classifier	Tf-idf	71
Logistic Regression	Tf-idf	54
Multinomial Naïve Bayes	Tf-idf	80

The above table describes the accuracy of the model performed using machine learning algorithms. The machine learning algorithms have not been much accurate so we have considered an other model based on deep learning neural networks which have given accuracy more than machine learning models. The below table given the accuracy given by deep learning models.

Table 2: Accuracy given by Deep Learning Models

Model	Methodology	Accuracy
LSTM(Long-Short Term Memory)	SG=0	90
	SG=1	85
GRU(Gated-Recurrent Unit)	SG=0	92
	SG=1	86

6. CONCLUSIONS

The Deep Learning models have performed more accurate than Machine Learning algorithms as we are using Neural Networks. The impact of modification on one weight is the primary distinction between a neural network and regression. Regression allows you to modify a weight without affecting the function's other inputs. With neural networks, this isn't the case. A single modification can have a cascading effect on the other neurons in the network since the output of one layer is transferred onto the next layer of the network. For Further Extensions we can performed using BERT based model which have a chance of more accuracy than the above performed methods.

References

[1] Oguzhan Gencoglu, "Automatic Classification of Forum Posts: A Finnish Online Health Discussion Forum Case", EMBEC 2017, NBC 2017: EMBEC & NBC 2017pp 169-172J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.
[2] Sumit bhatia, Cliff Brunk and Prasenjit Mitra, "Analysis and automatic classification of web search queries for diversification requirements", Proceedings of the American Society for Information Science and Technology Volume 49, Issue 1, Version of Record online: 24 JAN 2013.
[3] Lorenzo A. Rossi, and Omprakash Gnawali , "Language Independent Analysis and Classification of Discussion Threads in Coursera MOOC Forums", 15th IEEE International Conference on Information Reuse and Integration (IRI 2014), At San Francisco, CA.
[4] Bernard J. Jansen, and Danielle Booth, "Classifying Web Queries by Topic and User Intent", April 14–15, 2010, Atlanta, GA, USA
[5] D. Irazú Hernández, Parth Gupta, Paolo Rosso, and Martha Rocha "A Simple Model for Classifying Web Queries by User Intent", January 2012.
[6] Kristof Coussement, and Dirk Van den Poel "Improving customer complaint management by automatic email classification using linguistic style features as predictors", Decision Support Systems 44 (2008) 870– 882.