International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 11 | Nov 2022 www.irjet.net p-ISSN: 2395-0072

Sentiment Analysis: A comparative study of Deep Learning and Machine Learning

Rachit Manshani[1], Toshi Thatte[2], Kanhaiya Suryawanshi[3]

Abstract - *As the population of customers grows, it becomes increasingly tedious to analyze customer feedback. By 2025 it is estimated that more than 75% of the data relating to customer reviews will be unstructured, meaning that the challenges to analyze and gain insights from these reviews will increase.*

With help of sentiment analysis by machine learning or deep learning, data gathered via reviews can be categorized under the labels of positive, negative and neutral. Sentiment analysis makes it convenient to understand the customer's mindset and preferences, assisting companies in changing their strategies and products to better cater the needs of their customer base.

Key Words: Sentiment Analysis, Machine Learning, Deep Learning, LSTM, CNN, RNN, Logistic Regression algorithm, TF-IDF.

1. INTRODUCTION

Via Sentiment analysis, we classify a block of text under an umbrella of labels like positive, negative and neutral. Using Sentiment analysis, one can determine the general mindset of the customer pool which in turn helps the companies to introspect on the quality of services or products that they are providing, thus assisting them a great deal in upgrading themselves.

Various machine learning algorithms and deep learning techniques can be used for sentiment analysis, though the objective of this paper is to determine which work approach will yield the best results [1]. At the end of this research, we'll have a generalized idea on how sentiment analysis can be performed via machine learning and deep learning and which practices to employ for the optimum pre-processing and modelling of the dataset.

2. WORK DONE

E. Lakshmi and Edward deduced pre-processing of the data to improve a raw phrase's consistency structure. They utilized the LSA technique and cosine similarity.

Basant Agarwal et al. used the term pattern system for sentiment analysis. It works on extracting certain contextual and syntactic information from the dataset. The author wanted to put forward aspect-focused opinion polling based on unlabeled free form customer reviews.

M. Karamibekr and A.A. Ghorbani on the other hand suggested an alternative approach for sentiment classification of a text weighing on verbs as a vital opinion expression and hence can be used as a tool for sentiment analysis.

SentiFul is a sentiment lexicon/algorithm which utilizes and expands it by synonyms, antonyms, hyponyms etc. They identified four types of affixes based on their position in sentiment features namely propagation, weakening, reversing, and intensifying. For sentiment analysis, a lot of researchers have explored and used soft-computing methods, mostly fuzzy logic and neural works.

Significant contribution has been made by using a fuzzy domain sentiment ontology tree extraction algorithm. This algorithm makes a fuzzy domain sentiment ontology tree based on feedback, which involves extracting sentiment terms, product attributes, and feature relationships, and accurately predicting the polarity of the reviews [2]. They deduced and standardized the method of increasing the strength of reviewer's opinions in the presence of adverbial modifiers by designing membership functions. They used the technique to analyze adverbial modifier trigram patterns [10].

3. APPROACH

3.1 Determining the Dataset

There are several datasets available that can be used to efficiently for Sentiment Analysis and we'll determine which dataset will be the appropriate for the research we have in mind. Some prominent datasets are:

1) IMDB Movies Review Dataset:

This dataset contains about 50,000 movie reviews from IMDB. However, in this set only highly polarized reviews are being considered. From a total of 10, the negative has a score of less than 4 and the positive has a score of greater than 7.

2) Cornell Film Review Data:

This dataset contains around 220,000 conversations between pairs of different movie characters.

3) Movie Lens 25M dataset:

This dataset is collated from the Movie Lens website, this dataset comprises around 25 million ratings with 1 million tag applications given to about 62,000 movies in total.

4) Rotten Tomatoes Dataset:

In this dataset, each entry represents a movie available on Rotten Tomatoes website with the URL that is used for analyzing with data values like movie title, description of the movie, genre to which the belongs, duration, directors, actors, ratings given by the users and ratings given by the critics.

After analyzing several datasets, we decided to proceed with the Rotten Tomatoes Movie Review Dataset for the sake of this research as we felt that the above-mentioned dataset catered most to our needs.

Here, the submissions are evaluated using classification accuracy which is the percentage of labels that are predicted correctly for every parsed phrase. The sentiment labels are as follows:

0 - negative

1 - somewhat negative

2 - neutral

3 - somewhat positive

4 – positive

3.2 Initial Inspection

First most, we can see from the given dataset that each sentence is broken into multiple phrases, each having their own sentiment classification. Each sentence is denoted by 'SentenceId' column. Hence, we start with checking out the total number of unique sentences and understanding more about the sentence structure within both the training and test datasets. We then find the average count of phrases per sentence, per dataset.

There are much larger proportion of phrases compared to sentences in our dataset, and the average word length of phrases at 7 is quite low. This will need to be considered when cleaning the text in order to strike the right balance between making the data neater and losing too much data that renders Machine Learning more difficult.

3.3 Text Preprocessing

In this step, we combine both the datasets (training and testing) and clean. Noise removal is one of the most essential text preprocessing steps as it helps us to emit out certain characters, digits etc. which might hinder our text analysis

For example, language stop words (for instance - the, of, is, am, in etc.) URLs or punctuations and links - all of these will be considered as noise. Noise removal step deals with elimination of all types of noisy entities present in the text of our dataset.

Typically, we would define a function to clear all this noise, but this dataset contains a huge number of single word entries. Removing numbers, punctuation marks, special characters and whole words would wipe out a large chunk of the available observations, potentially resulting in more trouble during our modeling. Therefore, we are going to keep them in and proceed with a more 'light touch' noise removal function.

Our first step would be to introduce the basic text cleaning function. We make all the text characters lowercase and remove whitespaces. Then we perform conversion to string. We apply this function and create a new column for the cleaned text. With the text cleaned now, we can split the data back into training and test sets.

We now mark the sentiment column as -999 within the test set which indicates that there is zero chance of cross-contamination.

3.4 Preparation of Text Matrix

After the data pre-processing a text matrix is created for analyzing our dataset efficiently and making it algorithm compatible. For doing the same we'll be using the approach of "TF-IDF" (Term Frequency-Inverse Document Frequency) model.

In this model 'TF' can be defined as the number of times a term, say 'dog', occurs in a document and IDF can be stated as the logarithm of the ratio of total documents available to the number of documents containing the term 'dog'. Combined, the TF.IDF formula states the relative relevance of the term 'dog' in a list of considered documents [6].

$$TF(t,d) = \frac{number\ of\ times\ t\ appears\ in\ d}{total\ number\ of\ terms\ in\ d}$$

$$IDF(t) = log\frac{N}{1+df}$$

$$TF-IDF(t,d) = TF(t,d) * IDF(t)$$

We'll first get the total n-gram count or all the single- or two-word combinations from our dataset. Once extracted, we'll proceed to check the sparsity or the number of non-zero values present in our dataset. Our sparsity comes out to be 0.01% which works in our favor as the sparser our data is, the more challenging it'll be for us to model it. Sparsity and missing data can be distinguished as when there is missing data, it means that many data points are unknown, on the other hand, if the data is sparse, all the data points are known, but most of them have zero value.

Since we have calculated the term counts for both the training and testing datasets, we can proceed with applying the TF.IDF transformer for calculating the weights for each term in both the sets.

3.5 Model Selection

Machine Learning can be defined as the study of computer programs that utilize algorithms and statistical models to learn through deductions and patterns without being pre-programmed to do so.

There is a humongous number of ML algorithms which can be categorized into three distinct labels:

1. Supervised Learning: In this type, the algorithm is initially trained on labeled data. The ML algorithm is first given a small training dataset to learn from. This training dataset is a part of the main dataset and has the role to teach the algorithm the basic nature of our problem, its probable solution, and data points to work with. The algorithm then looks for connections between the parameters given, establishing a cause-and-effect relationship between the variables in the dataset. At the end of this process, the algorithm has an overview of how the data fits in and the connection between the input and the output. This solution is then put in place to employ the final dataset.

Depending on the nature of the target variable, it can be a discrete target variable task (Classification) model or a continuous target variable task (Regression) model.

- Classification Supervised Learning: It is a Supervised Learning task where output contains defined labels or discrete value. The main goal in this type of supervised learning is to predict discrete values belonging to a particular class and then assess them on accuracy.

- Regression Supervised Learning: It is a Supervised Learning task where output has a continuous value rather than the discrete values present in Classification learning. The main idea here is to predict a value as closer to the actual output value and then final evaluation is done by determining the error value. The insignificant the error, the better the accuracy of our model.

Examples of Supervised learning algorithms:

- Logistic Regression
- Support Vector Machine
- Decision Forest

2. Unsupervised Learning: Unsupervised learning can be described as the training of an algorithm using data that is neither labeled nor classified and letting the algorithm deduce on that data without any guidance. The major role of the machine is to gather unsorted information based on various factors like patterns, similarities and differences, all without any prior training.

Unsupervised Learning can be classified into the following two categories:

Clustering: A clustering problem is where we aim to classify the inherent groupings in the dataset.

Association: An association rule learning problem is where we aim to find orders that describe large portions of our dataset.

Examples of Unsupervised machine learning are:

- K-means clustering
- Fuzzy means

3. Reinforcement Learning: In this case, the algorithm discovers data via trial and error and then concludes what action will result in higher efficiency. Three major components that comprises reinforcement learning: the agent, the environment, and the actions. The agent can be considered as the learner and decision-maker, the environment comprises everything that the agent interacts with, and the actions are what the agent performs. One such example is the Markov Decision process.

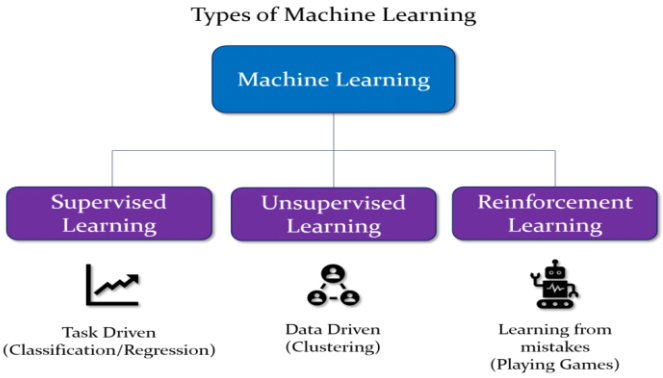

Fig -1: Classification of Machine Learning Algorithms

Now for our research's sake, we'll be considering Logistic Regression as our chief algorithm as it is convenient in handling multi class classification in a relatively shorter duration.

Logistic Regression: It is a supervised learning classification algorithm that can be used to predict the probability of a YES/NO event (binary event) happening. Therefore, the outcome must be a discrete value. It can be either No or Yes, 1 or 0, etc. but instead of giving the precise value, it gives the probabilistic values which lie between the two extremes [9].

The equation used by the basic logistic model is Ln (p/1-p) =a0+a1*x+a2*x This is called the logistic function.

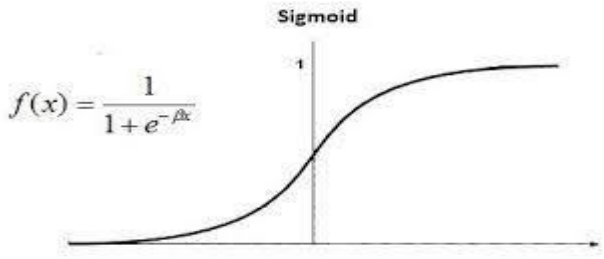

Fig - 2: Principal Graph for Logistic Regression

The basic idea of this is to test, train, find accuracy and then create a confusion matrix. The classification report with metrics such as precision, Recall, F1 score etc. will give us insights about how our model is performing.

	Precision	Recall	F1-score	Support
0	0.05	0.53	0.11	708
1	0.16	0.4	0.23	10648
2	0.92	0.61	0.73	123779
3	0.29	0.51	0.38	19221
4	0.11	0.57	0.16	1716
Avg/Total	0.78	0.57	0.63	156072

Fig - 3: Classification Report

The accuracy obtained from the above model is **56.2%** which is not suitable for our requirements and therefore we'll be turning to deep learning.

3.6 Deep Learning Approach

Deep learning mimics the basic structure and functionality of the human brain for recognition tasks and patterns. Deep learning can obtain certain useful features which in turn increases the efficiency of our predictive model making it possible to analyze our data more precisely [3].

Theoretically, the model used in deep learning is a mathematical function - f: X → Y. Deep learning is the evolution of a more in-depth Artificial Neural Network (ANN) using more than one hidden layer for modeling of the provided data [11].

It contains three prominent layers:

1. Input Layer: this layer receives data from variable X.

2. Hidden Layer: This layer receives data from the input layer. Each hidden layer trains a unique set of features. Increased numbers of hidden layers increase the complexity and abstraction of our model.

3. Output Layer: the layer's neurons receive data from the last hidden layer producing an output value and is the culmination of the calculation of the variable X to variable Y.

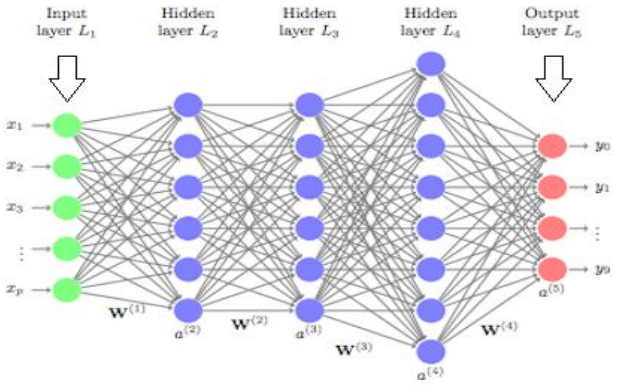

Fig - 4: Deep Learning Model

Deep learning can properly analyze data in its sequential form, something where our traditional machine learning models fail, as depicted from the inadequate accuracy obtained from the conventional Logistic Regression Model [4].

In this approach we'll be using the Long short-term memory (LSTM) algorithm. LSTM is a type of a Recurrent Neural Networks (RNN) which is distinguished by its uniqueness of processing of sequential information which is derived from the internal memory stored by directed cycles and hence RNN can not only remember but use previous computed information by applying it to the forthcoming element.

LSTM, moreover, is capable to use long memory as an input for activating the functions present in the hidden layer. LSTM was introduced to overcome the vanishing gradient problem usually associated with typical RNNs models. In short if a sequence is long, then it becomes difficult for RNN to carry a certain piece of information from one instance to a previous one, a foible removed by LSTM by using a memory cell [7].

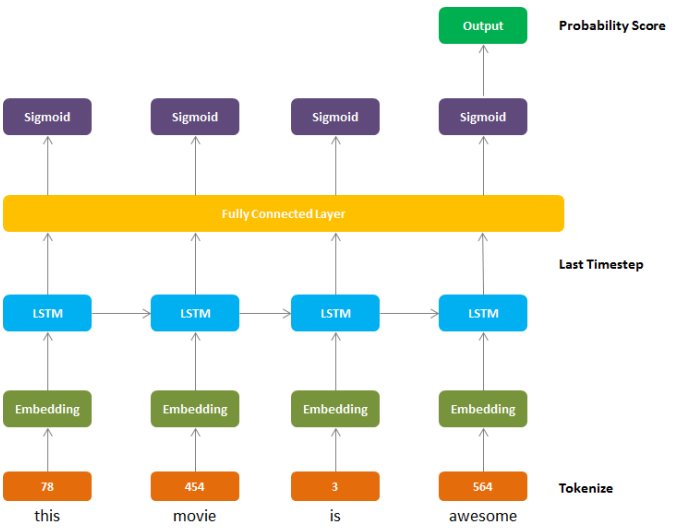

Fig - 5: LSTM Architecture

Furthermore, we'll be using a CNN model alongside our LSTM model to improve the overall efficiency.

CNN (Convolutional Neural Networks) is a type of neural networks which is distinguishable from its counterparts because of its ability to efficaciously process data having a grid like structure, for example an image [5].

A CNN consists of three layers:

1. Convolution layer: This layer is the foundational building block and manages the significant portion of the computational load. Convolution layer is tasked with performing a dot product between two matrices, the kernel (set of learnable parameters) and the restricted portion of the receptive field.

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

Formula for Convolution Layer

Fig - 6: Convolution Layer Formula

2. Pooling layer: This layer is tasked with replacing the output at certain positions with a derived summary statistics of nearby outputs which in turn helps in reducing the required amount of weight and computation.

$$W_{out} = \frac{W - F}{S} + 1$$

Formula for Padding Layer

Fig - 7: Padding Layer Formula

3. Fully Connected Layer: This layer helps in mapping the representation between the output and input. Neurons in this layer have complete connectivity with all the neurons in preceding and succeeding stages.

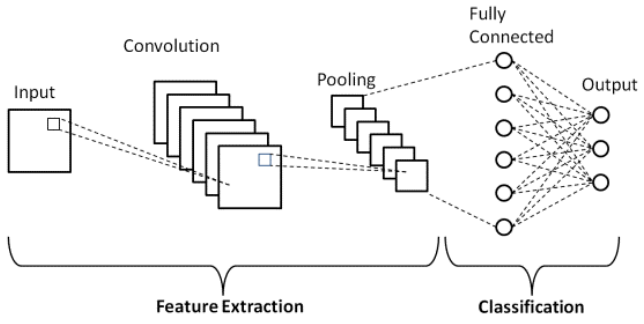

Fig - 8: CNN Architecture

After loading the necessary libraries from Keras package, we proceeded with certain processing steps to prepare our data further. These steps are:

1. Tokenization: Tokenization can be defined as breaking of raw text into smaller parts also known as tokens. This is done to ease the understanding of the context and progressing the model by analyzing the sequence of words as now the tokens can be considered as discrete elements.

2. Indexing: We'll perform indexing by giving our words an index and converting them into a dictionary-like structure. We'll perform this by using texts_to_sequences () which converts our text into integer sequences, thus creating a mapping of words and their specific integer indices.

3. Index Representation and padding length: We'll now transform our indexed list into a 2D array. This array will be of the shape of (num_samples, num_timesteps) where num_timesteps is either the longest sequence length or the 'maxlen' argument. In our case we'll set out 'maxlen' to 20. Values are padded to sequences which are shorter than 'maxlen' and similarly sequences which are longer are curtailed to obtain the desired length.

Finally proceeding to our hybrid model of LSTM and CNN combined. We'll proceed with making an embedding matrix using the word vectors trained on Common Crawl and one hot encoding our y variable or training set. Our model contains a commencing layer of LSTM that will collect all the word embeddings corresponding to each token as an input. Our main aim is that the final output tokens will contain the information of all the tokens preceding it; Or we can say, the LSTM layer is fabricating a new encoding for the original input. The output is then transferred into a convolution layer whose function will be to extract all the local features [8].

Moving over, the CNNs output will be amalgamated to a smaller dimension and ultimately derived as a positive/negative label. The final accuracy received using the hybrid model is **86.12%** which is far more satisfactory than the one obtained via machine learning.

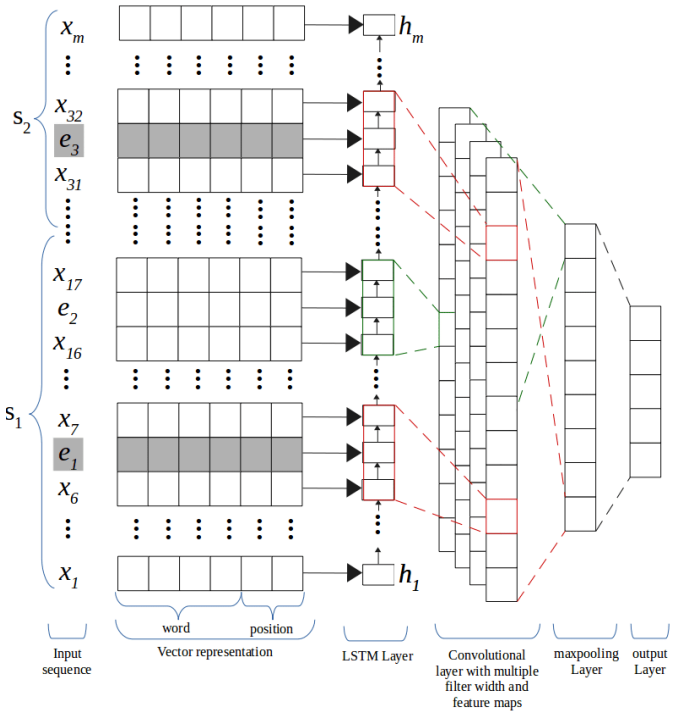

Fig - 9: Hybrid LSTM-CNN Model

4. RESULTS AND OBSERVATIONS

As we used a modified version of the infamous Rotten Tomato dataset which is further divided into training dataset and testing dataset. The corresponding sentiment labels that we have used are:

0 - negative

1 - somewhat negative

2 - neutral

3 - somewhat positive

4 – positive

First, we have utilized the machine learning algorithm - Logistic Regression. However, the accuracy of the model i.e., **56.2%** obtained from logistic regression was not satisfactory so we turned towards deep learning.

377	246	67	13	5
2275	4397	3029	813	131
4125	21461	72968	21836	3384
268	1137	3414	9683	4717
27	32	104	582	969

Fig - 10: Confusion Matrix for Logistic Regression Model

After using a hybrid model of LSTM-CNN, we did receive an optimum performance from our model. The corresponding accuracy obtained was **86.12%** which catered to our needs more.

Epoch 00009: val_loss improved from 0.31239 to 0.31127, saving model to best_model.hdf5

Epoch 10/15

39168/140454 [=======>.....] - ETA: 33s

- loss: 0.3120 - acc: 0.8612

Hence, we can clearly observe that our deep learning model outperforms our machine learning model which is primarily because deep learning possesses the ability to capture any kind of sequential dependency between different words in a sentence, this is made possible because of the Recurrent Neural Networks, which LSTM is a special type of.

5. CONCLUSION

To determine which approach (Machine Learning or Deep Learning) is most suitable for the purpose of this research i.e., for precise sentiment analysis of our dataset, we calculated the accuracies of our chosen architectures.

Our research indicates that our hybrid deep learning model of LSTM-CNN (Accuracy - 86.12%) comply with our needs better than the machine learning model, Logistic Regression (Accuracy - 56.9%) and hence, deep learning algorithms are more suitable for sentiment analysis on the rotten tomato dataset.

REFERENCES

[1] Liu B. Sentiment analysis: mining opinions, sentiments, and emotions. The Cambridge University Press, 2015.

[3] PangB and Lee L. Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval, 2008.2(1–2): pp. 1–135.

[4] Goodfellow I, Bengio Y, Courville A. Deep learning. The MIT Press. 2016.

[5] Glorot X, Bordes A, Bengio Y. Deep sparse rectifier neural networks. In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS 2011), 2011.

[6] Rumelhart D.E, Hinton G.E, Williams R.J. Learning representations by back-propagating errors. Cognitive modeling, 1988.

[7] Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, and Kuksa P. Natural language processing (almost) from scratch. Journal of Machine Learning Research,2011.

[8] Goldberg Y. A primer on neural network models for natural language processing. Journal of Artificial Intelligence Research,2016.

[9] L. Liu, X. Nie, and H. Wang," Toward a Fuzzy Domain Sentiment Ontology Tree for Sentiment Analysis," Processed of the 5th Image International Congress on Signal Processing (CISP), pp. 1620 – 1624,2012.

[10] R. Srivastava, M. P. S. Bhatia," Quantifying Modified Opinion Strength: A Fuzzy Inference System for Sentiment Analysis," International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1512-1519,2013.

[11] ApoorvAgarwal, BoyiXie, Ilia Vovsha, Owen Rambow, Rebecca Passonneau. Sentiment Analysis on twitter data. Department of Computer Science, Columbia University, New York, NY 10027 USA.