Comparison of Traditional Machine Learning and Deep Learning Approaches for Sentiment Analysis

Dhvani Kansara[1*] and Vinaya Sawant[2]

[1] Dwarkadas. J. Sanghvi College Of Engineering, Mumbai, India
[2] Assistant Professor, Dwarkadas. J. Sanghvi College Of Engineering, Mumbai, India
dhvani.djk@gmail.com

Abstract. With the advent of deep learning paradigms, many new algorithms have been flourishing which can be leveraged for the task of sentiment analysis. In this paper, we present an experimental comparison of traditional machine learning algorithms- Naive Bayes, Logistic Regression, Random forest using bag of words model along with modern deep learning algorithms like Recurrent Neural Networks with Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) using word vectors on reviews and tweets using three different datasets to classify a binary output, based on positive or negative polarity of the sentiment. We compare the results to find which model provides the best accuracy and analyze reason behind the differences in accuracies. The goal is to explore Natural Language Processing (NLP) models trained on a certain dataset and then benchmark their respective performances, thereby ultimately demonstrating that deep learning algorithms prove to be more effective. This is because Deep learning ensures that they understand the language that they are interpreting by taking *word embeddings* as input and pass them to the hidden layers to discern the context of the sentiments, thereby making more accurate predictions.

Keywords: Sentiment Analysis, Machine Learning, Classification Algorithms, Deep Learning, Bag of Words, Word Vectors, Embeddings, NLP.

1 Introduction

Natural language processing (NLP) is the field lying at the intersection of computer science, artificial intelligence and linguistics. Sentiment Analysis, which is a sub-domain of NLP is a process of determining the emotional tone of a comment based on words contained in it, in order to understand the attitude and opinion behind it. This process is also called opinion mining. It has multiple applications mainly in the form of social media monitoring. In recent years there has been great advancements in this field. Various machine learning algorithms have shown to prove effective in categorizing the sentiment from a text. Recent advances have been focusing on using deep learning algorithms for this purpose. This proliferation is due to the fact that opinions are central to almost all human activities and are key influencers of our behaviours. Our beliefs and perceptions of reality, and choices we make, are, to a considerable degree, conditioned upon how others see and evaluate the world[1].

In this paper, we discuss two paradigms: traditional approaches for classification which have been in use since the past few decades and the recent breakthroughs leveraging deep learning algorithms. The models work on raw data further cleaned and pre-processed, considering removal of stop words, punctuations and mark-ups with stemming (process of reducing inflected words to their word stem) and then checking the overall sentiment by assigning polarity based on resultant cleaned sentence. The models also take into account inversion terms or negations (such as - “not bad”) which reverse the polarity of the sentence as a whole. Later, this data is fed into a learning-based model that uses a supervised learning algorithm. In case of traditional models, this text is pre-processed using the bag of words methodology. It gives a combination of types of semantic features that attempt to model the syntactic structure of sentences, intensification, negation, subjectivity or irony[19]. In this paper, we use Naive Bayes, Logistic Regression and Random forest as traditional classification approaches. On the other hand, deep learning models learn from a multiple layers of representations or features of data and produces results accordingly. For this, we use vector representations of sentiments as inputs to the deep layers. We use modified Recurrent Neural Networks (RNN) or Long Short Term Memory (LSTM), Convolutional Neural Networks (CNN) and a combination of CNN and LSTM to obtain experimental results. We then analyse the accuracies obtained in all these algorithms.

We present each paradigm along with the description of pre-processing phases and an overview of the algorithms used. We also try to analyse reason behind difference between the accuracies. We use three different datasets for this purpose and classify movie reviews, hotel reviews and tweets, thereby concluding that deep learning models prove to have an upper edge in all the cases.

2 Methodology

2.1 The Traditional Approach

Extensive research has been carried out in the past few years to apply basic machine learning algorithms for sentiment analysis. We leverage this research to ensure that our approach will provide the best results. In order to obtain a good performance accuracy, the pre-processing phase is carried out prior to the classification process.

Pre-processing. We utilize packages in python to help assist the pre-processing phase. The reviews maybe taken directly from a website, so it may include html mark-up. Thus, we first use the BeautifulSoup package to strip all html mark-up. We then strip the stop words using the Porter Stemming package in Natural Language Toolkit (NLTK) to stem and remove stop words like “a”, “and”, “is”, ”the” because these are frequently occurring words which do not carry any significant meaning. Then, we convert this categorical data of cleaned reviews and tweets into numeric data so that we

can apply machine learning algorithms on them. We use an approach called Bag of Words [2] for this. This model creates a dictionary by taking all words in the dataset and then assigns integers according to the count of each word of the dictionary appearing in the given text. For example, consider the following two sentences:
Sentence 1: "The movie had amazing actors"
Sentence 2: "The view from the hotel room was amazing"
So, our vocabulary will be as follows:
{the, movie, had, amazing, actors, view, from, hotel, room, was}
To get our bags of words, we count the number of times the word from the vocabulary list occurs in each of our sentences.
In Sentence 1, "the", "movie", "had", "amazing", "actors" each appears once, so the feature vector for Sentence 1 is:
{ 1, 1, 1, 1, 1, 0, 0, 0, 0, 0 }
Similarly, the feature vector for Sentence 2 is: { 2, 0, 0, 1, 0, 1, 1, 1, 1} In this way we obtain a sequence of integers for each review and tweet.
In the datasets, we have a very large number of reviews, which will give us a large vocabulary. To limit the size of the feature vectors, we chose a maximum vocabulary size of 5000 words i.e. we consider the top 5000 most frequently occurring words in our vocabulary. (Considering we already removed stop words, this is enough). Thus, we obtain a list of size 5000 with integer values mapped according to the vocabulary for each row.

The Algorithms Part I. The dataset is divided into 80% tuples as training data and remaining 20% tuples as test data and the algorithms are applied on it. Accuracy is calculated based on how correctly the model predicts the test data. Three traditional classification algorithms were experimented,

1. Naive Bayes
2. Logistic Regression
3. Random Forests

Naïve Bayes Algorithm. It is based on the formula that, for a label y, we have the independent data feature x_i as in Eq. (1).

$$P(y|x_1,...x_n) \propto P(y)\ \Pi_{i=1}^{n} P(x_i|y) \quad (1)$$

Where $P(y|x_1,...x_n)$ is the posterior probability stating that x belong to class y. To create the classifier model, the probability of given set of inputs for all possible values of the class variable y is found, the output with maximum probability is the final classification result. This can be expressed mathematically as in Eq. (2).

$$y = \text{argmax}_y P(y)\ \Pi_{i=1}^{n} P(x_i|y) \quad (2)$$

This algorithm assumes the property of conditional independence among all attributes, which may not be true in all cases thus reducing the accuracy to a certain extent[9].

Logistic Regression. It is one of the most commonly used binary classification algorithm in machine learning which gives discrete binary output between 0 and 1. It uses a logistic function, also called as sigmoid function whose equation is as shown in Eq. (3)

$$f(x) = \frac{1}{1+e^{-x}} \tag{3}$$

Taking into consideration the weight of each input variable the output is predicted, this weighted input is passed to the sigmoid function to obtain a probability value between 0 and 1. Figure 1 [18] shows the graph of sigmoid function, indicating that, for any real number input, the sigmoid function maps it to a value between 0 and 1. These values will then be transformed into either 0 or 1 using a threshold classifier. For instance, if the threshold is 0.5, then, if the output of the sigmoid function is greater than 0.5, then that value is classified as 1 or positive and if it is less than 0.5 then output is classified as 0 or negative [10].

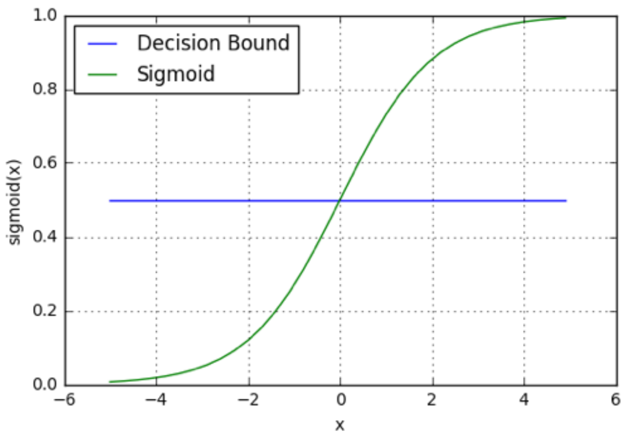


Figure 1: Sigmoid Function

Since it is a discriminative classifier which does not rely on the assumption of attribute independency as in case of Naive Bayes, thus this model may generally have a better accuracy comparatively.

Random Forest Classifier. It is formed by a collection of multiple decision tree classifiers collectively called as a “forest”. Decision tree classifier concept is based on the rule-based system. Individual decision trees are generated using selection of attribute at each node to determine the split. This node is selected in a strategic manner such that the most efficient tree is formed. The flow of the tree to classify a certain tuple will be in the form of an if-then rule. The same set rules can be used to perform the prediction on the test dataset. Multiple such trees are formed with different combinations of training and testing data. Then, it takes the test features and uses the rules of each randomly created decision tree to predict the outcome and stores the predicted outcome. Votes for each predicted result is calculated. Result which has received the maximum votes is considered as the final prediction. As shown in Figure 2 [16], multiple trees (different rules) are generated from given input instances or attributes and majority of the votes of the classifier output is considered as the final class [11]. Overfitting is avoided in this algorithm because every time a random set of inputs are used to build the decision tree. We use 100 estimators, i.e. 100 different decision trees to train the model and 'entropy' as splitting criteria for determining the best split point.

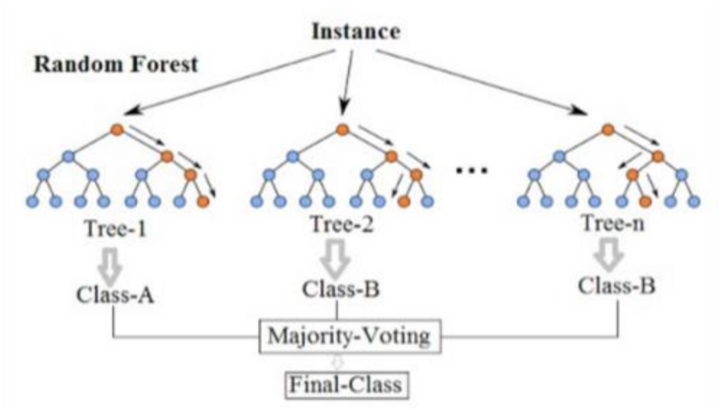
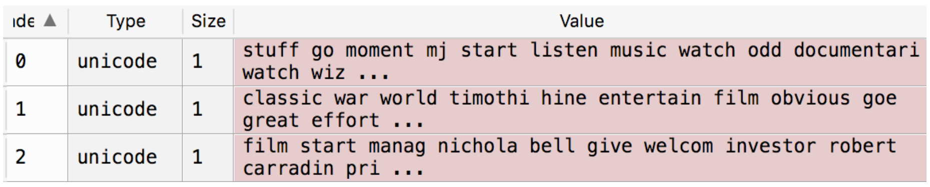


Figure 2: Working of Random Forest Classifier

2.2 Deep Learning Approach

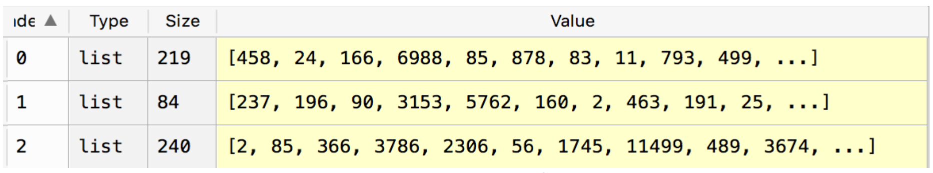
Deep Learning architecture consists of input layers, hidden layers and output layers. Each layer is made up of neurons, which are basically weight matrices. The higher the weight, the more influence layer has over the next. The input is passed through the hidden layers to give a certain output. Backpropagation is used to minimize the loss incurred between predicted and actual outcome. It does so by changing the weight matrices in the dense layers such that error is minimized. In this way the learning occurs.

Pre-processing. The cleaning of the datasets is done as in the previous case i.e. by removing HTML mark-ups, punctuations and stop words. For input to the dense layers we form a vector in latent space for each review, this approach is called Word-to-Vectors. We use tokenizer for dividing sentences into tokens (or words). Unique words are identified and unified in a list, from all inputs taken into consideration collectively. We then assign an integer value for each of the unique word in the list. So now we have a dictionary of words each assigned a specific integer number. Then, each review is represented by a list of these integers which form a part of that review. The transformation is as shown in Figures 3 and 4.



ide ▲	Type	Size	Value
0	unicode	1	stuff go moment mj start listen music watch odd documentari watch wiz ...
1	unicode	1	classic war world timothi hine entertain film obvious goe great effort ...
2	unicode	1	film start manag nichola bell give welcom investor robert carradin pri ...

Figure 3: Cleaned Reviews



ide ▲	Type	Size	Value
0	list	219	[458, 24, 166, 6988, 85, 878, 83, 11, 793, 499, ...]
1	list	84	[237, 196, 90, 3153, 5762, 160, 2, 463, 191, 25, ...]
2	list	240	[2, 85, 366, 3786, 2306, 56, 1745, 11499, 489, 3674, ...]

Figure 4: Token representation of reviews

Next, we need to truncate and pad the input sequences so that they are all the same length thus making them suitable for modeling. We pad the short reviews with 0s in the beginning and truncate the longer reviews. During training, the model will learn that the 0s carry no information thus preserving the content in each case and equalizing the length of each input, which is necessary for computation. We constrict a total size of 300 for each review.

The Algorithms Part II

The datasets are divided into 80% tuples as training data and remaining 20% tuples as test data and the algorithms are applied on it. Accuracy is calculated based on how correctly the model predicts the test data. The following algorithms were experimented:

1. Recurrent Neural Network (Long Short Term Memory)
2. Convolutional Neural Network
3. Convolutional Neural Network + Long Short Term Memory

Recurrent Neural Network (Long Short-Term Memory). Just as human’s thoughts are persistent that is we don’t start thinking from scratch every time we come across a problem, we use our previous knowledge from our memories to better understand the problem. Similarly, RNNs remember all the relations while training itself and are thus used particularly for sequential data [6]. The output of the previous state also goes into the input of the next state along with current input vector, this helps the model to remember context while training. That is why it is used for sentiment analysis and due to its nature it can be used to correctly predict sentiments of statements such as “Though initially I liked the movie, but towards the end it became dull and boring” this sentence has a negative sentiment but also positive words like “liked” which can confuse the system, but not in case of recurrent neural network, which can remember long term dependencies.

In traditional recurrent neural network, during the back-propagation phase, if the values of weight matrix of the recurrent neural layers are very small (less than 1.0) that ultimately they do not have any effect on the input signal and thereby preventing the neural network from training further. The problem is called vanishing gradient problem because the gradient signal vanishes slowly when passing through different layers [3]. Thus, RNNs have problems learning long range dependencies i.e. relations between words that are several steps apart. Conversely, we could also face exploding gradient problem if the weight matrix values become too huge, causing training to diverge.
For this reason, we use a modified version of recurrent neural network called Long Short Term Memory (LSTM) Neural Network which is capable of learning long term dependencies. We need to update information less chaotically and account for all the learning and the memory that it has learnt over a vast period of time, in order to make more accurate.

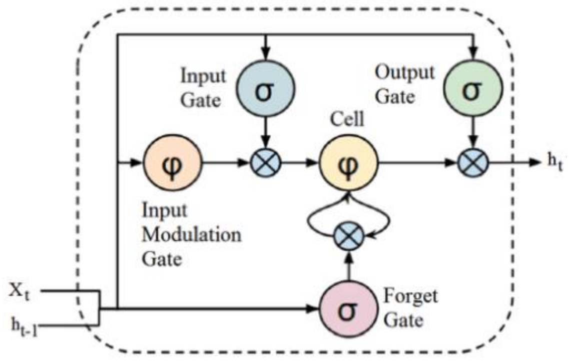


Figure 5: LSTM Functioning

Labels in the figure 5 and equations below represents:
X_t – current input vector |
h_{t-1} -Output of previous block |
h_t – output of current block
σ – sigmoid function |
φ – tanh function |
b – coefficients |
W – weights

Figure 5[17] describes the functioning of a LSTM model. The cell state is the most important component which "remembers" values over arbitrary time periods thus giving LSTM a long term memory, it represents all the learnings across time.
An LSTM cell consists of 3 gates as shown in Figure 5:

i. A forget gate which is used to formulate an attention mechanism which helps filter out relevant data by knowing what to forget and what to remember. It takes current input(X_t) and previous output(h_{t-1}) and passes it through a sigmoid function and judges if the data is worth remembering, which outputs 1 if data is to be remembered or 0 if it is to be forgotten. Thus its equation is given by eq. 4.

$$f_t = (W_f [h_{t-1}, X_t] + b_t) \tag{4}$$

ii. An input gate which decides what new information we are going to store in the cell state by involving a sigmoid function to X_t and h_{t-1}. The equation of input gate is given by eq. 5.

$$i_t = (W_i [h_{t-1}, X_t] + b_i) \tag{5}$$

The tanh layer (gives output between -1 and +1) is responsible for creation of vector of new candidate value, $\tilde{C}_t$ as given by eq. 6.

$\tilde{C}_t = \tanh(W_C [h_{t-1}, X_t] + b_C)$ (6)

This new candidate value is multiplied with the input gate and forget gate output along with previous cell state, C_{t-1} is used to formulate the new cell state C_t as given in eq. 7.

$C_t = (f_t * C_{t-1}) + (i_t * \tilde{C}_t)$ (7)

iii. An output gate which decides what information we are going to output as in eq. 8.

$o_t = (W_o [h_{t-1}, X_t] + b_o)$ (8)

Thus, output of current block is given by multiplying output gate value with new cell state passed into a tanh layer as shown in eq. 9 [3,4,17].

$h_t = o_t * \tanh (C_t)$ (9)

Thus, this model is able to remember context by storing relevant information in its long term memory.

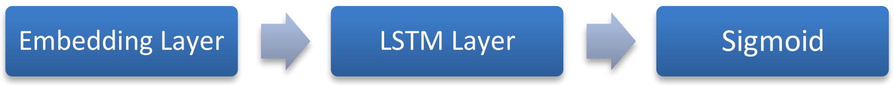


Figure 6: Layers in training LSTM Model

Embedding Layer. Above figure 6 shows layout of our model, before the LSTM layer, we add an embedding layer whose role is to map semantic meaning to geometric space. This is done by associating a numeric vector to every row in the dataset, such that the distance between any two vectors would capture part of the semantic relationship between the two associated rows. For example, consider two words "potato" and "panda" which are not semantically related so their vectors would be far apart but "green" and "grass" will have closer values. The output values will be in the form of vectors with related word vectors having close by distances, thus closer values. Also, vector arithmetic is applicable to these word embeddings i.e. for example,

$$\text{Vector}_{\text{man}} - \text{Vector}_{\text{woman}} \approx \text{Vector}_{\text{king}} - \text{Vector}_{\text{queen}}$$

Thus, vectors are formed based on the contexts of the words. So for example, if the machine assigns -1 for gender male and +1 for female, eventually learning these will enable queen getting a gender dimension of +0.97 and king of -0.95 and for say for mango and peach sort of genderless. Another dimension can be how royal they are so man and woman may have values closer to 0 whereas king and queen will have a high value for this dimension. And mango and peach might also have low values. Other dimensions may include whether or not is it a fruit, ages, cost, is it a noun or verb, size, etc. thus one can come up with multiple dimensions to represent an item. In our case, vectors will be formed based on how closely related two reviews (rows in the form of tokens) are. We use an embedding dimension of 100 which means that each review that is in our dataset is mapped into a 100 dimension dense vector of floating numbers. The vector will look like- (0.15,.. 0.23,.. -0.55).

This output is fed into the input of LSTM layer, in this layer, we apply a dropout of 20% to reduce overfitting, which drops 20% neurons in this layer. Another hidden layer is added following this which uses the sigmoid function to give a binary output.

Convolutional Neural Networks. Convolutional neural networks were designed to honor the spatial structure in image data whilst being robust to the position and orientation of learned objects in the scene [13]. They use pixel values and feature matrix for this. The same principle can be used to help learn structure of paragraphs of words, namely the techniques invariance to the specific position of features. This is applied over one dimensional sequence of reviews to learn its sequence in a similar fashion. So, instead of image pixels, the input to CNN model are sentences represented as matrix. Each row corresponds to one word, these are word embeddings, explained previously. The width of the feature filter is same as the width of the input matrix and height is varied. The figure 7 shows the layers of CNN.

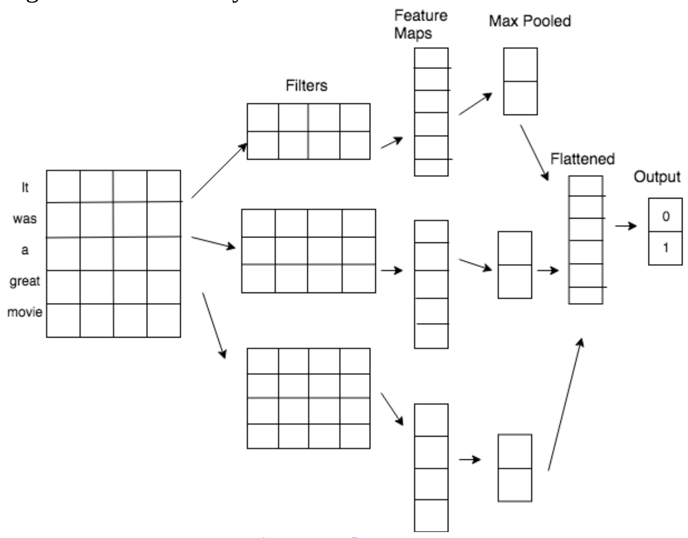


Figure 7: CNN Layers

The input sentence is passed into a feature detector or a filter, which are basically vectors used to learn the sequence of the statements. As the filter is sliding (or convolving), it is multiplying its weight values with the original word embedding value matrix of the review. The multiplications are summed up to a single number, which is a representative of the receptive field. After scanning is done, we get an array which is called as the feature map. In CNN, we use multiple filters to scan the input. In our model, we make use of 100 filters each of size 3, which forms the first convolutional layer. The convolutional layer is responsible for the main feature extraction. We also used rectifier activation function to remove the linearity (if any) present in the data. Following this layer is the max pooling layer which is used to reduce the spatial size of the representation, which in turn reduces the computation complexity of the network, this reduces the number of dimensions. Here, care is taken that the most important information is retained. The output from here is then flattened to form a 1D matrix which is fed into input of the next dense hidden layer, which uses rectifier activation function that transforms the incoming signals of the neurons and pass these signal to the input of the next hidden layer, which uses sigmoid as its activation function to classify a binary output. CNN has a special spatially local correlation by enforcing a local connectivity pattern between neurons of adjacent layers. Such a characteristic is useful for classification in sentiment analysis where we expect to find a certain pattern of sequences that maybe present in different parts of a sentence [14,7]. Figure. 8 shows different layers used in training the CNN model, the sub layers inside CNN responsible for its core functioning are shown in figure. 7.

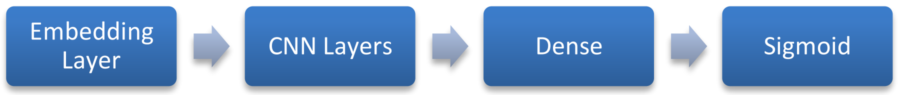


Figure 8: Layers in Training LSTM Model

Convolutional Neural Network + Long Short Term Memory. This model is a hybrid of the previous two models. The CNN will gather distinct features for positive and negative sentiment and these spatial features will then be learned as sequences by the LSTM layer[15].

Accordingly, we use a one-dimensional CNN and max pooling layer after the Embedding layer which is then fed as the consolidated features to the LSTM. We use a set of 100 filters with a filter length of 3. The pooling layer used is of the standard length 2 to halve the feature map size. The max-pooled output is then fed as the input to the LSTM layer, followed by a hidden layer, which uses sigmoid as activation function to classify a binary output. Figure 7 shows layers in the model. The intuition behind this model is that the convolution layer will extract local features and the LSTM layer will then be able to use the ordering of said features to learn about the input's text ordering.

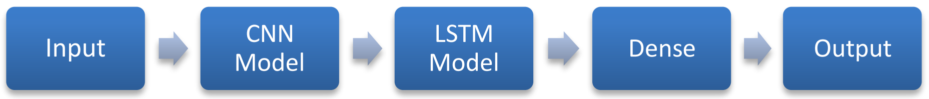


Figure 9: Layers in training CNN+LSTM Model

3 Experiments

The following three datasets were used:

1. IMDB Movie review dataset

This dataset was taken from the Kaggle website. It consists of 25000 rows each containing 3 attributes- A unique identifier, review for a movie and its corresponding binary sentiment- 1(positive) and 0(negative). We divide the dataset into 20000(80%) rows in training set which will learn from the sentiment labels and test it on remaining 5000(20%) rows as the test set to find out how the model performs.

2. Twitter Dataset

This dataset was also taken from the Kaggle website. It consists of three columns, tweet id, tweet text and its corresponding binary sentiment. Since we are considering only binary outcomes i.e. 0 or 1, the accuracies obtained in testing this dataset are low comparatively because we are not considering a neutral output, which may be frequent among tweets. Again, we divide the dataset into 80%-20% training and testing data respectively.

3. Datafiniti's Business Database

This dataset was taken from data.world and it contains a data about 1,000 hotels and 35,000 rows with attributes: hotel location, name, rating, review data, review username. From this we use hotel review data and corresponding rating. Since the rating is on a scale of 5, we consider rating > 2 as positive i.e. assign binary 1 sentiment and rating < 3 as negative i.e. assign binary 0 sentiment during the pre-processing phase. We then divide the dataset into 80%-20% for training and testing respectively.

In all three cases, the models learn from the training dataset about the words which help classify a review as positive or negative. Then, it predicts the outcome (sentiment) for reviews in the test dataset. Thus, the input is the given review/tweet and the output is predicted label- 0 or 1 of that input text. To find the accuracy, we compare the predicted labels with the actual labels of those corresponding reviews in the test dataset. Accuracy is therefore given by the formula in Eq. 10

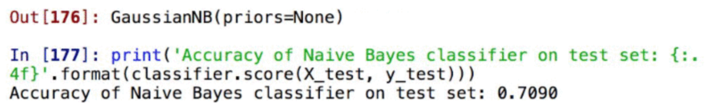
$$\text{Accuracy} = \frac{Correctly\ classified\ rows}{Total\ number\ of\ rows} \tag{10}$$

We use spyder as IDE to code in python in order to perform pre-processing and modelling. For traditional machine learning models, we use sklearn libraries to train and test data. For deep learning methods, we train models on keras build upon tensorflow as backend. In the deep learning approach, for training we use 1 epoch with a batch size of 32 in each case, meaning we update the weights once in each hidden layer, with 32 training tuples at a time, iteratively, till all training tuples exhaust. Before training, we configure the learning process by compiling with 'adam' as optimizer function for efficiently calculating the gradient descent during the backpropagation for minimizing the loss incurred during while updating weights. To calculate this loss, we use binary_crossentropy as the loss function for all our models.

We also ensured that we chose the best methods of pre-processing in each case such that accuracy is not compromised. Figure 9 shows the outputs of fitting train data (of hotel reviews) onto respective models and then testing them on the test data.

4 Results

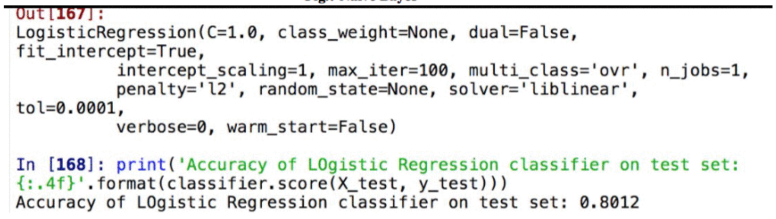
Figure 10 shows the outputs of applying all six algorithms on Hotel Review Dataset. It mentions the algorithm applied along with their respective accuracies. In the same manner, all other datasets were also experimented and their accuracies are as obtained in table 1. From this, we can infer that in all three cases, deep learning algorithms prove superior over traditional machine learning algorithms. The drastic characterization of change in accuracies can be seen in figure 11.



```
Out[176]: GaussianNB(priors=None)

In [177]: print('Accuracy of Naive Bayes classifier on test set: {:.
4f}'.format(classifier.score(X_test, y_test)))
Accuracy of Naive Bayes classifier on test set: 0.7090
```

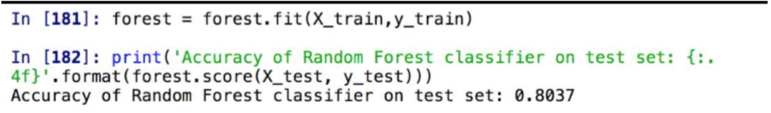
Fig.: Naive Bayes



```
Out[167]:
LogisticRegression(C=1.0, class_weight=None, dual=False,
fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear',
tol=0.0001,
          verbose=0, warm_start=False)

In [168]: print('Accuracy of LOgistic Regression classifier on test set:
{:.4f}'.format(classifier.score(X_test, y_test)))
Accuracy of LOgistic Regression classifier on test set: 0.8012
```

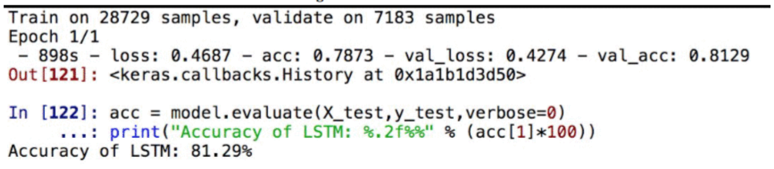
Fig.: Logistic Regression



```
In [181]: forest = forest.fit(X_train,y_train)

In [182]: print('Accuracy of Random Forest classifier on test set: {:.
4f}'.format(forest.score(X_test, y_test)))
Accuracy of Random Forest classifier on test set: 0.8037
```

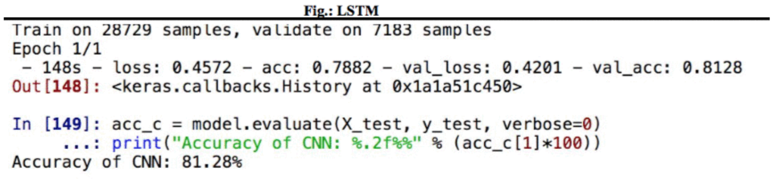
Fig.: Random Forest



```
Train on 28729 samples, validate on 7183 samples
Epoch 1/1
 - 898s - loss: 0.4687 - acc: 0.7873 - val_loss: 0.4274 - val_acc: 0.8129
Out[121]: <keras.callbacks.History at 0x1a1b1d3d50>

In [122]: acc = model.evaluate(X_test,y_test,verbose=0)
     ...: print("Accuracy of LSTM: %.2f%%" % (acc[1]*100))
Accuracy of LSTM: 81.29%
```

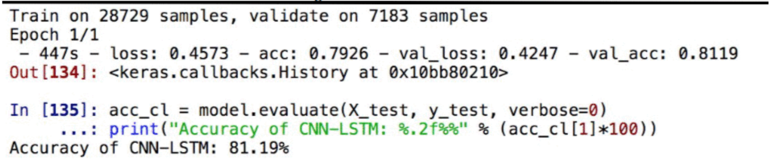
Fig.: LSTM



```
Train on 28729 samples, validate on 7183 samples
Epoch 1/1
 - 148s - loss: 0.4572 - acc: 0.7882 - val_loss: 0.4201 - val_acc: 0.8128
Out[148]: <keras.callbacks.History at 0x1a1a51c450>

In [149]: acc_c = model.evaluate(X_test, y_test, verbose=0)
     ...: print("Accuracy of CNN: %.2f%%" % (acc_c[1]*100))
Accuracy of CNN: 81.28%
```

Fig.: CNN



```
Train on 28729 samples, validate on 7183 samples
Epoch 1/1
 - 447s - loss: 0.4573 - acc: 0.7926 - val_loss: 0.4247 - val_acc: 0.8119
Out[134]: <keras.callbacks.History at 0x10bb80210>

In [135]: acc_cl = model.evaluate(X_test, y_test, verbose=0)
     ...: print("Accuracy of CNN-LSTM: %.2f%%" % (acc_cl[1]*100))
Accuracy of CNN-LSTM: 81.19%
```

Fig.: CNN-LSTM

Figure 10:

Hotel Reviews dataset output for each algorithm

Table 1: Accuracies (in %) of each classification algorithm on 3 datasets

Dataset / Algorithm	IMDB reviews	Hotel reviews	Twitter tweets
Naïve Bayes	71.98	70.9	61.28

Logistic Regression	85.48	80.12	72.90
Random Forests	84.98	80.37	72.44
LSTM	87.58	**81.29**	74.54
CNN	**88.98**	81.28	74.44
CNN + LSTM	88.28	81.19	**74.92**

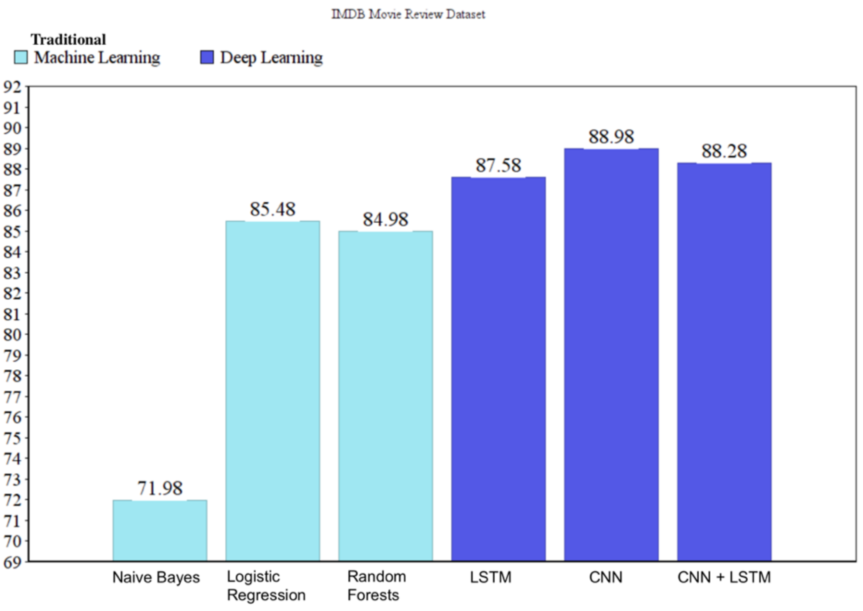


Figure 11: IMDB Movie Reviews Accuracies Comparison for machine learning and deep learning approaches

From the above results, it is evident that deep learning algorithms have an edge over traditional machine learning algorithms. Elaborately because, a deep learning technique learns categories incrementally through its hidden layer architecture, defining low-level categories like letters first then little higher level categories like words and then higher level categories like sentences. Each node in the hidden layer is given a weight that represents the strength of its relationship with the output and as the model develops, the weights are adjusted. Although the traditional classification models using logistic regression and random forests gives a decently good accuracy comparatively, and these have been widely used, but the deeper models have a lot of potential and the fact that they prove to give better accuracies in all three cases cannot be overlooked.

The mascot which gives this advantage to deep learning models in NLP is the *embedding layer,* the fact that featurized representation is possible using embeddings is what outsmarts them compared to traditional machine learning algorithms. Deep learning ensures that they understand the language that they are interpreting by taking word embeddings as input which retain context information and pass them to the hidden layers to learn these features during the training phase itself and thereby making more accurate predictions. Also, the ML algorithms cannot learn the sequence of words which form the sentences. Thus making them less suitable for sentiment analysis tasks.

Additionally, in traditional Machine learning techniques, most of the applied features need to be identified by a domain expert in order to reduce the complexity of the data and make patterns more visible for learning algorithms to work. We need to perform dimensionality reduction to find the best features to pass over the ML algorithms, which is not the case in deep learning. The biggest advantage of Deep Learning algorithms is that they try to learn high-level features from data in an incremental manner thus giving better performance right off the bat. This eliminates the need of domain expertise and hard core feature extraction[8]. Manually designed features are often over-specified, incomplete and take a long time to validate whereas learned features are easy to adapt.

Though, for smaller datasets traditional ML algorithms may outperform deep learning algorithms, because deep learning require sufficiently large amounts of data to work well. Also, CPU utilization is about 10 times more in case deep learning models, thus increasing the time employment.

5 Conclusion

The classification performance of the different models on movie reviews, hotel reviews and tweets gave a rough idea of the utility of these models for sentiment analysis. The performance of the deep learning models was overwhelming. Major reason for this being that the hidden layers understand the copious amount of data exceedingly well and have a deeper semantic understanding of the sequences, plus contextual relationship is retained using word embeddings which boosts the understanding of the model. In fact, they promise to perform much better as seen by the improved accuracies in the above experiments. The complementary advantage is that we can use deep learning models when there is lack of domain understanding for feature introspection, because one has to worry less about feature engineering. Thus, owing to profound contextual understanding deep learning models outperform traditional machine learning algorithms for most sentiment analysis and other NLP based tasks.

6 Future Scope

We can also leverage other deep learning models to improve the results. A lot of research is done in this field leading to development of various techniques for sentiment analysis using Autoencoders, Recursive Neural network, Gated Recurrent Units or using a pretrained GloVe vector which contains trained word embeddings. One can also experiment with the number of epochs and hidden layers to find the best accuracy. Thus with abundance of data, one can improve the efficiency of classification using various disparate deep learning approaches.

References

1. Lei Zhang, Shuai Wang, Bing Liu, F: Deep Learning for Sentiment Analysis: A Survey
2. https://www.kaggle.com/c/word2vec-nlp-tutorial#what-is-deep-learning, last accessed 2018/07/30
3. http://deeplearning.net/tutorial/lstm.html, last accessed 2018/07/30
4. http://colah.github.io/posts/2015-08-Understanding-LSTMs/, last accessed 2018/07/30
5. Aditya Timmaraju, Vikesh Khanna, F: Sentiment Analysis on Movie Reviews using Recursive and Recurrent Neural Network Architectures. Stanford University
6. Leila Arras, Gregoire Montavon, Klaus-Robert Muller, Wojciech Samek, F: Explaining Recurrent Neural Network Predictions in Sentiment Analysis
7. Ye Zhang, Byron C. Wallace, F: A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification". University of Texas, Austin
8. Prerna Singhal, Pushpak Bhattacharyya, F: Sentiment Analysis and Deep Learning: A Survey Prerana Singhal and Pushpak Bhattacharyya. Indian Institute of Technology, Powai.
9. I. Rish, "An empirical study of the naive Bayes classifier", T.J. Watson Research Center
10. J.S. Cramer, "The origins of Logistic Regression", Tinbergen Institute discussion paper
11. Jiawei Han, Micheline Kamber, Jian Pei, Data mining concepts and techniques, The Morgan Kaufmann Series in Data Management Systems (2012), Morgan Kaufmann Publishers, Elsevier
12. https://machinelearningmastery.com/predict-sentiment-movie-reviews-using-deep-learning/, last accessed 2018/07/25
13. Sepp Hochreiter, München, Germany and Jürgen Schmidhuber, Switzerland,F: Long short term memory. Neural Computation archive, Volume 9 Issue 8.
14. http://www.joshuakim.io/understanding-how-convolutional-neural-network-cnn-perform-text-classification-with-word-embeddings/, last accessed on 2018/08/02

15. https://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/ last accessed on 2018/07/30
16. https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d
17. https://medium.com/@kangeugine/long-short-term-memory-lstm-concept-cb3283934359, last accessed 2018/07/30
18. http://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html
19. Maite Taboada, F: Sentiment Analysis: An Overview from Linguistics, Article in Annual Review of Linguistics, DOI: 10.1146/annurev-linguistics-011415-040518