Evaluating Machine Learning Approaches for Sentiment Analysis of Journal logs in Mobile Applications

Sunil Shrestha (PUL074BCT044)

Abstract

This research paper illustrates the findings of a comparative study of various machine learning models for sentiment analysis, with the end goal of deploying it to analyze user journal logs. The primary objective of the research is to find the best model to integrate into a mental health application published on the Android platform. This app uses a sentiment analysis model to perform in-device classification on the user logs and creates visual reports based on the stored data, assisting in self-awareness and management. The suitability of various classification algorithms including Multinomial Naive Bayes (unigram, bigram, and trigram), Complement Naive Bayes (unigram, bigram, and trigram), Bernoulli Naive Bayes, Support Vector Machines (SVM), and Long Short-Term Memory (LSTM) networks were explored and estimated. The same IMDB movie review dataset was used for training and validating each of the models. Afterward, the suitability was evaluated using supervised learning metrics: accuracy and confusion matrices. The findings show significant variation in train-test accuracy among the various models, with the LSTM network ranking top based on metrics score. The paper aims to document a concise summary of the methods and evaluation criteria employed to make the comparison. A detailed comparative analysis is provided, highlighting the distinct strengths and weaknesses of each method, and offering insights into the suitability of these classification techniques for sentiment analysis in mental health contexts. In the end, this study aspires to deliver a fair selection methodology for the best machine learning model for classifying text data on mobile applications.

Introduction

Sentiment analysis is a process used for the identification of emotions in text. It has applications in multiple fields including promoting mental health through self-reflection practices. In developing countries like Nepal, access to conventional therapy sessions is accessible to only a few well-off families while the remaining population is forced to ignore the prevalence of mental health issues due to lack of sufficient mental health professionals [1]. The solution to these challenges requires a revolutionary innovative solution that must be easily accessible and effective for addressing the problem [2]. In this paper, we illustrate the possibility of employing a machine learning model for sentiment analysis to perform continuous diagnosis and willful tracking for self-reflection.
This study assists in the choice of the best sentiment analysis methodology for mental health by comparing several machine learning models based on well-known metrics. The machine learning models used in the study use supervised learning to train the models on labelled text data from the IMDB review dataset [3]. These models are trained to classify emotional sentiments by analyzing written journal entries. The goal of the study is to fairly select the most suitable model for integrating in a user-facing application. This application will classify the user's log data using the trained model and provide insight into the user's mental health. The application will also use sentiment analysis to track and store mood data. This data will facilitate self-analysis and increase awareness.

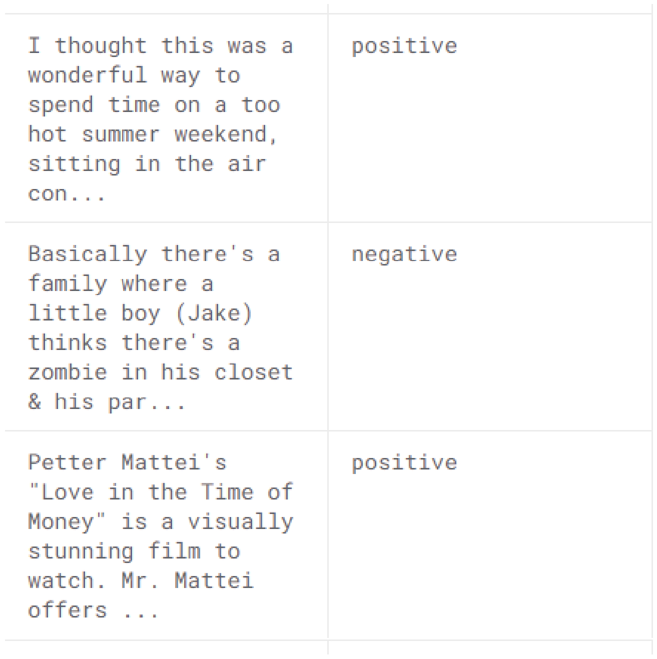
The models used include Multinomial Naive Bayes. We use its unigram, bigram, and trigram variations. Complement Naive Bayes is also used with its unigram, bigram, and trigram variations. Other models are Bernoulli Naive Bayes, Support Vector Machines (SVM), and Long Short-Term Memory (LSTM) networks. These models were trained on a dataset of movie reviews. This dataset was chosen for its size. It also contains both positive and negative examples. We used performance metrics to evaluate the models. These metrics are accuracy and confusion matrices.

The results show different levels of performance. The LSTM model shows the best result. The paper discusses the method used. It also discusses the metric used for selection. The analysis shows the strengths and limitations of each model. This helps understand which model is best for applications that need good text analysis. This paper provides guidelines to select appropriate models for applications that require finding the sentiments in user text accurately.

Methodology

Sentiment analysis, which also is known as opinion mining, is the computational process of identifying whether a piece of text expresses a positive, negative, or neutral sentiment [4]. The sentimental analysis model works by accepting text data like the user's journal logs as input and predicts the related sentiments as a score between 0 to 1, 1 being very positive and 0 very negative. Thus, the final sentiment is determined based on the score that corresponds to the range in the spectrum table.

A deep learning model is trained using supervised learning to classify the text into sentiment by using the labeled IMDB movie reviews dataset. The dataset consists of 50,000 rows with two columns i.e. review and sentiment. The dataset is balanced as it contains 25,000 texts with positive sentiment and another 25,000 texts with negative sentiment. The first column contains sentences and the second column shows whether that sentence is positive or negative [3]. The sample of data can be seen in the figure below.



I thought this was a wonderful way to spend time on a too hot summer weekend, sitting in the air con...	positive
Basically there's a family where a little boy (Jake) thinks there's a zombie in his closet & his par...	negative
Petter Mattei's "Love in the Time of Money" is a visually stunning film to watch. Mr. Mattei offers ...	positive

Figure 1. Sample of IMDB review dataset

Theory

I. Multinomial Naive Bayes

Multinomial Naive Bayes Classifier is the type of probabilistic classifier that works with mostly text-related data to classify them to multiple classes based on Bayes Theorem [5]. It calculates the probability of each tag involved in each class and classifies the best result with the highest probability based on the final result [6].

Unigram model: This is the type of model where a single sequence of words is used. By using single word occurrence the probability is calculated and the same for each word present in a whole. Thus, this creates the significance of a single word in the result.

Bigram Model: This is the type of model where a double sequence of words is used. By using the combination of the same double words, the probability of occurrence is calculated and the same for other word combinations as a whole.

Trigram Model: The Trigram model means dealing with a triple sequence of words. This means there would be combinations of three words to signify the importance of each word in the presence of the other two. This results in a higher possibility of finding the minute details of our language characteristics for text-based analysis.

II. Complement Naive Bayes

In Complement Naive Bayes we calculate the probability of an item belonging to all classes rather than a single class in Multinomial Naive Bayes [7]. Thus, it is known as Complement Naive Bayes, an adaptation of Multinomial Naive Bayes. Complement Naive Bayes is usually used for imbalance sets. Even though our data are not imbalanced, we tried fitting the data to the model to analyze the result as the previous model was unable to classify neutral data very nicely.

III. Bernoulli Naive Bayes

Bernoulli Naive Bayes is one of the types of Naive Bayes classifier that is especially suited for binary/boolean features. It assumes each feature to be a binary-valued (0/1) variable. [8]

IV. SVM Classifier

Support Vector Machine(SVM) is the machine learning technique for classification using hyperplanes. The algorithm tries to find the best hyperplane that could dissect the different types of data. The data is cleaned, stopwords are removed and PorterStemmer is used for stemming and then CountVectorizer is used.

V. Long Short Term Memory (LSTM):

LSTM networks are a type of recurrent neural network (RNN) designed to solve the vanishing gradient problem in standard RNNs. They achieve this by introducing a special architecture with three primary components: forget gate, input gate, and output gate. [9]

VI. K-fold Cross Validation

K-fold cross validation in machine learning cross-validation is a technique for evaluating predictive models in data science. It involves splitting the training dataset into k subsets or folds, where each fold is used as the validation set in turn while the remaining k-1 folds are used for training. K-fold cross validation is used to evaluate the performance of a model with limited training datasets. In this method, data is divided into k subsets and k numbers of models are trained each using (k-1) subsets for training and 1 subset for validation.

Results and Discussion

Table 1 illustrates the various accuracy scores obtained by training and testing the machine learning models on the IMDB review dataset.

Algorithm	**N-gram**	**Test Accuracy**	**Training Accuracy**
Multinomial Naive Bayes	Unigram	61.80%	75.55%
	Bigram	50.50%	92.81%
	Trigram	43.28%	93.56%
Complement Naive Bayes	Unigram	64.05%	79.25%
	Bigram	50.56%	95.18%
	Trigram	39.09%	93.13%
Bernoulli Naive Bayes	Bigram	47.7%	58.72%
SVC	-	85.44%	96.20%

LSTM	-	87.28%	92.56%

Table 1. Table of train-test accuracies of various machine models on the dataset

Multinomial Naive Bayes - Unigram

The accuracy of the multinomial naive Bayes on the unigram model is 61.8% for test sets and 75.55% for training sets. With some new data from the perspective of the user used for analysis, the performance was good as it produced the correct result for all three sentences simulating user testing. This clearly shows that it worked very averagely even if it was the input it had not seen before or the input it had learned before. So, the performance was not satisfying for our application.

Multinomial Naive Bayes - Bigram

Using the Bigram model the output of the model is very average as test sets result averaged 50.05% while the performance with previously seen data was outstandingly 92.81%, which clearly showed that the learning process was good enough that further if similar input was provided result may be correct but for new data, the result is just satisfactory so using this model would degrade the performance of our whole application. With some data we have provided for user testing and a confusion matrix it clearly shows that the model finds it hard to evaluate neutral texts and ends up classifying it as positive.

Multinomial Naive Bayes - Trigram

The result shows there is better accuracy in what the model has learned, that is 93.56% with respect to 43.28% on new different datasets which is far from our mark of model for better application. Similar to the Bigram model here we could not classify neutral sentences correctly which reduced accuracy on test sets drastically.

Complement Naive Bayes - Unigram

As the below figure shows the result is not satisfactory in this model as well with an average of 64.05% for test sets while somewhat improved 79.25% for training sets using the unigram model. However, the difference between the multinomial is felt after analyzing the confusion matrix that shows that neutral data are somewhat classified properly. Finally, analyzing the result we get that as in Multinomial Naive Bayes classifying neutral to positive, these models classified it as negative.

Complement Naive Bayes - Bigram

By using the Bigram model for Complement Naive Bayes, results on training sets are improved drastically but the results on test sets which are usually important are decreased to 50.56%. Thus, the performance of this can also be concluded as unsatisfactory even though results on the same user dataset are the same.

Complement Naive Bayes - Trigram

Using the trigram model the test results are drastically reduced to a very poor 39.09% so, the model is not used while the train results have improved to 93.13% which would not affect much more on the performance of our final application. The accuracy decrease is also clearly visible in the user data provided where it classified all as negative when one was positive, one was negative, and the last one neutral. Thus, the results using the Complement Naive Bayes with Trigram model have also resulted in unsatisfactory results.

Bernoulli Naive Bayes

As the Bigram model was best performing for both the Naive Bayes Classifier, we tried it with the Bernoulli Naive Bayes Classifier, the results were very unsatisfactory with an accuracy of less than 47.77% for train sets and 58.72% for train datasets. The conclusion from the confusion matrix can be that the classifier now is classifying large data to neutral only which is also visible from the simple three sentences we provided.

Support Vector Machine

SVM classifier with linear kernel is used for training. Using this method, the training set accuracy reached 96.20% while the test set accuracy was 85.44%. The confusion matrix revealed that it could classify reviews with positive sentiments better but it struggled with negative reviews.

LSTM Classifier

The LSTM model obtained the best results among the different machine learning models in the experiments. The accuracy in the training set is 92.56% and the accuracy in the test set was 87.20%.

The first step in training of LSTM network is preprocessing of data in which the text is converted to lowercase, punctuations are removed. Then, tokenization is done and a word-index mapping dictionary is created. The words and labels are encoded. The text is represented as a sequence and the sequences are padded and truncated to make them of equal length.

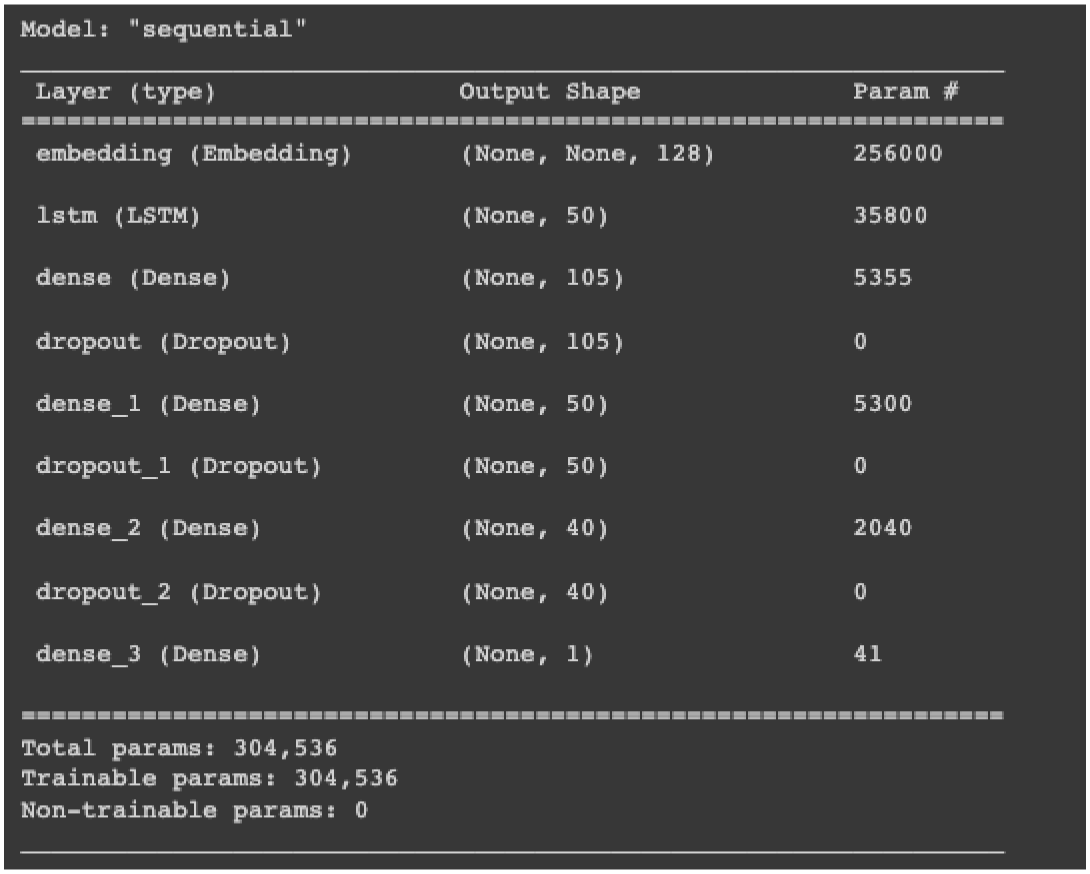
After that, the LSTM network is designed using Keras API of Tensorflow library to build a model [Figure 2]. The embedding layer is used first to generate the embedding as dense vectors of fixed size. Then, hidden layers and output layers are added. A random search is performed to search the best hyperparameters for the model. Binary cross entropy loss function and Adam optimizer were used.

K-fold cross validation results

The LSTM classifier model was again trained and validated using K-fold cross validation. The results below in table 2 clearly show training and test accuracy in different folds of k. The training accuracy maxed at k=3 which shows some overfitting, it kept decreasing slightly with higher fold value. On the other hand, accuracy on the test set was maximum at 88.002% for k=5.

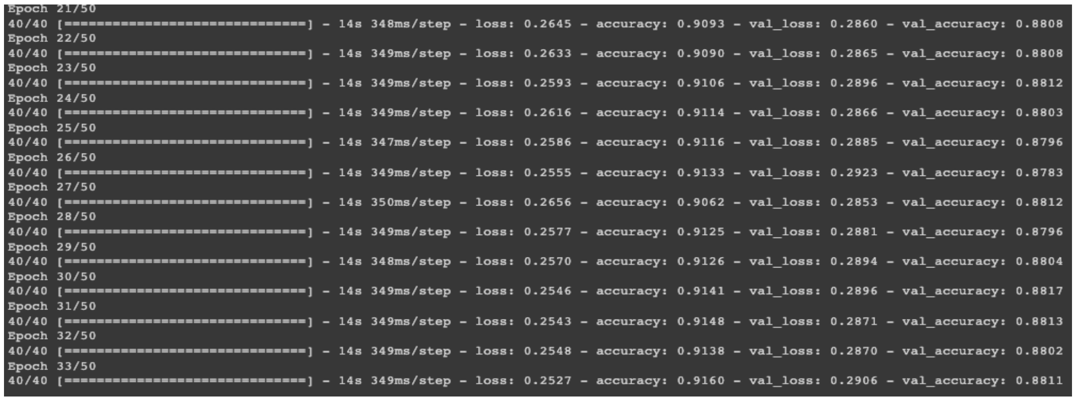
K-fold	**Training Accuracy**	**Test Accuracy**
3	92.006%	87.80%
5	91.57%	88.002%
10	91.43%	87.94%

Table 2. Table of train-test accuracies of LSTM model on with different k-folds



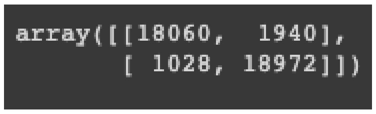
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 128)         256000

 lstm (LSTM)                 (None, 50)                35800

 dense (Dense)               (None, 105)               5355

 dropout (Dropout)           (None, 105)               0

 dense_1 (Dense)             (None, 50)                5300

 dropout_1 (Dropout)         (None, 50)                0

 dense_2 (Dense)             (None, 40)                2040

 dropout_2 (Dropout)         (None, 40)                0

 dense_3 (Dense)             (None, 1)                 41

=================================================================
Total params: 304,536
Trainable params: 304,536
Non-trainable params: 0
_________________________________________________________________
```

Figure 2: Sequential model summary



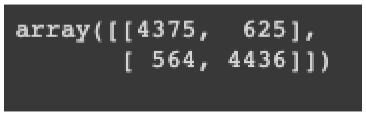
```
Epoch 21/50
40/40 [==============================] - 14s 348ms/step - loss: 0.2645 - accuracy: 0.9093 - val_loss: 0.2860 - val_accuracy: 0.8808
Epoch 22/50
40/40 [==============================] - 14s 349ms/step - loss: 0.2633 - accuracy: 0.9090 - val_loss: 0.2865 - val_accuracy: 0.8808
Epoch 23/50
40/40 [==============================] - 14s 349ms/step - loss: 0.2593 - accuracy: 0.9106 - val_loss: 0.2896 - val_accuracy: 0.8812
Epoch 24/50
40/40 [==============================] - 14s 349ms/step - loss: 0.2616 - accuracy: 0.9114 - val_loss: 0.2866 - val_accuracy: 0.8803
Epoch 25/50
40/40 [==============================] - 14s 347ms/step - loss: 0.2586 - accuracy: 0.9116 - val_loss: 0.2885 - val_accuracy: 0.8796
Epoch 26/50
40/40 [==============================] - 14s 349ms/step - loss: 0.2555 - accuracy: 0.9133 - val_loss: 0.2923 - val_accuracy: 0.8783
Epoch 27/50
40/40 [==============================] - 14s 350ms/step - loss: 0.2656 - accuracy: 0.9062 - val_loss: 0.2853 - val_accuracy: 0.8812
Epoch 28/50
40/40 [==============================] - 14s 349ms/step - loss: 0.2577 - accuracy: 0.9125 - val_loss: 0.2881 - val_accuracy: 0.8796
Epoch 29/50
40/40 [==============================] - 14s 348ms/step - loss: 0.2570 - accuracy: 0.9126 - val_loss: 0.2894 - val_accuracy: 0.8804
Epoch 30/50
40/40 [==============================] - 14s 349ms/step - loss: 0.2546 - accuracy: 0.9141 - val_loss: 0.2896 - val_accuracy: 0.8817
Epoch 31/50
40/40 [==============================] - 14s 349ms/step - loss: 0.2543 - accuracy: 0.9148 - val_loss: 0.2871 - val_accuracy: 0.8813
Epoch 32/50
40/40 [==============================] - 14s 349ms/step - loss: 0.2548 - accuracy: 0.9138 - val_loss: 0.2870 - val_accuracy: 0.8802
Epoch 33/50
40/40 [==============================] - 14s 349ms/step - loss: 0.2527 - accuracy: 0.9160 - val_loss: 0.2906 - val_accuracy: 0.8811
```

Figure 3: Accuracy and loss of model after each training epoch



```
array([[18060,  1940],
       [ 1028, 18972]])
```

Figure 4: Confusion matrix for training set



```
array([[4375,  625],
       [ 564, 4436]])
```

Figure 5: Confusion matrix for test set

Conclusion

Thus, from above experiment on different methods of sentimental analysis and analyzing its training and test set accuracy. We find out LSTM works better than the rest of others with better accuracy on both types of data known and unknown. The SVC method also provided good accuracy but was biased with the positive set of sentiment in data. Thus, LSTM was the chosen final method for sentiment analysis. Then, we performed different values of k-fold validation. The k-fold validation was performed for k equals to 3, 5 and 10. And, we can clearly see from the k-fold validation table, k-fold validation with k=5 had the best result among us so we used LSTM with k=5 for sentimental analysis purposes.

References

[1] Luitel, Nagendra P et al. “Process evaluation of a district mental healthcare plan in Nepal: a mixed-methods case study.” BJPsych open vol. 6,4 e77. 28 Jul. 2020, doi:10.1192/bjo.2020.60

[2] Hanna, B. S. H., & Hanna, A. S. H. (2022). Role of Artificial Intelligence in Mental Wellbeing: Opportunities and Challenges - SciAlert Responsive Version. Science Alert. Retrieved April 10, 2022, from https://scialert.net/fulltext/?doi=jai.2022.1.8

[3] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).

[4] Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. https://doi.org/10.1561/9781601981516

[5] Kibriya, A. M., Frank, E., Pfahringer, B., & Holmes, G. (2004). Multinomial naive bayes for text categorization revisited. In Lecture notes in computer science (pp. 488–499). https://doi.org/10.1007/978-3-540-30549-1_43

[6] T. M. Mitchell, Machine Learning (1997), McGraw-Hill Science/Engineering/Math (pp. 177-178)

[7] Seref, B., & Bostanci, E. (2018). Sentiment Analysis using Naive Bayes and Complement Naive Bayes Classifier Algorithms on Hadoop Framework. 2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), 1–7. https://doi.org/10.1109/ismsit.2018.8567243

[8] Singh, G., Kumar, B., Gaur, L., & Tyagi, A. (2019). Comparison between Multinomial and Bernoulli Naïve Bayes for Text Classification. *2019 International Conference on Automation, Computational and Technology Management (ICACTM)*, 593-596.

[9] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term memory. Neural Computation, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735