ISSN 2616-5775 Vol. 5, Issue 8: 81-89, DOI: 10.25236/AJCIS.2022.050812

Automatic Emotion Recognition of Text Stories Based on Deep Learning

Xinyi Yao

Shanghai High School International Division, Shanghai, China
cynthiayao11@outlook.com

Abstract: *Due to the immense amounts of texts on the internet and the qualitative nature of human sentiment and the characteristics of machine learning and deep learning algorithms, they are potential candidates to be appli-ed in textual sentiment analysis. To compare the effectiveness of different algorithms, processed data using TF-IDF is input into different algorithms respectively, and the accuracy scores of the trials using the identical data-set are recorded for comparison. It turns out that the Extra Trees classifier and the Random Forest classi-fier performed the best among machine learning algorithms, suggesting the significance for reducing overfitting in this specific task given that the less overfitting-proof Decision Tree has performed worse. LSTM has a better ac-curacy score than CNN, though it is to be noted that the former runs significantly slower than the latter, indicating efficiency to be a potential topic to be considered.*

Keywords: *Machine Learning, Emotion Recognition of Text, Deep Learning*

1. Introduction

The proliferation of the internet has made possible the current availability of social network platforms, where those who have access can publish texts that are made public to other users. There exists too many of emotion-rich texts such that it is hardly possible for all to be reviewed by other humans. Crucial information in these texts is thus likely to be overlooked. Furthermore, emotional value of texts on social media platforms is even more difficult for machines to realize also because of the significance of context of the texts. With the amount of available data overflowing on the internet, it is thus harder for the computer to detect subtle relationships in human language before extracting the correct features. The real-life applications of emotion classification include many fields such as healthcare, online education, e-commerce, marketing, and sociology.

Emotion is an important factor of human consciousness, and its subjectivity as well as qualitative nature has made it more difficult for rational procedures to detect it successfully. To classify the emotional context in texts, NLP, or Natural Language Processing, is an effective tool. Developed with concerns for computers to comprehend human language, NLP is a branch of artificial intelligence which combines computational linguistics and machine learning to enable computers to perceive the meaning as well as sentiment within human speech and text. Understanding the text, to some extent at least, is crucial to grasping its sentimental opinion before further classifying them according to more specific branches of emotions.

Past studies on text classification have achieved numerous results on emotion detection. As for the representation of text documents before text classification, feature engineering is a frequently used method which builds a new set of features that include higher-level concepts such as topics. Some popular tools for extracting emotion features, or GPEL's (general purpose emotion lexicons), would be WordNet-Affect, EmoSenticNet, NRC word-emotion lexicon, etc. However, GPEL's detect the emotion in words poorly in that the memberships of words in GPEL are binary and contrary to the nature of emotions, making them function limitedly in feature extraction for such purposes. The study "Lexicon based feature extraction for emotion text classification" by Bandhakavi et.al addresses this weakness by utilizing domain specific emotion lexicon, or DSEL, to directly analyze word and phrase-level emotions and classify them into classes via machine learning. Furthermore, unigram mixture models are also put into use, enabling the ability to quantify both emotionality and neutrality of words and represent texts along with their emotional attributes.

Aside from feature extraction, popular classifiers used would include logistic regression and SVM.

ISSN 2616-5775 Vol. 5, Issue 8: 81-89, DOI: 10.25236/AJCIS.2022.050812

"Sentiment analysis of twitter data related to Rinca Island development using Doc2Vec and SVM and logistic regression as classifier" by Universitas Indonesia is an example. Logistic regression and SVM are used as the classifiers to form an overview of Indonesian Twitter users 'opinions on a controversial domestic issue. Comparatively assessing, SVM has a higher accuracy rate than logistic regression overall in this study.

Another frequently used machine learning classifier would be random forests, or RF. "An Approximation Method for Fitted Random Forests" by Sai K Popuri points out that its main advantage is the reduction in its variance of the forecast. However, the size of the model would vary and, in cases with millions of data points and numerous features, the size of fitted RF models might exceed the limits of available space. This further obstructs the ability of such models to be downloaded to small devices on-demand. Thus, while RF has a high accuracy, there are also challenges to reduce its size without losing much trustworthiness of its results.

CNN, or Convolutional Neural Network, is also a commonly seen architecture used for similar purposes. In "TextConvoNet: A Convolutional Neural Network based Architecture for Text Classification" by Soni et.al, a CNN specified towards text classification is evaluated. It captures the n-gram features from the text and the inter-sentence n-gram features, made possible by the alternate input representation for the incoming data. As an example of CNN's, TextConvoNet is proven to be effective in its classification for binary and multiple classes.

2. Method

2.1. Data Pre-processing

Prior to training the model with data, it is required for the data to be pre-processed; it needs to be converted to a format that fits the required input of the model, and excess data should be cleaned away to keep is from affecting the accuracy.

2.2. Term Frequency-Inverse Document Frequency (TF-IDF)

Term Frequency-Inverse Document Frequency, or TF-IDF, is the calculation of the relevance of a word in a corpus compared to that of a text. The terminology term frequency stands for the number of instances which a given word appears in a specific document. Document frequency, on the contrary, tests the frequency of the given word not in one document, but a collection of documents which forms the so-called corpus collection. Upon utilization, the document frequency is used in the form of inverse document frequency, which is $\mathrm{idf(t)} = \log(\frac{\mathrm{N}}{\mathrm{df(t)}})$, with N being the corpus. TF-IDF combines these two calculations and determines the significance of a term t is to a single text d in a larger series N as

$$tf - idf(t,d) = tf(t,d) \times idf(t) \quad (1)$$

2.3. Conventional Machine learning Classifier

The Naïve Bayes is based on Bayes Theorem has three methods that are *Multinomial, Gaussian and Bernoulli* Naïve Bayes. In these methods, the *Bernoulli* Naïve Bayes and the *Multinomial* Naïve Bayes can be applied to discrete variables, and the *Gaussian* Naïve Bayes can only be applied to continuous data. These three strategies can be seen as probabilistic models.

The Decision Tree Classifier can be classified as ID3, C4.5 and Cart. But Cart is the best method among the three. It is using binary trees, however, the others are using multi-way trees. These three methods are different due to their different classification ways that are information gain, information gain ratio and Gini coefficient.

The random forest classifier is a supervised machine learning algorithm commonly used for classification problems because of its flexibility and convenience in applying. It consists of multiple decision trees and makes use of randomness in order to reduce overfitting and increase accuracy. Randomness is embedded in that the algorithm creates each tree from a random selection of data and one prediction is retrieved from each tree before the best solution is selected by votes among the trees.

When called, the random forest algorithm first selects a random number of features among all features given, and then chooses one most significant feature as the root node according to the highest information gain per node. Then, the root node splits into smaller nodes and eventually forms a forest of decision

ISSN 2616-5775 Vol. 5, Issue 8: 81-89, DOI: 10.25236/AJCIS.2022.050812

trees. Finally, for classification problems, the result is calculated by choosing the most popular result among all trees. For regression problems, on the contrary, the average of all results from the trees is calculated to be the final overall result.

The extra forest is improved on random forest. It is different from random forest because of its ways of classification. The extra forest is preciser and more random than random forest. The extra forest uses entire datasets to build a tree while the random forest uses a random subset of the entire datasets to produce a tree. The extra forest select character randomly from the whole character set and the random forest choose character from the subset of all characters. Besides, when the extra forest is built, its branch is dividing randomly.

2.4. Convolutional Neural Network (CNN)

Convolutional Neural Network, or CNN, is another deep learning architecture that can be applied to solving NLP problems. As its name suggests, the crucial operation of the entire algorithm is convolution, in which data in a source matrix is passed through a convolution filter, or a kernel. It samples the pixels on the source image in specified strides, or step lengths, and sums up the element-wise dot product of the values. The formed matrix is then passed down and computed again.

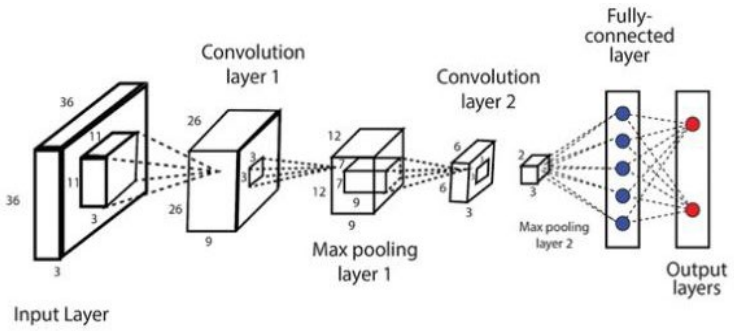

Figure 1: The procedure of CNN.

CNN is effective due to its property of sensing special relationships. Because kernels often overlap with each other during convolution, the pixels in the overlapping parts are included in multiple convolution filters, allowing the relationship between similar pixels to be found more easily.

2.5. Long Short-Term Memory Network (LSTM)

Long Short-Term Memory Network, or LSTM in short, is a variation of recurrent neural network (RNN) which is effective especially in predicting outputs from long sequences of data, such as text inputs in the form of sentences and stock price changes over a time period.

It monitors the chronological sequences of the data and attempts to capture their long-term relationships. The main characteristics that enable this feature is its feedback loop and its unique memory cells that stores information from earlier layers and puts them into use after a timespan, improving the issue faced by traditional RNN's. This noble improvement in the LSTM allows it to predict outputs using long sequences of data without inducing a vanishing gradient, a major weakness in prior RNN models.

Contrary to the RNN, which stores previous feedback in its memory cells for a short period of time, the LSTM preserves earlier output for a longer period and has finer control over the parts of the memory for being passed onward or forgotten. LSTM's hidden layers are consisted of gated cells, which control whether the portion of the output gets passed through or discarded.

3. Materials

3.1. Constructing Dataset

The database used to train this system is a collection of posts on the Chinese social media platform Weibo, with each post labelled an index number in the front. That is to say, the initial data is in the format "label, review".

The Python package jieba, which provides text segmentation functions for Chinese words, is used

ISSN 2616-5775 Vol. 5, Issue 8: 81-89, DOI: 10.25236/AJCIS.2022.050812

here. Using the jieba.lcut() function, text data is segmented from sentences into words separated by a single space. Then, confounding characters are cleansed off by filtering the text according to a list of stop-words. To make the process of calling both the text and label separately more convenient, the data is reformatted as "label \t review" and written into a new txt file ready to be retrieved.

The pre-processed text and label are inputted as parameters respectively into the TF-IDF function. Prior to calculating the TF-IDF, the text and label data are together separated using train_test_split (text, label, test_size=0.02, random_state = 101), calling the train_test_split() function imported from sklearn. model_selection. The purpose of this is to allow the test and train data to be retrieved separately for later uses, and for training purposes the random_state parameter needs to be defined to keep the data from changing every time train_test_split() is called. CountVectorizer ().fit_transform () imported from sklearn_feature_ extraction.text converts the text into matrices. Then, using the TfidfTransformer (use_idf=True).fit_transform () function from the same package, the transformer is fit using the matrix-form train and test data. The transformed X returned is then returned from the TF-IDF function, which is later retrieved in the main function as train and test data for algorithms including Random Forest, Decision Tree, Extra Trees, Logistic Regression, Multinomial Naïve Bayes and Gaussian Naïve Bayes.

Data preprocessing for CNN and LSTM, on the other hand, requires another step. The data needs to be converted to One Hot encoding for the algorithm to process it properly. One Hot encoding is the process of converting categorical data variables so they can be provided to machine learning algorithms for an improved accuracy in predictions. For training the model, the number labels in the pre-processed dataset are also converted into One Hot encoding for better predictions. This is achieved by importing OneHotEncoding from sklearn. preprocessing and calling the function OneHotEncoder (sparse=False). fit_transform(y).

3.2. Machine Learning Algorithms

3.2.1. Random Forest Classifier

One of the machine learning algorithms applied is the Random Forest Classifier. Upon implementation, the algorithm is called via first importing RandomForestClassifier from sklearn.ensemble. Then the classifier is created using the RandomForestClassifier function with parameters n_estimators=100, max_depth=20, random_state=0. Here, the parameter n_estimators represents the number of trees that the classifier uses, and max_depth is the maximal depth for the trees to reach during calculation. Clarifying the random_state parameter ensures that the random assignment the algorithm generates every time the code is run stays consistently random, which is necessary for the results to remain stable. X and y are passed down from prior processing via TfidfTransformer (use_idf=True).fit_transform(X).

After the model is trained and ready, its accuracy is tested through an attempt to predict the y value given the test data for X and comparing the predicted y and actual y test.

3.2.2. Decision Tree Classifier

The Decision Tree Classifier is another machine learning algorithm used here. The package sklearn.tree is imported here, and the classifier is created using the function DecisionTreeClassifier(). It is then conveniently trained and tested, returning the predicted score. The accuracy score is calculated using this score and returned to the main function.

3.2.3. Extra Trees Classifier

The Extra Trees algorithm is imported from sklearn.ensemble and created using the function ExtraTreesClassifier(). Similarly, this algorithm is also trained using the processed x and y values passed on by the TF-IDF function and returns the accuracy score of its predictions to the main function.

3.2.4. Logistic Regression Classifier

The Logistic Regression Classifier with parameter random_state=0 is imported from the package sklearn.linear_model and trained using processed data from the TF-IDF function earlier. The predicted results are calculated via the predict() function, and the accuracy score of this result is returned to the main function.

3.2.5. Multinomial Naïve Bayes Classifier

The Multinomial Naïve Bayes Classifier is imported via the sklearn.naive_bayes package and trained using process data similarly as in other machine learning algorithms. The difference here, however, is

ISSN 2616-5775 Vol. 5, Issue 8: 81-89, DOI: 10.25236/AJCIS.2022.050812

that the y values used are transformed into 1-D using y_train.argmax (axis=1). The reason for this is that the y parameter in MultinomialNB () is required to be a set of values instead of the matrix returned by the TF-IDF code's numpy.utils.to_categorical(LabelEncoder().fit_transform() function, which takes place during the earlier pre-processing of the data. The output of the predicted y value, or y_pred, on the contrary, does not need to be converted into a matric once again upon calculating the accuracy score, because it is of the same type as y_test.

3.2.6. Gaussian Naïve Bayes Classifier

Another naïve bayes algorithm used is the Gaussian Naïve Bayes Classifier, imported from sklearn.naive_bayes. Similarly, this algorithm is created with the GaussianNB() function and then trained using the data that has been pre-processed in the TF-IDF function. Here, the x values need to be converted into arrays using toarray() because of the input requirements of the Gaussian Naïve Bayes Classifier. Similar to that of the Multinomial Naïve Bayes Classi-fier, the y values used for training also need to be transformed into 1-D with y_train.argmax (axis=1). The accuracy score of the algorithm, sklearn.metrics.accuracy_score(), is then returned to the main function.

3.2.7. Deep Learning Models

The two deep learning models used are LSTM and CNN. The package imported are, respectively, lstm and cnn. Prior to training the models, the data that is cut earlier is processed again using a different approach compared with that of machine learning algorithms. The text is first input into the function pd.get_dummies(), during which it forms a data frame of indicator variables from string input. Then the data frames are processed by the Tokenizer function imported from tensorflow.python.keras.preprocessing.text and are made into sequences according to TF-IDF before they are pad as well, which is filling up the blank spaces to make all sequences equal in length. This process enables the model to take in the data and train itself.

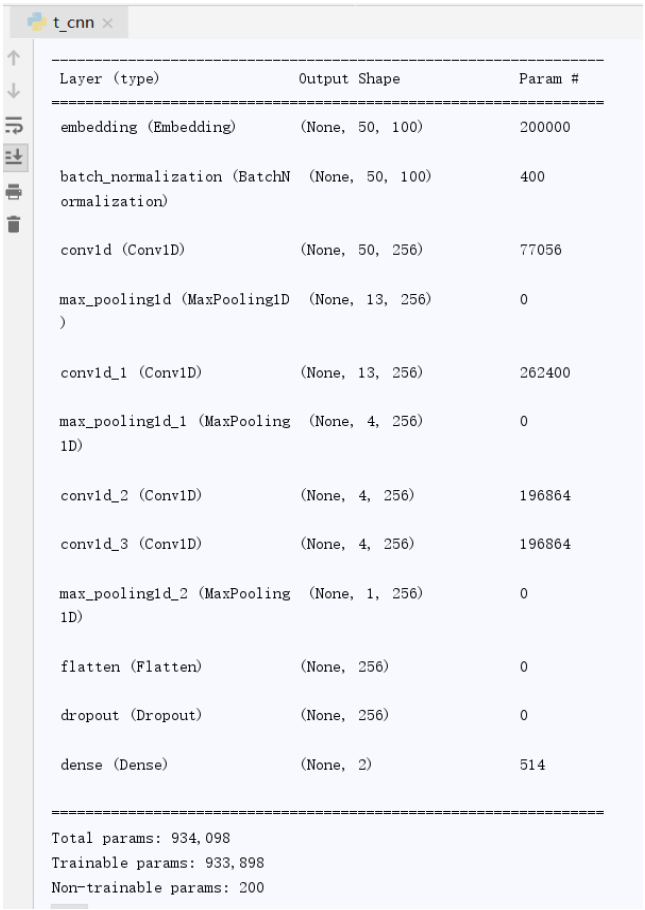

Figure 2: The structure of CNN.

ISSN 2616-5775 Vol. 5, Issue 8: 81-89, DOI: 10.25236/AJCIS.2022.050812

rnn_lstm

```
9622/9622 [==============================] - 61s 6ms/sample - loss: 0.1680 - accuracy: 0.9630 - val_loss: 0.3799 - val_accuracy: 0.8960
Model: "sequential"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding (Embedding)        (None, 50, 100)           200000

 batch_normalization (BatchN  (None, 50, 100)          400
 ormalization)

 lstm (LSTM)                  (None, 100)               80400

 dropout (Dropout)            (None, 100)               0

 dense (Dense)                (None, 100)               10100

 batch_normalization_1 (Batc  (None, 100)              400
 hNormalization)

 dense_1 (Dense)              (None, 2)                 202

=================================================================
Total params: 291,502
Trainable params: 291,102
Non-trainable params: 400
```

Figure 3: The structure of LSTM.

The structure of the two models are shown below. The sequence of layers is optimized after multiple trials to reach the highest test accuracy.

4. Result Analysis

The accuracy score ranking of the 6 machine learning algorithms are shown here. Extra Trees seems to have the highest accuracy with Random Forest and Logistic Regression closely behind, while Gaussian Bayes has the lowest score.

One of the possible reason for Extra Trees to outperform the other algorithms is that Extra Trees has a decent resistance against overfitting. As its name suggests, Extra Trees is consisted of multiple decision trees, and the number of independent trees reduce the significance of overfitting in each one and balancing the forest as a whole. Its advantage is due to its characteristic of dropping out random features and its bagging method as well, which further keeps the model from overfitting and ensures accuracy.

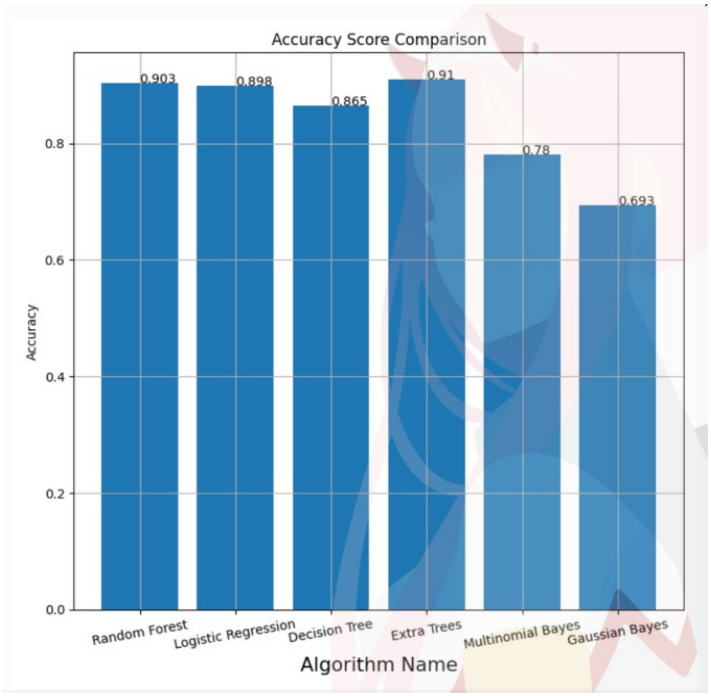

Figure 4: The comparison of machine learning.

ISSN 2616-5775 Vol. 5, Issue 8: 81-89, DOI: 10.25236/AJCIS.2022.050812

This characteristic of Extra Trees is even more randomized compared to that of Random Forest and thus reduces variance, a possible reason to why Extra Trees outrun Random Forest in their accuracy.

Logistic Regression falls closely behind Random Forest by a small difference in their accuracy scores. A potential cause for this would be that Logistic Regression classification is specialized to perform effectively when the data is linearly separable, or when there exists a variable that can act as the threshold between the two outcomes. For this dataset, however, it is unlikely for a single variable that manipulates the conclusion of the classification, since the emotion in a sentence usually depends on the entire sentence conclusively. The lack of a singular decisive variable might be why Logistic Regression fail to operate effectively in this case.

Gaussian Bayes has a poor performance on this dataset possibly because of the mismatch between the distribution of the given data and the desired input for the algorithm. Gaussian Bayes assumes Gaussian distributed input data, and performs its predictions based on this assumption. The dataset in this case, however, is matrices of numbers that represent sentences. It would be unlikely for this data to be distributed the way this algorithm desires.

Contrary to Gaussian Bayes, Multinomial Bayes has a significantly higher score, though not among the highest scores in all algorithms. One possible reason for this would be that Gaussian Bayes demands continuous data, whereas Multinomial Bayes does not; in this case, the input data given is discrete. Thus, Gaussian Bayes' requirement for continuity and distribution of input data might be the major factor to its accuracy score.

The deep learning algorithm evaluated are also compared. Though theoretically deep learning algorithms are to outperform prior ones, this turns out to not be the case. This might be due to the computational abilities of the hardware used, or the lack of artificial optimization to the layers of deep learning algorithms. The resulting accuracy scores for the testing dataset, sequentially 0.825 and 0.8625 for CNN and LSTM, seemingly point out that CNN is slightly less effective than LSTM here. The reason for this difference is likely to lie in the structural difference between CNN and LSTM, in that LSTM has structures that function to recall data from earlier layers in the current layer whereas CNN does not. In other words, LSTM perform better at connecting words within a sentence sequentially.

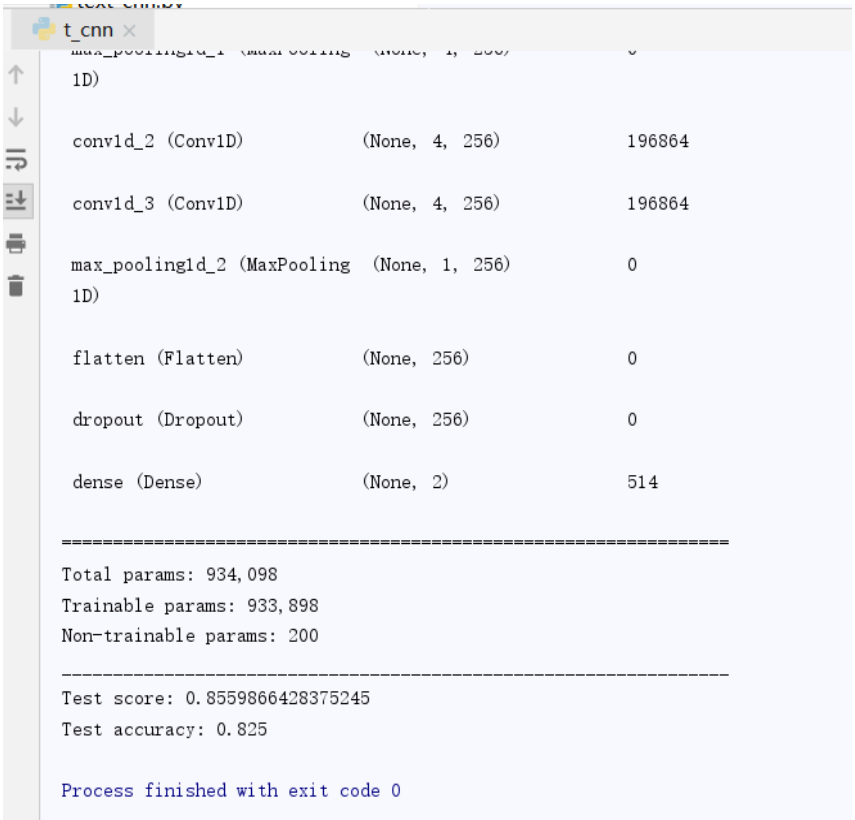

Figure 5: The accuracy of CNN.

ISSN 2616-5775 Vol. 5, Issue 8: 81-89, DOI: 10.25236/AJCIS.2022.050812

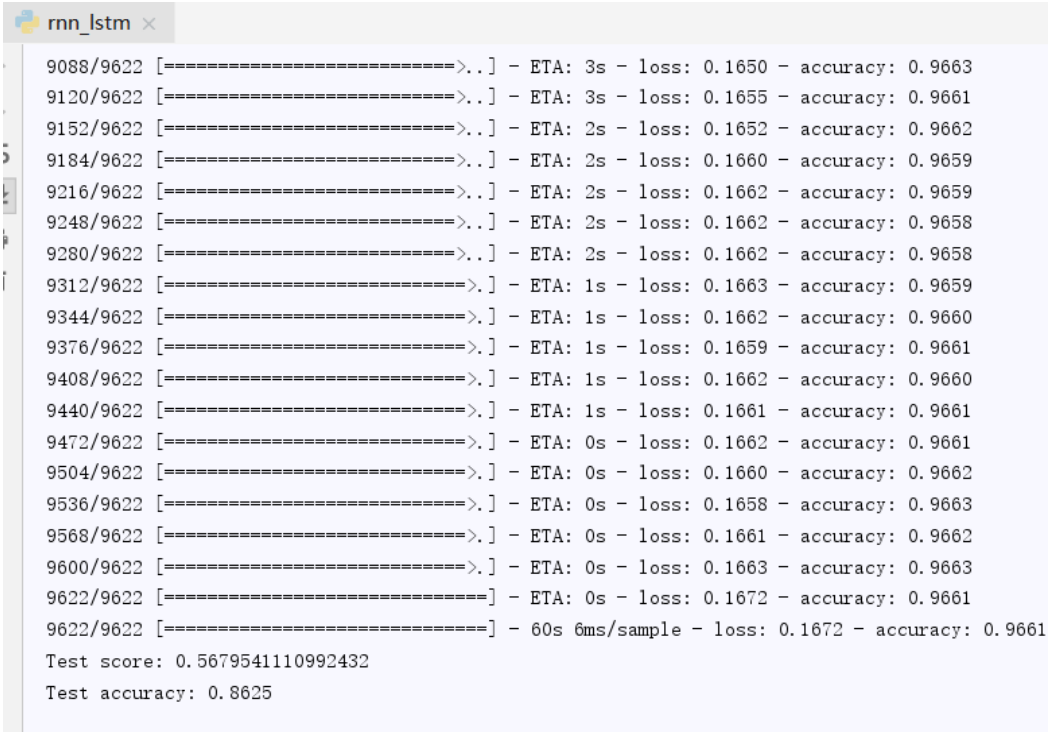

Figure 6: The accuracy of LSTM.

It is important to note that, during the data processing stage, stop words are removed from the input, yet the removal was restrained to be a rather small one. The reason for this is that, given the Chinese language has the characteristic lack of inflections in singular characters, the context of sentences are more dependent on the so-called stop words. Despite in other languages such words would be removed, in Chinese they affect the meaning of the sentence. Thus here, LSTM has the advantage of utilizing the relationships between words to evaluate the sentimental meaning. It is doubted, however, if LSTM will still outrun CNN in similar tasks with other languages.

Additionally, though LSTM has a higher test accuracy than CNN, it takes much longer for LSTM to be trained compared to that of CNN. We would suggest the efficiency and accuracy of the algorithms to be again pondered carefully upon application.

5. Conclusion

Overall, out of all algorithms, Extra Trees manages to perform the best, with Random Forest being a close runner-up. The characteristic decision tree structure might have allowed these two similar algorithms to perform well in this task, and the reason why they outrun their fellow Decision Tree is likely due to their ability to reduce overfitting via a touch of randomization and dropouts. Gaussian Bayes turns out to perform poorly, probably due to the contradicting desired input for the algorithm and the given datasets.

The machine learning algorithms are evaluated independent from the deep learning algorithms due to the latter's on hardware efficiency. Among the two, LSTM seems to perform better, likely due to its ability to grasp the connection between words sequentially.

Acknowledgments

This project cannot be completed without the support from my teachers. I offer my sincere appreciation for the learning opportunities provided by them. Throughout this project I found myself going through self-exploration while I venture in knowledge more sophisticated than any I had before, and my gratitute for this experience cannot be expressed enough. My heartfelt thanks to all of you who was there and helped me with my work.

ISSN 2616-5775 Vol. 5, Issue 8: 81-89, DOI: 10.25236/AJCIS.2022.050812

References

[1] Great Learning Team et al. "Multinomial Naive Bayes Explained." GreatLearning Blog: Free Resources What Matters to Shape Your Career! 7 Mar. 2022, https://www.mygreatlearning.com/blog/ multinomial-naive-bayes-explained/.
[2] Ameer, Iqra et al. "Multi-Label Emotion Classification on Code-Mixed Text: Data and Methods." IEEE Access, 2022, https://ieeexplore.ieee.org/document/9682721.
[3] Balodi, Tanesh. "Convolutional Neural Network with Python Code Explanation: Convolutional Layer: Max Pooling in CNN." Convolutional Neural Network with Python Code Explanation | Convolutional Layer | Max Pooling in CNN, 2020. https://www.analyticssteps.com/blogs/convolutional-neural-network-cnn-graphical-visualiza-tion- code-explanation.
[4] Bandhakavi, Anil, et al. "Lexicon Based Feature Extraction for Emotion Text Classification." 3 June 2017, https://doi.org/10.1016/j.patrec.2016.12.009. Accessed 13 July 2022.
[5] "Decision Tree Classifier." Decision Tree Classifier - an Overview | ScienceDirect Topics, 2021 https://www.sciencedirect.com/topics/computer-science/decision-tree-classifier.
[6] Elton. "Convolutional Neural Networks (Cnns) for NLP." Python Wife, Python Wife, 8 Dec. 2021, https:// pythonwife.com/ convolutional-neural-networks-cnns-for-nlp/.
[7] ML: Extra Tree Classifier for Feature Selection." GeeksforGeeks, 1 July 2020, https:// www.geeksforgeeks. org/ml-extra-tree-classifier-for-feature-selection/.
[8] Popuri, Sai K. "An Approximation Method for Fitted Random Forests." ArXivLabs, 5 July 2022, https:// arxiv.org/abs/2207.02184.
[9] Soni, Sanskar, et al. "TextConvoNet:A Convolutional Neural Network Based Architecture for Text Classification." ArXivLabs, 10 Mar. 2022, https://arxiv.org/abs/2203.05173.
[10] "Understanding Logistic Regression." GeeksforGeeks, 28 June 2022, https://www.geeksforgeeks. org/understanding-logistic-regression/.
[11] https://zhuanlan.zhihu.com/p/32085405.