Popular Songs Mood Recognition Based on Lyrics Sentiment Analysis

Xin Yi
University of Waterloo
Waterloo, Canada
x22yi@uwaterloo.ca

Ruiyin Jiang
University of Waterloo
Waterloo, Canada
r49jiang@uwaterloo.ca

Xiaomin Yan
University of Waterloo
Waterloo, Canada
x77yan@uwaterloo.ca

***Abstract*—This paper proposes an application-oriented project which focuses on popular songs mood recognition. Recently, social application use emotional tags to describe objects catering to the user's personal experience and mood. Music softwares are no exception based on the fact that most popular music aims to create an atmosphere through its melody and lyrics. In previous work, the identification of music emotion is primarily based on identifying audio signal features, but the text of the lyrics is valuable in terms of readability and operability, especially for popular songs. In this paper, according to the performance of different machine learning methods on lyrics mood recognition and prediction from this project, the application of lyrics sentimental analysis can be expected, such as generating user-specific mood playlists.**

***Index Terms*—Music mood recognition, sentiment analysis, machine learning, neural network,**

I. Introduction

The work of sentiment analysis is to identify and extract subjective information from given data through machine learning, then to perform quantitative or qualitative based on predetermined rules. It allows us to classify the sentiment of data, such as positive or negative, according to what it contains. In such a media era, digital data like music has a distinct relationship with the social life which have attracted the attention of researchers. By more directly combining the research on the emotional expressiveness of music with science, it will be possible to design more explanatory empirical results in the context of theoretical predictions [1].

Due to its explicit attributes, sentiment analysis of music mainly focuses on two aspects: audio signal and lyrics text. Studies [2] have shown that part of the semantic information of songs resides exclusively in the lyrics as much as tunes which shows the emotional expression ability of lyrics. For popular songs, most of which have lyrics in addition to the tune, the lyrics provide another layer of emotion, attitude, and narrative [3]. Accordingly, how to transit from unstructured text objects to insights arouses our interest. In this work, methods of mining lyrics features are explored in order to classify songs by mood like happy and sad.

MoodyLyrics is regarded as our dataset for training, Given that our dataset consists of most English songs and some Spanish songs, the attempt to build a bilingual corpus is insisted.

The following step is vector representation. Sparse vectors and dense vectors of fixed dimensions are both taken into consideration. Therefore, several feature vectorization methods like Word Embedding Bag of Words and TF-IDF are given a tryout. Then, a comparative study on classic machine learning algorithms with deep networks has experimented on the preprocessed data.

In addition to training from the overall lyrics of the songs, we also look forward to whether the method of extending the corpus by dividing the lyrics into individual sentences with labels can improve the classification and prediction. We reflect on this thought with the experimental results as an extended study.

The rest of this paper is arranged as follows: Section is a review of related work in the filed of music mood classification and sentiment analysis based on text. Section gives a description of the methodology and the workflow of our project. Then, Section introduced the lyrics dataset for actual operation. Section shows our main contribution, which presents the implementation and how different machine Learning tools and techniques are utilized and how they can be evaluated. Section displays the peformance on the testing data and how we can put our selected model into application which is generating music playlists. Finally, a conclusion with future outlooks is displayed in Section .

II. Related Work

Music mood classification has been studied by researchers since 1990s and it was mainly achieved by two methods: text-based and audio-based sentiment analysis approach. Lyrics sentiment analysis is regarded as an important topic in many studies of text-based music mood classification.

A previous work [4] implemented lyrics-based sentiment classification focusing on both knowledge-based and machine learning approaches. In the proposed knowledge-based approach, HowNet [5] is used to sentiment words detection and sentiment units location. Recent knowledge based approach text analysis on sentence level is also introduced in details in paper [6], for example, using the presence of sentiment lexical items like single word or n-grams to detect the emotional sentence. Or by exploiting association rule mining for a feature-based analysis. However, such approaches are quite

limited in both efficiency and accuracy, since they mainly rely on knowledge bases and the text analysis is not enough due to the reason that one sentence may contain various opinions or emotions.

In machine learning approach, like other natural language processing problem, it can be divided into supervised learning method, semi-supervised learning method and unsupervised learning method. We can consider the sentiment analysis problem to be a text classification problem. Analyzing the word frequency of lyrics or extracting words with emotional characteristics can make text classification get better results.

For supervised learning methods, the most common algorithms are naive Bayes and support vector machines. The navie Bayes classification is a probabilistic classification method. For a given item to be classified, the probability of its occurrence in different categories needs to be calculated, and the category with the highest probability is the one which the item belongs to. SVM is a linear classifier, originally a binary classification model, which divides high-dimensional data into different categories by hyperplane [7]. In hyperplane, the function can be represented as

$$W^T X + b = 0, \tag{1}$$

and the training process is to find the appropriate weight and bias through a large amount of data.

In addition to the two algorithms mentioned above, there are some neural networks based on supervised learning, such as CNN and RNN.

CNN has been widely used in computer vision, especially in the field of image classification. The core of this method is the application of the convolutional layer, which uses sliding Windows, also known as cores, to process data, which can also solve a lot of problems in natural language processing [8]. Taking the most common text classification task as an example, The function of convolutional layer is to detect features, while the function of pooling layer is to screen features and extract only key information, which can reduce the amount of training. For NLP tasks, the input is usually a sentence or document. If it is a sentence, each line represents a word, and each word can be represented by a vector. There are many ways to represent the vectors, such as Word2vec, one-hoe and so on. After determining the input word vectors and filters, convolution operation can be performed to obtain feature map. After pooling layer, softmax will be used to classify the feature vectors of sentences.

In recent years, CNN has also made many significant breakthroughs in the field of natural language processing, with excellent results in speech recognition and sentiment analysis. Researchers have also begun to conduct more in-depth studies on the application of CNN, such as the sentiment analysis of short texts, the use of Bag of Words and the application at the character level [9].

Another neural network used in NLP is recurrent neural network, abbreviated as RNN, in which a special network structure LSTM is more commonly used.

III. Proposed Method

In our proposed method, figure 1 shows how we train and select lyrics sentiment analysis model. Various steps in training are described as below:

- Training data preparation: Download the original lyrics data and convert them to the format that can be used for training;
- Preprocessing: Preprocess the lyrics data by techniques like text cleaning and stop words removal ;
- Feature extraction: Create vector representations for the song lyrics as their features;
- Training and selecting: Train and adjust different machine learning algorithms and neural networks to see and compare their performance.

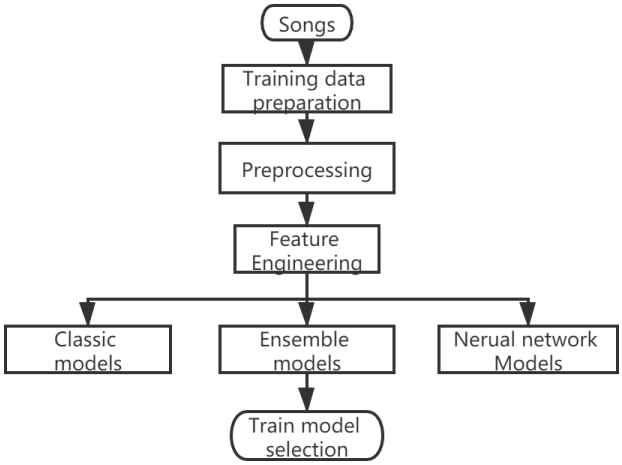

Fig. 1. Model selection workflow.

To generate the playlist, we process the songs for test using the similar steps in the training stage, and fit the processed testing data to the selected model which has the best performance in our experiments. Then the models will make predictions of sentiments and finally generate the playlists by their emotions. The workflow of how playlists are generated is shown in figure 2

IV. Dataset

In this section we will introduce the training dataset and testing dataset we used in our project. For the training process, we use MoodyLyrics [10], which is a manually generated song lyrics dataset. The reason why we choose this dataset for training is that it is bigger when comparing to other current public song lyrics datasets, and the labels are correct and accurate since they are annotated by experts and user tags. It contains 2595 different songs which are labeled by four different moods categories: happy, sad, angry and relaxed. In this dataset, only content words of lyrics and their valence and arousal are used to assign songs with their labels. The authors of paper [10] used Russell's Valence-Arousal model [11] as shown in

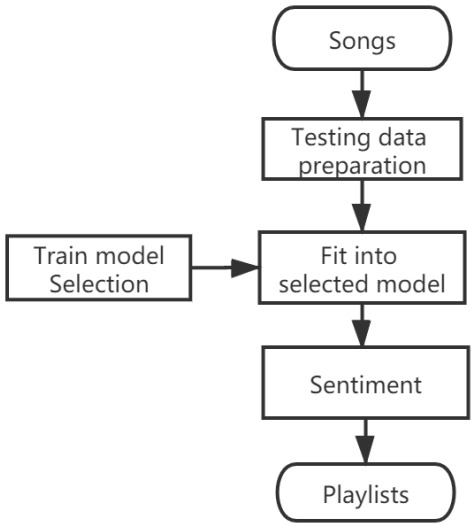

Fig. 2. Playlist generation workflow.

figure 3 with the four categories for the annotation process, where the valence describes the positive or negative intensity of an emotion and gives the unpleasant and pleasant degree, while the arousal represents how strongly or rhythmically the emotion is and gives activation and deactivation degrees.

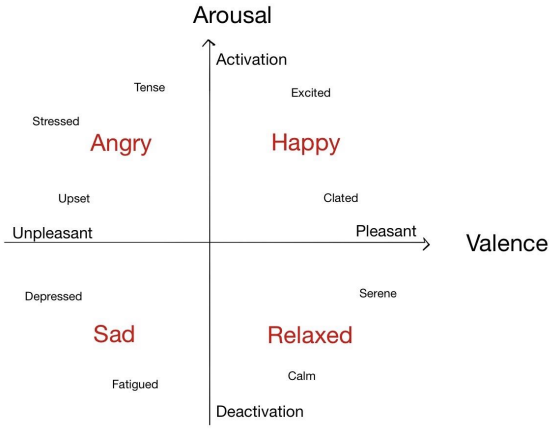

Fig. 3. Valence and Arousal planar model.

MoodyLyrics dataset gives song index, artist, song title and their emotion labels, the song lyrics are not given since they are copyrighted and restricted. To applied this dataset to our project, we downloaded song lyrics from LyricsWikia using python script. Regardless of those were failed to be downloaded, finally 2464 lyrics files in total are obtained successfully and are used to train in the machine learning models and neural networks in our proposed system. And for the testing process, we use NJU-MusicMood-v1.0 [12]. It contains over 300 songs of the four mood categories which are divided to 71 angry songs, 106 happy songs, 101 relaxed songs and 99 sad songs. Although they are not completely balanced, it is acceptable for us to do the evaluation for our classification task.

V. Implementation

A. Training data preparation

Firstly, we need to extract the lyric files according to the MoodyLyrics dataset. We use a python script to download the lyrics of the songs contained in MoodyLyrics CSV file by each song's artist and title from LyricsWiki, and save them to a lyrics dataset directory which contains four sub directories namely angry, happy, relaxed and sad. If failures happen during the downloading process, the songs that are failed to be downloaded will be recorded in a log file in convenience of dataset analysis. Then we convert the lyrics files and their labels to array forms, finally a DataFrame format dataset is generated and ready for next steps.

B. Preprocessing

Firstly we load the DataFrame format dataset created by former preparations to read the lyrics. Since the raw dataset is with no particular preprocessing steps nor feature engineering techniques, we need to preprocess the data before using them as inputs to our classification models.

In the song lyrics, there are symbols and punctuation which split one whole lyrics into sentence or phrases, however, such punctuation need to be removed. Then we need to perform word segmentation and stop word removal, and then connect the remaining list. There are also stop words, which are a set of commonly used words in a language, since they are less meaningful for our training, we remove such word in both English language and Spanish language, so that we can focus on more important words in the lyrics instead of the words that contains less information.

C. Features

1) Word2vec: In order to apply machine learning algorithms in natural language processing problem, we need to firstly build corresponding vector representation. Word2vec is one of a more sophisticated word embedding technique proposed by Mikolov et al. [13] [14] for word expression. The idea behind it is that words which have similar contexts tend to have similar meanings. Word2vec takes a large corpus as input and produces a vector space with each unique word in the corpus being assigned a corresponding vector in the space. It turns text into integers form that neural networks can process as inputs. It can be implemented in Keras by using embedding layer, which can be regarded as a lookup table, where the rows are vector representations of each word in the vocabulary. In Keras, the data preparation that reapresenting each word by a unique integer is performed using the Tokenizer API. The embedding layer is defined as the first hidden layer of the neural network and is initialized with random weights. And it takes the size of vocabulary in the text data, the size of the

TABLE I
VALIDATION RESULTS FOR CLASSICAL CASES

Classifier	Accuracy	Precision	Recall	Fitting Time
Naive Bayes	0.776	0.800	0.801	0.008s
Logistic Regression	0.923	0.925	0.925	0.843s
SVC	0.925	0.926	0.926	1.936s

TABLE II
VALIDATION RESULTS FOR ENSEMBLE CASES

Ensemble	Base	Structure	Accuracy	Fitting Time
Adaboost	LR	n = 300	0.899	20.03s
RandomForest	Tree	n = 150	0.799	3.77s
Stacking	LR+SVC	LR	0.925	12.39s
Stacking	LR+SVC	SVC	0.931	12.47s

vector space in which words will be embedded, the length of input sequences as arguments.

2) TF-IDF Vectorizer: The TF-IDF is the product of two statistics, term frequency and inverse document frequency. The TF-IDF value can evaluate the importance of a term to a specific document in a corpus. The importance of the word increases in proportion to the number of times it appears in this document, but also decreases in inverse proportion to the frequency of its appearance in the whole corpus. In practice, the TfidfVectorizer function is used to construct a bag of words model of TF-IDF. For a term *t* in a document *d*, the weight of term *t* in document *d* is given by:

$$W_{t,d} = TF_{t,d} log(N/DF_t) \tag{2}$$

- W_{td}: the TF-IDF weight of term *t* in document *d*.
- TF_{td}: the number of occurrences of term *t* in document *d*.
- DF_t: the number of documents in the corpus containing the term *t*.
- N: the total number of documents in the corpus.

D. Classification Models

1) Classical algorithms: Several classical classifiers are tested about their performance on the prepossessed data. Hyperparameter tuning is carried out through GridSearchCV and 10-times 10-folds cross validation.

- Logistic Regression: The hyperparameter in logistic regression is C. the smaller the C is, the greater the regularization intensity is.
- Support Vector Machines: The kernel function can increase the feature size of the model (low-dimensional to high-dimensional), so that SVM has better nonlinear fitting ability, thus the kernel choice is rather important. While C represents how much we care about classification errors If the value of C is too large, the classifier will make every effort to make fewer mistakes on the training data, but in fact this is impossible, so it causes overfitting. If the value of C is too small, the classifier will deviate the classification and cause poor results.
- Naive Bayes: We prefer the MultinomialNB classifier, because most of the sample features are multivariate discrete values according to tf-idf calculation.

When exploring different algorithms, the best 2 results of training and validation performance considering accuracy and running time come from Logistic Regression and SVC.

2) Ensemble models: Ensemble learning aims to combine several machine learning methods as base-level classifiers into one predictive model which is the meta-classifier in order to decrease variance, bias, or improve predictions.

- Stacking: Stacking is one of the framework which uses a specific learning algorithm to build a meta-classifier do the combination of basic classifiers [15]. Then, based on the results given by all basic-level classifiers, the induced meta-level classifier is used to accomplish the final prediction.All the base-level models are used to train on the entire set. Then the meta-model is trained on the outputs of the base-level models as features. The choice of the base classifier is very important, not only to ensure that each classifier can make an effective prediction for this data set, but also to ensure that the characteristics of the classifiers are not too similar. According to the training results from those classical classifiers, we apply Logistic Regression and SVC with the best hyperparameters to stack classifiers. In each run Logistic Regression classifier and SVC classifier are the base-learners and one of them is the meta-learner.

Other types of Ensembles are also taken into consideration like AdaBoostClassifier and RandomForestClassifier, whose goals are to combine multiple weak classifiers reasonably to make it a strong classifier.

- Boosting: The AdaBoost algorithm is implemented by adjusting the weight corresponding to each sample. At the beginning, the weight corresponding to each sample is the same, that is, where n is the number of samples, a weak classifier is trained under this sample distribution. For samples that are classified incorrectly, increase their corresponding weights; for samples that are classified correctly, reduce their weights, so that the incorrectly classified samples are highlighted. By introducing these weights, the error rate can be reduced further. After many iterations, these weak classifiers each of which with a certain weight boost to obtain the final desired strong classifier. In AdaBoostClassifier, the base estimator is the designated weak classifier, the algorithm specifically includes AMME and SAMME.R. The main difference between the two is that the second one uses probability as the weight introduced. N estimator is the number of iterations and learning rate is the weight reduction coefficient of each weak learner. In actual operation, we tune the hyperparameters separately. The final choice is: the base estimator is Logistic Regression with C equals

15, the algorithm is set to AMME with 300 iterations at a learning rate of 0.8.

- Random Forest: The hyperparameter n-estimators which is the number of base classifiers, has a monotonic effect on the accuracy of the random forest model. The larger the n-estimators, the better the effect of the model. But correspondingly, any model has a decision boundary. After n estimators reaches a certain level, the accuracy of the random forest often does not rise or begin to fluctuate. Moreover, the larger the n-estimators, the larger the amount of calculation and memory required, and the training time is also It will get longer and longer. For this parameter, we are eager to strike a balance between training difficulty and model effect. Based on GridSearchCV, 300 is chosen.

We use a ROC curve to explore the impact on accuracy of the best ensemble model, which is the stacking model using SVC as its final estimator. The AUC(Area Under the Curve) value measures the area under the ROC curve. The criteria for judging the quality of a classifier by AUC. When AUC reaches 1, it means this is a perfect classifier. However, in most prediction scenarios, there is absolutely impossible. AUC equals 0.5 indicates the model works like random binary prediction, which means it has no value for prediction. From the ROC curve, we can tell that the stacking model performs well on the four categories of data based on the the results of training and validation.

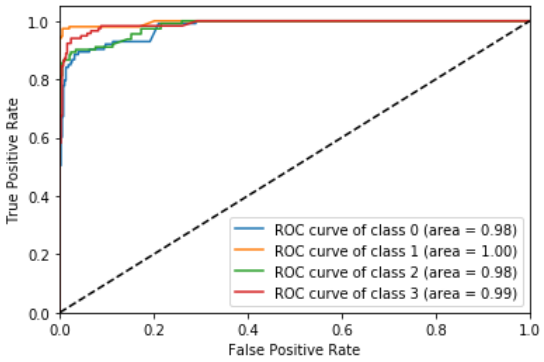

Fig. 4. ROC curves of the best ensemble classifier

3) Convolutional neural networks: Convolutional neural networks(CNN) is widely used in image classification, object detection and semantic segmentation tasks. It may also achieve good performance on natural language processing problems combining with word embedding techniques. The architecture of the CNN model we use is shown in 11, which consists of an embedding layer, an one-dimensional convolutional layer, a global average pooling layer, dropout layer after each operation layers and two fully-connected layers. We tried some complex CNN models but the result shows their performance is not ideal, thus a simple network structure is applied. The details of the network design is introduced below.

Firstly a matrix $S \in \mathbb{R}^{d \times n}$ is formed, where d is the length of each input data, and n is the representation of its n .

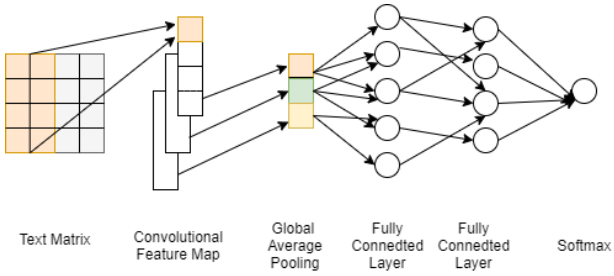

Fig. 5. CNN network structure.

The first layer is a lookup table $X \in \mathbb{R}^{d \times |V|}$, where the d-dimensional vector is from each word in the input text and V is the vocabulary, so the i^{th} word in V is represented by the i^{th} column of X.

Then the $X \in \mathbb{R}^{d \times |V|}$ matrix, which is the output of the embedding layer is put into the first convolutional layer. The number of filters is set to 16, and the kernel size is set to 2 with valid padding and relu activation. We use a window which is decided by kernel size to slide over the input matrix. We use $S_{[i:i+n]}$ to represent word vectors S_i to s_{i+n}, thus, by convolution using the given filter F, we obtained feature C_i representation:

$$c_i := \sum_{k,j} (S_{[i:i+n]})_{k,j} F_{k,j} \tag{3}$$

After the matrix representation passed through the convolutional layer, it is used as input to the pooling layer. In our proposed method, global average pooling is used to compute the mean value for each feature map. As introduced in paper [16] compared to other ordinary pooling, the goal of global average pooling is not to make the network invariant to small translations and it does not reduce the size of input. However, its goal is to partly or completed replace the fully connected layer. Since the huge number of parameters in fully connected layers makes it slower train because a large amount of computational power is required, also, it often cause over-fitting as well, even if regularization techniques like dropout can be applied to avoid this problem, it is still not very well controlled. Global average pooling is a simpler choice of convolution structure for establishing the relationship between the feature map and the category, and it does not require parameters so that it can avoid over-fitting in this layer. Also, it sums up spatial information and is more robust to input spatial changes. Taking these advantages into consideration, we use global average pooling layer along with fully connected layers in our network.

Dropout is also applied here to avoid over-fitting. The mechanism of dropout is that, for each hidden layer, each training sample, each iteration, we ignore a random fraction, p, of the layer outputs. We add dropout layer after Embedding layer, global average pooling layer and the fully-connected hidden layer, and we finally set the dropout rate as 0.1 after the process of parameter tuning.

Fully connected layer computes $\alpha(W * x + b)$, where $W \in \mathbb{R}^{m \times m}$ represents for the weight matrix and b is the bias, and α represents for the activation, here relu function is used. There are 32 units for the first fully connected layer and 4 units for the last fully connected layer. Finally, the outputs are fully connected to a softmax function, which returns the class $y_pred \in [0, 1, 2, 3]$with largest probability since there are four classes.

$$y_{pred} := \arg\max_j \frac{e^{x^T w_j + a_j}}{\sum_{k=1} K e^{x^T w_k + a_j}} \tag{4}$$

Where w_j is the weights vector and b_j is the bias of class j.

The network is trained for 200 epochs using adam optimizer and categorical crossentropy loss function, the batch size is set to 128. After the training, finally the accuracy on training set reaches to 98% and the accuracy on validation set reaches to 91%. The train loss and validation loss are respectively 0.04 and 0.28. The figure 6 and figure 7 respectively shows the accuracy and loss of this CNN model.

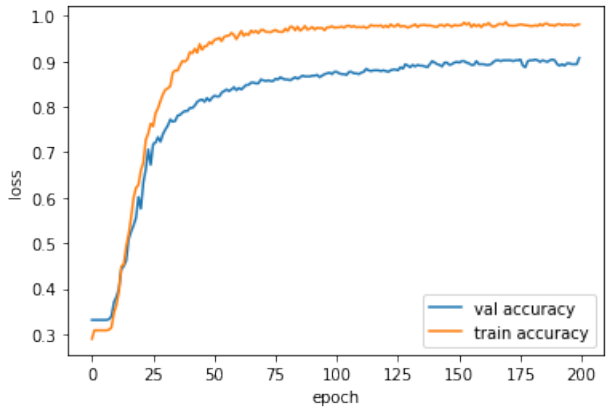

Fig. 6. Accuracy of the CNN model.

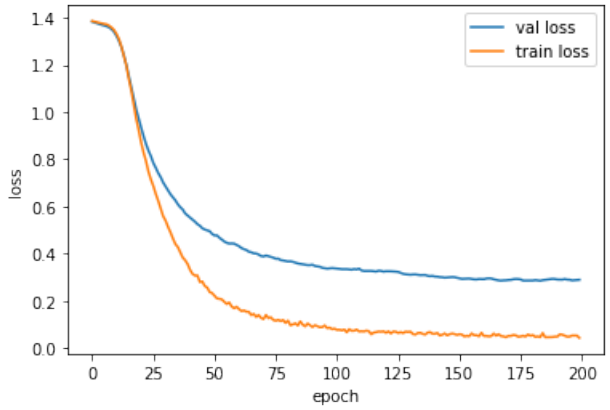

Fig. 7. Loss of the CNN model

E. lstm

LSTM is a special network structure belonging to recurrent neural network. Compared with the traditional neural network, RNN can guarantee the continuous existence of information through the cyclic operation of information, which is very important for the real-time prediction and analysis of sequence information. Take reading as an example. When people understand a text, they will think coherently based on the contextual information they have obtained, instead of just thinking about the meaning of the current reading fragments.

We can understand LSTM in terms of a simple RNN structure. There is a simple recurrent network shown as below: The activation in the figure represents the network structure

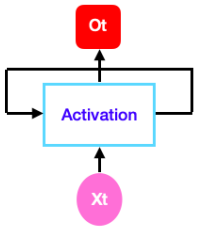

Fig. 8. RNN network structure.

which contain the activation function in RNN, the Xt means the input at time t and the Ot means the output at time t.

We can think of this simple structure as a recurrent network, continuously receiving input X and sending output O. The information looping through Activation ensures that the previous information is preserved in each step of the calculation. The recurrent network can also be expanded into a chain structure, which means that we can use the information already stored to process the current task accordingly. In theory, RNN can solve this problem of long dependence, that is, it can connect the previous information with the current task, and predict the task even if the effective information is far apart. However, in practice, it is found that the prediction task has a low probability of success.

The LSTM can solve this problem above. Compared with the standard RNN which contains only one network layer in each unit, there are four network layers inside the LSTM. In addition to the activation function tanh, there are also three sigmoid functions working as the gate, which play the role of letting the information selectively passing through. We can follow steps to understand the LSTM structure.

The first step is to determine the information to be discarded using the sigmoid unit.

$$d_t = \sigma(W_d * [O_{t-1}, x_t] + b_d) \tag{5}$$

Then we need to identify the useful new information that needs to be added.

$$ad_t = \sigma(W_{ad} * [O_{t-1}, x_t] + b_{ad}) \tag{6}$$

The information is then passed to the tanh unit as candidate information for the next cell.

$$\widetilde{C_t} = tanh(W_C * [O_{t-1}, x_t] + b_C) \tag{7}$$

Next, the old cell need to be updated.

$$C_t = f_t * C_{t-1} \tag{8}$$

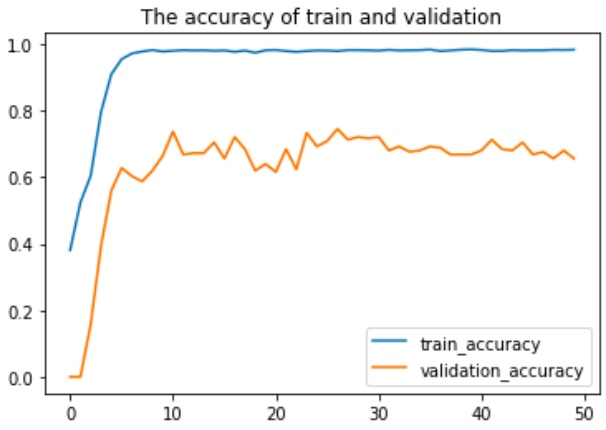

Fig. 9. The accuracy of RNN training and validation.

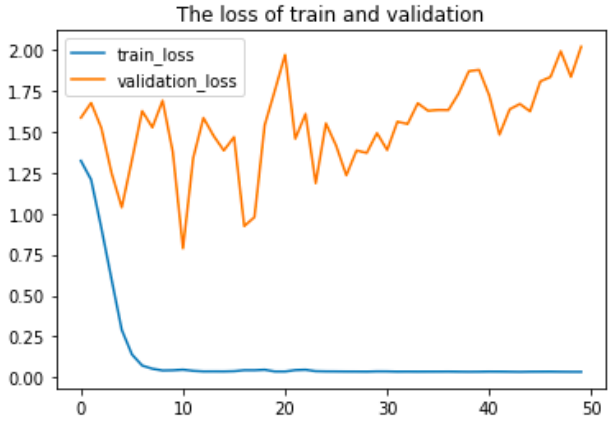

Fig. 10. The loss of RNN training and validation.

F. Document-level vs. Sentence-level

In fact, for lyrics, the next level of text unit is a paragraph or a sentence. Therefore, we also considered training with a set of short individual samples to see whether a better result can be reached. According to the characteristics of the lyrics, we split the source text into sentences. In the preprocessing part, the extra work is to remove only the punctuation that symbolizes the end of a sentence. Encountered short sentences separated by commas, each comma is replaced by a space to form a complete sentence.

How to choose a collection of sentences that is representative for each song is worth thinking about. Because of previous experience in word embedding and TF-IDF calculation, for the text documents after removing the stop words, RegexpTokenizer is used to match alphabetic characters since both Spanish words and English words appear. Then the bag of words dataframe is created is generated by frequency distributions encoder. In view of the average length of the lyrics text, we choose the first 5 high-frequency words to form a series of high-frequency word lists. Then, all sentences with the most frequent 5 words in each lyrics are combined with their label to build our new dataset MoodySentences as we called.

TABLE III
COMPARISON OF DIFFERENT MODELS

Model	Train/ Validation Accuracy	Time
Naive Bayes	0.784/ 0.776	0.329s
Logistic Regression	0.925 / 0.923	24.779s
SVC	0.935/ 0.926	105.832s
CNN+Word2vec	0.981 / 0.907	61.6s
LSTM+Word2vec	0.982 / 0.727	20.71s
RandomForest+TFIDF	0.798/ 0.799	107.755s
Adaboost+TFIDF	0.980 / 0.899	40.846s
Stacking1+TFIDF	0.982 / 0.925	26.372s
Stacking2+TFIDF	0.982 / 0.931	26.110s

In training, the validation accuracy of the classic model dropped from 0.9 to less than 0.7. CNN works better with the accuracy which is more than 0.8 when validating.

For exploring why there is a obvious gap, we look at another sentences collection called Emotions dataset for NLP from Kaggle. However, the classifiers at this time returned to good performance. After that, we compare the sample sentences in the two datasets, we find that a non-negligible amount of short sentences and too many sentence repetitions exist in the lyric texts of MoodyLyrics. Although we try to keep those components separated by commas, there are already many short sentences separated by periods in the original text. This is to say, the generated sentence dataset may to some extent destroy the distribution of features to the original lyrics documents. In addition, comparing with the more informative lyrics, each example is not so unique. Thus, text and sentence are not only the difference in the number of words contained, they also have completely different expression effects.

G. Results

Table III the train accuracy, validation accuracy and run time of various method, in term of running time, Naive Bayes takes the least computational time among these methods, but the accuracy is much lower than any other methods low for this classification task. From the table III we also can see, compare with classical machine learning algorithms, both ensemble models and neural networks achieve better training performance. In this work, our neural networks CNN and LSTM do not really achieve an ideal performance. There is a main problem in the implement of neural networks which is over-fitting, where the network achieves rather high training accuracy but low validation accuracy. We found deep neural networks may not perform very well on this dataset, since the dataset only contains thousands song lyrics, which is not enriched to train complex networks.

Very often it is the case that the classical algorithms are not optimal options. However, for those ensemble models created

with classical models, after tuning the parameters and fitting on training samples, the stacking and boosting models are characterized by the high prediction in all validation cases. To be honest, this does not meet our expectations because structures of these classifiers are simple. However, it is proved that sometimes larger ensembles may not work better than smaller ones [17]. Maybe the light structure of our ensembles also contributed to a certain extent. Afterwards, when we choose some other examples from the dataset MusicMood created by NJU [12], the results also justified. Ensemble models withstand the tests especially in the face of new lyrics data. In fact, the accuracy towards different classes vary from each other. From the performance report, it can be tell that the classification experience of happy songs and sad songs are the most outstanding.

In addition, compared to deep learning methods, whether for training or testing, boosting and stacking return results quickly. Based on all these advantages, ensembles are used to realize our ideas at the application level which is talked in next section.

VI. Generate Playlist

Next we move on to generate playlists according to one song's emotion. Our idea is to select a specific playlist and keep all titles and lyrics with the songs. Then, use lyrics to generate corresponding mood tags, and accordingly separate songs to each collection corresponding to each mood tag from the original playlist to form 4 new mood-related playlists.

After we compared the train accuracy and selected the model for our lyrics classification task, we use our best model for testing and generating playlists based on MusicMood created by NJUb12. The confusion matrix is shown as below. Through this matrix, performance of the classifier towards different categories can be easily seen. It is also possible to observe the relations between the results from any 2 different categories given by the model.

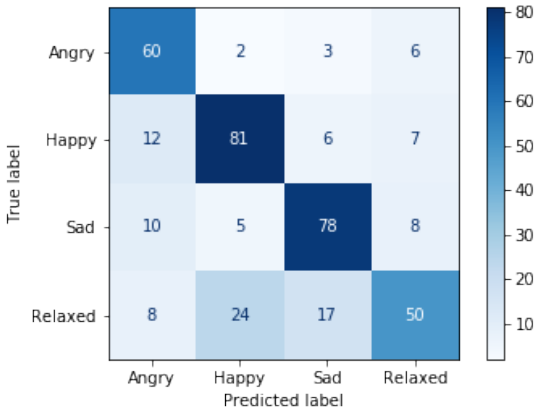

Fig. 11. Confusion matrix of the best testing results

From the figure 11 we can see, basically the songs are correctly classified to four moods, especially for the categories 'Angry', 'Happy' and 'Sad'. However, only half of 'Relaxed' songs are correctly assigned to the corresponding label, over 40% of 'Relaxed' songs are classified as either 'Happy' or 'Sad', which indicates the fact that our model work well on most songs but does not perform very well on relaxed songs.

VII. Conclusions and Future Works

Our project on the application of classic machine learning algorithms with calculating weight by TFIDF in ensemble methods, i.e. Boosting and Stacking, shows some thought-provoking findings. These ensembles bring different classifiers specialized in predicting some areas of the dataset [15]. For some specific problems, the classic method may be rather effective.

For future works, we can examine more playlists from trendy music players like Spotify, they are more time-sensitive and have catered to user choices to a certain extent, which helps us adjust the model to improve its availability.

From the perspective of model structure, we can explore more to optimize the neural network layers and the hyper-parameters. For example, to adjust our CNN model, we can try using multiple kernel size to obtain more detailed features and try other methods to to reduce over-fitting. To improve our model performances, we can also focus more on feature engineering. In this work, we generated word embedding vectors to represent words in song lyrics. However, there are more features can be taken into consideration such as echoism, which is a measurement of repeated words or phrase, rhymes, percentage of adjectives, subjectivity degree and so on. Such features may also be helpful for the getting better classification results.

References

[1] Nayak, S., Wheeler, B. L., Shiflett, S. C., & Agostinelli, S. (2000). Effect of music therapy on mood and social interaction among individuals with acute traumatic brain injury and stroke. Rehabilitation Psychology, 45(3), 274.

[2] Besson, M., Faita, F., Peretz, I., Bonnel, A. M., & Requin, J. (1998). Singing in the brain: Independence of lyrics and tunes. Psychological Science, 9(6), 494-498.

[3] Pettijohn, T. F., & Sacco Jr, D. F. (2009). The language of lyrics: An analysis of popular Billboard songs across conditions of social and economic threat. Journal of Language and Social Psychology, 28(3), 297-311.

[4] Xia, Y., Wang, L., & Wong, K. F. (2008). Sentiment vector space model for lyric-based song sentiment classification. International Journal of Computer Processing Of Languages, 21(04), 309-330.

[5] Zhendong, D., & Qiang, D. (2006). Hownet And The Computation Of Meaning (With Cd-rom). World Scientific.

[6] Cambria, E., Schuller, B., Liu, B., Wang, H., & Havasi, C. (2013). Knowledge-based approaches to concept-level senti 3111–3119.

[7] Liu, B. Sentiment Analysis and Opinion Mining. Synthesis Lectures on Human Language Technologies. 2012. pp. 1-167. ment analysis. IEEE intelligent systems, 28(2), 12-14.

[8] Liao, S., Wang, J., Yu, R., Sato, K., & Cheng, Z. CNN for situations understanding based on sentiment analysis of twitter data. Procedia Computer Science. 2017. 111, pp. 376–381.

[9] Johnson, R., & Zhang, T. .Effective Use of Word Order for Text Categorization with Convolutional Neural Networks.

[10] Çano, E., & Morisio, M. (2017, March). Moodylyrics: A sentiment annotated lyrics dataset. In Proceedings of the 2017 International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence (pp. 118-124).

[11] Russell, J. A. (1980). A circumplex model of affect. Journal of personality and social psychology, 39(6), 1161.

[12] Xue, H., Xue, L., & Su, F. (2015, January). Multimodal music mood classification by fusion of audio and lyrics. In International Conference on Multimedia Modeling (pp. 26-37). Springer, Cham.
[13] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
[14] Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems. 2013. pp. 3111–119.
[15] Graczyk, M., Lasota, T., Trawiński, B., Trawiński, K. (2010, March). Comparison of bagging, boosting and stacking ensembles applied to real estate appraisal. In Asian conference on intelligent information and database systems (pp. 340-350). Springer, Berlin, Heidelberg.
[16] Lin, M., Chen, Q., & Yan, S. (2013). Network in network. arXiv preprint arXiv:1312.4400.
[17] Zhou, Z. H., Wu, J., Tang, W. (2002). Ensembling neural networks: many could be better than all. Artificial intelligence, 137(1-2), 239-263.