Song Mood Detection - Mid Check Point Report

Yi Chen (yc2455), Yaxuan Huang (yh798), Guangwei Jiang (gj94),
Anqi Dong (ad829), Zhilin Li (zl635)

Abstract

This project analyzes song lyrics to determine the mood of a song, known as music emotion detection. We believe that music mood detection will be needed for all music platforms. In development, to obtain accurate results, we processed the data by reading, analyzing, and cleaning it. We also improved the model and developed a dataset better suited to its requirements.

Background

The project is to analyze song lyrics to determine the mood, aiming to develop methods for accurately attributing music emotions. The insights gained from the project can improve existing recommendation systems, enhance user experience, answer questions related to accurate mood classification, predict new song moods, and identify important features for mood classification.

Our team explored related research and datasets, such as FMA, Moodlyrics, and muSe. We also referenced the following articles: Mood Classification using listening data[1], Music Mood and human emotion recognition based on physiological signals: a systematic review[2], and Mood Classification with Lyrics and ConvNets[3]. These articles explain how music mood classification systems work technically, including the taxonomy and identification of music moods, as well as the use of machine learning.

Dataset

Our dataset is MoodyLyrics, a dataset retrieved from the paper MoodyLyrics: A Sentiment Annotated Lyrics Dataset[4]. MoodyLyrics is a widely used dataset that serves as a benchmark for music emotion recognition models. It was constructed using song lyrics from MillionSongSubset. Based on Russell's model, the songs were classified into one of four quadrants based on their normalized Valence and Arousal values. The resulting dataset contains 2,595 songs from various genres, with 4 emotion categories, including Happy, Angry, Sad, and Relaxed.

However, due to the copyright regulations, all song lyrics are removed. So we used the LyricsGenius, a Python library that allows developers to retrieve lyrics from the Genius.com database to collect the lyrics. It uses the Genius API to search for and download song lyrics by artist and title. After that, our dataset has four columns: Title, Artist, Mood, and Lyrics.

Milestones/Analysis

1. Data Cleaning

The process of data cleaning includes deleting duplicate data, solving the data imbalance issue, and preparing for further modeling process. We first check if there is duplicated data and delete 86 duplicated rows. Then we visualized the distribution of four Mood categories and found a slight data imbalance issue. We then used an oversampling method to balance the data.

In addition, we removed all the null values of the lyrics since there are only 75 of them. Next, we cleaned the lyrics including removing punctuations, Arabic numbers, newline characters, song titles, artist names, etc. Also, we removed all the stopwords and calculated the Percentage change in the number of words after stopword pruning. We found that although there is a minimum impact on lyrics, which still require many words for analysis, only a few songs reduced their size by more than 50%. While this may pose some challenges for these particular songs, it is a small percentage of the overall dataset and does not negate the effectiveness of using stopword deletion as a pre-processing technique for our model.

Last, we encoded Count_vector and TF-IDF according to the cleaned lyrics for further use in classification, especially for KNN and K-means clustering.

2. Feature Engineering

We first extracted some interesting features for further analysis. We processed the data using a bag of words analysis and converted the data using word embedding. To further extract other features, we moved on to sentiment analysis using TextBlob to extract polarity, lexical diversity, and average word length. We then used existing classifiers, here using k-means clustering to explore the emotion labels.

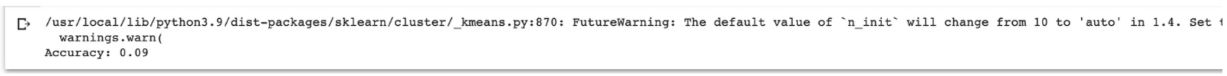
3. Model Training

We trained three different models: KNeighborsClassifier, LSTM, and Supervised K-Means. Initially, we used lyrics to train an LSTM and a Supervised K-Means model, but we soon discovered that the results were quite subpar.
Therefore, we chose to turn the lyrics into vectors using the "en_core_web_lg" pre-trained model. Then we trained with the KNeighborsClassifier model, which gave us good results with an accuracy of 67%. We believe that additional data cleansing may be necessary. Our model might be more accurate if we had better data.

Initial results

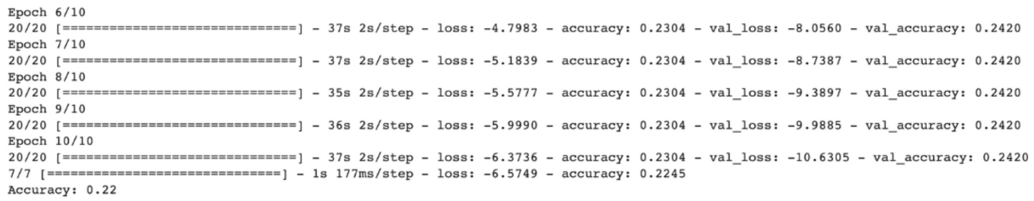
Here’s the accuracy of our three models: KNeighborsClassifier(0.67), LSTM(0.22), and Supervised K-Means(0.09).

- Supervised K-Means



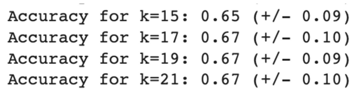
```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set t
  warnings.warn(
Accuracy: 0.09
```

- LSTM



```
Epoch 6/10
20/20 [==============================] - 37s 2s/step - loss: -4.7983 - accuracy: 0.2304 - val_loss: -8.0560 - val_accuracy: 0.2420
Epoch 7/10
20/20 [==============================] - 37s 2s/step - loss: -5.1839 - accuracy: 0.2304 - val_loss: -8.7387 - val_accuracy: 0.2420
Epoch 8/10
20/20 [==============================] - 35s 2s/step - loss: -5.5777 - accuracy: 0.2304 - val_loss: -9.3897 - val_accuracy: 0.2420
Epoch 9/10
20/20 [==============================] - 36s 2s/step - loss: -5.9990 - accuracy: 0.2304 - val_loss: -9.9885 - val_accuracy: 0.2420
Epoch 10/10
20/20 [==============================] - 37s 2s/step - loss: -6.3736 - accuracy: 0.2304 - val_loss: -10.6305 - val_accuracy: 0.2420
7/7 [==============================] - 1s 177ms/step - loss: -6.5749 - accuracy: 0.2245
Accuracy: 0.22
```

- KNeighborsClassifier



```
Accuracy for k=15: 0.65 (+/- 0.09)
Accuracy for k=17: 0.67 (+/- 0.10)
Accuracy for k=19: 0.67 (+/- 0.09)
Accuracy for k=21: 0.67 (+/- 0.10)
```

Next steps/Discussion

We have confirmed that converting lyrics to vectors using the pre-trained 'en_core_web_lg' model can produce better results. However, we have encountered some issues when using these vectors to train with the LSTM and Supervised K-Means models due to conversion problems. We hope to further address this issue or establish a new model to solve this problem. We will also try to classify the sentiment of lyrics using SVM, Gradient Boost, and Artificial Neural Network, compare and analyze the processing results of lyrics under various models, and visualize the analysis results effectively.
Based on other milestones we planned to implement, we will also add cross-validation approaches like k-fold cross-validation. For feature extraction, we will perform additional feature extraction methods to enhance and identify the most important features while removing noises. We will also look at other evaluation metrics such as the confusion matrix to get a better understanding of our models' performance.
New Milestones

- Model finetuning and performance evaluation - 0505
- Testing and evaluation - 0509
- Final writeup - 0515

Reference

[1]Korzeniowski, Filip, et al. "Mood classification using listening data." arXiv preprint arXiv:2010.11512 (2020).

[2]Chaturvedi, Vybhav, et al. "Music mood and human emotion recognition based on physiological signals: a systematic review." Multimedia Systems 28.1 (2022): 21-44.

[3]Akella, Revanth, and Teng-Sheng Moh. "Mood classification with lyrics and ConvNets." 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA). IEEE, 2019.

[4]Çano, Erion, and Maurizio Morisio. "Moodylyrics: A sentiment annotated lyrics dataset." Proceedings of the 2017 international conference on intelligent systems, metaheuristics & swarm intelligence. 2017.