RECOMMENDING SONGS FOR THE RIGHT OCCASION USING SUPPORT VECTOR MACHINES AND FLUTTER

Ruoxuan Dong[1], Austin Amakye Ansah[2], Shuyu Wang[3]

[1]Shanghai Soong Ching Ling School, No.2 Yehui Road,
Qingpu District, Shanghai, China
[2]The University of Texas at Arlington,
701 S Nedderman Dr, Arlington, TX 76019
[3]Computer Science Department, California State Polytechnic University,
Pomona, CA 91768

ABSTRACT

Most people listen to music to regulate mood, achieve self-awareness, and for socialization [9]. 95.6 percent of Americans aged 13 and older listen to some form of audio in their lives. According to a study, people stop finding new music by age 30. Our solution, Vocodex, provides a way for people to find new music related to their mood through emotional state description. Our solution is built with Flutter, which provides the tools needed to build a cross-platform app for Android, Ios, and the web [10]. Our app classifies emotions of up to four different classes and uses that information to fetch songs from a database. Songs are then played with our in-app music player. The model trained to analyze text achieved up to 94% accuracy when retrained on a heavier dataset.

KEYWORDS

Flutter, AI , Music, SVM

1. INTRODUCTION

Most people listen to music to regulate mood, achieve self-awareness, and for socialization [5]. According to Adweek, it is estimated that 95.6 percent of Americans aged 13 or older listen to some form of audio in their daily lives [6]. This figure amounts to 270 million Americans daily [6]. Twenty-seven percent of audio listeners use smartphones according to a global study conducted in 2019 [7]. According to another study, people stop finding new music by age 30 [8]. If people found music based on their emotional state, there is a possibility that they would find new music they never thought of looking for.

We propose a mobile application that asks users to describe their emotional state for new music recommendations. Our solution is cross-platform, for Ios, Android, and the web, and features a basic database containing various music options for up to 4 different emotional states. Some of the songs are popular and others are not as well-known, but they conform to their tendency to be classified among one emotional class. Our solution uses natural language processing and machine learning methods to analyze user input and provide a single recommendation [11]. Compared to other solutions we looked at, our app is the most user-intuitive and our machine learning model is simple to train in comparison.

David C. Wyld et al. (Eds): SE, SAIM, SIPM, CoNeCo, ICITE, ACSIT, CMIT, FCST, SNLP – 2024
pp. 135-145, 2024. CS & IT - CSCP 2024 DOI: 10.5121/csit.2024.140813

For the first experiment, we ran the model through a test and validation dataset to evaluate its accuracy. The goal of this was to see if there ways to improve the model performance. The test dataset contains 3,142 different samples. The training set contains 3,613 samples, and the validation set contains 346 samples. The confusion matrix for the first accuracy run yielded the following:

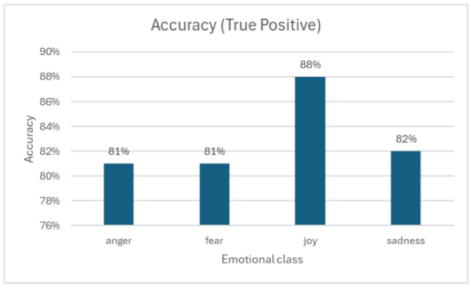

Figure 1. Accuracy 1

The second experiment yielded better results:

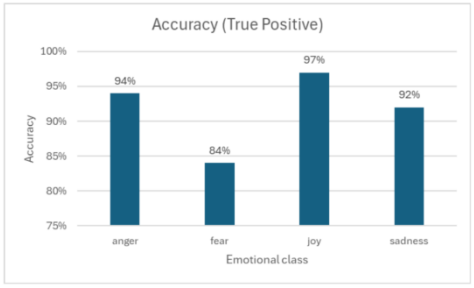

Figure 2. Accuracy 2

Nair, et al employed CNNs and LSTMs in a chatbot app that uses the Spotify API to generate a playlist in correlation to the user's responses to the bot's questions[2]. Their solution uses a CNN to learn features from the text in the dataset, and LSTMs to categorize the data into positive, neutral , or negative classes. When the user communicates with the bot, it uses all of the information from the previous messages to create a new spotify playlist for the user. Similarly to our approach, they employ a Support Vector Machine (SVM) to create a text-based emotional classification model.

The emotional recognition system from Madhavi et al, uses openCV to classify faces into one of 6 emotional classes and provide the user with a song playlist [3]. This solution employs CNN's for emotional classification from images and loads a predetermined list of songs for the user to choose from. Faces are loaded from a single image and analyzed without real-time processing.

Farkash and Petra employed LSTMs and NLTK to build a chatbot for personalized music recommendation [4]. Their system uses python to run the model and Flask for the website. Their solution uses keras to build an efficient deep neural network that can classify text data. When the user provides input that elicits a certain mood to the bot, a song is recommended as a link.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. The songID

Mixing python and dart objects lead to some errors while handling data in the app. One of these issues was that the songID passed is not equal to the real string value of the songID. This could have been because of decoding issues where invisible characters changed the actual value of the data.

2.2. Crashes

Throughout the app, there were many crashes initially due to popping out of a page when it is no longer in the tree. One such example is the registration/login page. After logging in, the registration page would no longer be in the widget tree, and so trying to pop out of the page to get to the homepage would cause an error. This error persisted in test builds.

2.3. Host the Python Server

Finding a good place to host the python server for free was an issue. With Vercel, there were build errors because the model size was too large. Many of the server hosting providers were also quite expensive, so for an app like this, it was best to attempt to find a free service that would at least keep the python server published indefinitely.

3. SOLUTION

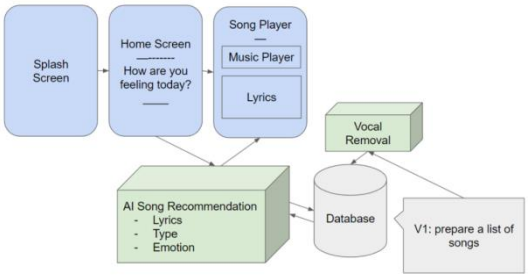

Figure 3. Overview of the solution

Vocodex uses flutter to handle the UI frontend, and a flask server built with python to handle sentiment analysis using a Support Vector Machine classifier (SVM) for a lightweight and transportable text classification model [12]. The app's most prominent feature is a music player that uses the audioplayers flutter library to play high quality sound. The song that is played is determined by the users emotional prompt. This goes through a Support Vector Machine classifier to predict the user's mood and select the right song from the database. The audio playback is handled through flutter streams that support most multimedia functions like playing, pausing, etc. The library's ability to play audio files from web urls was leveraged to play music without having to store any extra files on the user's device.

The music player features a square container that represents the song thumbnail. If the thumbnail does not exist, then a default blank image is used. The music player also has a lyrics view to fetch song lyrics from the database for each song.

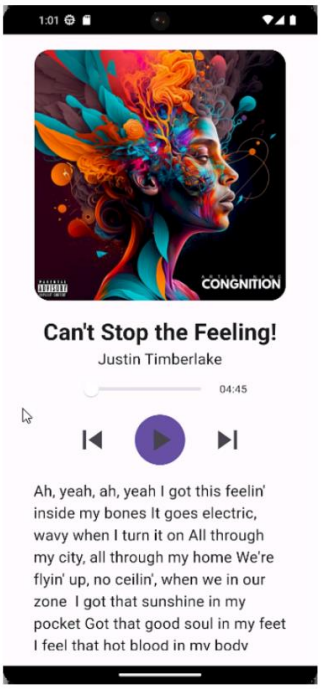

Figure 4. Music player interface with lyrics

Figure 5. Music Player class architecture

The above piece of code described the class architecture for the music player options. Four streams are created to handle real time music player events such as tracking the timestamp of the song and the music player state. Next, an Audio Player object is created using a reference from the previous page responsible for gathering user emotional data. Using text interpolation, the timestamps for the song are updated in real-time. Lastly, the dispose method cleans up resources in the widget when the page is not needed anymore.

Retrieving user input

This component is used to collect a user response for processing with the SVM. The page is quite simple and features a rectangular textbox which sends data to an external server running a python program that processes text and classifies its mood. The server runs a flask instance to process the data, and the data is used ephemerally to recommend the right songs.

Figure 6. Python code for nlp server

The above code is a python program for a flask server [13]. It describes three routes: One for the home page, or root endpoint, one to get the mood of the user, and another to get information from firebase for a random song based on the user's mood. The underlying code for fetching song information could also be leveraged to create song playlists, but it was decided that only one song per user input would be loaded.

User experience

The last component required is the profile page for the user. The idea was to provide a space where the user could see the previous songs they listened to, but it was later just delegated to be a theme switcher and a page where the user could log out. There is also the option to select a profile picture for account identification, but to also provide a better user experience.

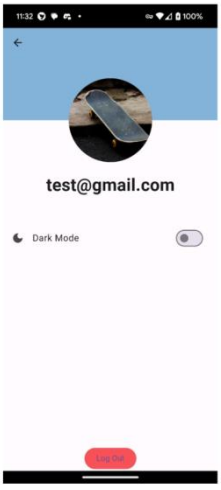

Figure 7. Screenshot of profile page

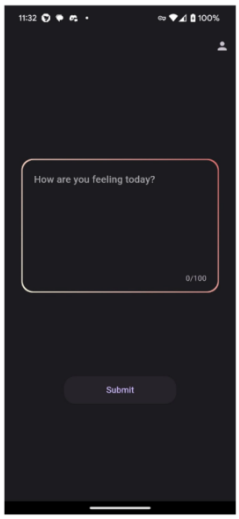

Figure 8. Screenshot of chatbot

```
FilePickerResult? result =
    await FilePicker.platform.pickFiles(
  type: FileType.custom,
  allowedExtensions: ['.jpg', 'png', '.jpeg'],
);

if (result != null) {
  File file = File(result.files.first.path!);
  Uint8List bytes = file.readAsBytesSync();
  String fileName =
      FirebaseAuth.instance.currentUser!.email! +
          '.jpg';
  await FirebaseStorage.instance
      .ref('profiles/$fileName')
      .putData(bytes);

  String downloadURL = await FirebaseStorage.instance
      .ref('profiles/$fileName')
      .getDownloadURL();

  await FirebaseAuth.instance.currentUser!
      .updatePhotoURL(downloadURL);

  setState(() {});
}
```

Figure 9. Code of a profile picture switcher

The above code details a profile picture switcher. The component opens a file picker page where the user can select an appropriate image. The result of the file pick is assessed, and if an image was selected, the image data is converted to raw byte data and uploaded to firebase at a specific location corresponding to the currently logged in user [15].

Currently, the only app styles are the light and dark themes. Users can switch between the two using the profile page and these settings are persisted to the app's storage.

4. EXPERIMENT

4.1. Experiment 1

In the initial phase of our research, we aim to scrutinize the predictive accuracy of the support vector machine (SVM) model employed within Vocodex for natural language processing tasks. This critical component is tasked with discerning the emotional tone conveyed by user input and subsequently retrieving a corresponding musical piece from our Firebase repository. Our evaluation will be grounded on a dataset comprising 3,142 distinct samples for testing, 3,613 for training, and 346 for validation.

The model's performance will be gauged through a meticulous process involving retraining the SVM to ensure updated learning and the generation of a comprehensive classification report. Leveraging the robust capabilities of the sklearn library, our methodical approach entails preprocessing the input text to excise all extraneous tokens. Post-cleanup, the data undergoes vectorization, transforming it into a format amenable to machine learning algorithms, and is then introduced to a Linear Support Vector Machine for training.

Upon fitting the SVM with the training data, we proceed to instantiate the classifier and vectorizer objects essential for predictions. The final stage involves deploying the test data through the model, with the outcomes meticulously compiled into a classification report. This document will reveal the model's proficiency in accurately classifying the moods, thereby underpinning the efficacy of our music recommendation system.

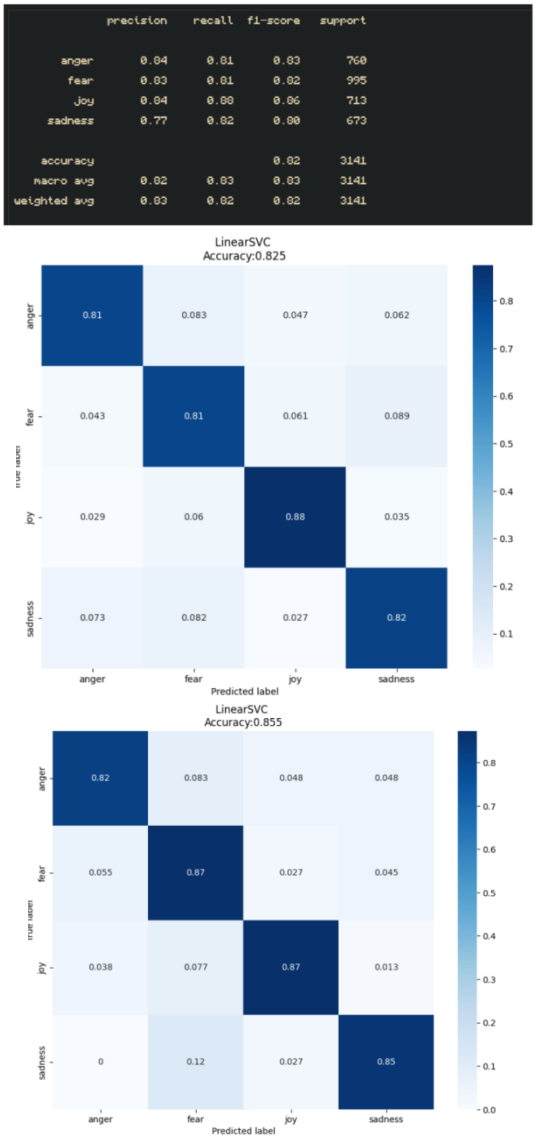

Figure 10. Figure of experiment 1

Based on these results, the AI must be quite reliable. The mean and median guess is 0% incorrectness. The AI was spot on with every single guess, meaning that it is quite reliable. The

reason for such a low percentage of incorrect guesses is because the AI has been provided hundreds of reference photos, meaning that it can correctly sort out each photo it receives. However, because of the small initial test size, the AI is not always perfect. 10 photos is quite little, meaning that the AI will still have room for error. Hence, I have decided to incorporate a manual input for the alcoholic beverage to assure that the accurate amount of alcohol is being entered every time. Also, the AI is unable to guess the volume of a drink, so I have added a function to manually input the volume.

4.2. Experiment 2

Our meticulous evaluation of the model's accuracy unfolds through the lens of a confusion matrix—a compelling visual representation that captures the success rate of emotion prediction across various sentences. In this matrix, each correctly predicted case resonates as a point on the diagonal, mirroring a harmonious intersection of prediction and reality. The intensity of the dark blue hues in our matrix symbolizes robust prediction capabilities, particularly for discerning a sentence's underlying emotion. Contrasting these triumphs, the lighter shades signal areas where the model's foresight falters.

The empirical evidence gathered thus far paints an encouraging picture: our model demonstrates an average accuracy of 82% across all emotion categories. Delving deeper, we uncover that the sentiments of anger and fear are pinpointed with an 81% precision. Joy, the emotion with the highest prediction frequency, soars to an accuracy of 88%, while sadness is identified correctly 82% of the time.

Driven by a relentless pursuit of excellence, our next endeavor seeks to elevate the model's predictive prowess. We are priming our system with a fresh dataset—2,000 samples reserved for testing, a robust training set of 16,000 samples, and 2,000 samples dedicated to validation. Our ambition is steadfast: to refine the accuracy of emotion recognition and set a new benchmark in the realm of sentiment analysis.

The experiment will repeat the same training, testing, and validation procedures as in experiment 1, but with a different set of training, testing, and validation datasets. The expectation is that with more training data, the model will be more accurate at predicting emotions over a range of sample data. The dataset was stripped of the “love” and “surprise” samples since the first dataset did not have those.

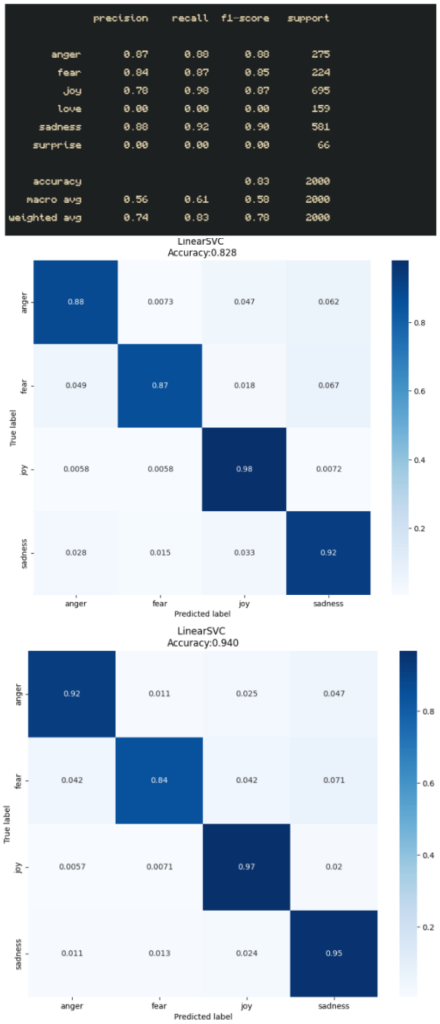

Figure 11. Figure of experiment 2

The model had an average accuracy of 94% when run on the validation dataset.

The same dataset classes as the first model were used in the second experiment. The only difference in the two experiments data-wise is that two classes, surprise and love were stripped.

```
val_data = val_data[val_data['emotion'] != 'love']
val_data = val_data[val_data['emotion'] != 'surprise']
```

Figure 12. Screenshot of two models

Anger had an accuracy score of 94% True positive, fear had 84%. Joy had the highest accuracy of true positives with 97%. Sadness was second highest, with 92%.

5. RELATED WORK

Convolutional Neural Network method to classify emotions over 6 different classes. The difference between their solution and ours is that they built a python program which uses OpenCV to process faces and provide a playlist [3]. Our solution uses a text classification approach and is built into an app that provides single songs.

Lastly, we looked at View of An AI Chatbot for Personalized Music Recommendations Based on User Emotions [4]. Their solution employs LSTMs and the NLTK python library to develop a Flask web application to get song recommendations. Our solution uses SVMs and NLTK with the flutter library for song recommendation. Their solution uses a chatbot to converse with the user before fetching a song, whereas our solution asks the user about their emotional state before recommending a song.

6. CONCLUSIONS

One feature that was planned for this project was the ability to see a playlist of songs previously listened to and move forward and backwards between them. If the project could have gone in any other direction, it would be worth considering streaming the music directly from youtube instead of storing it on a server. Another feature that could make the app a bit more explorative is a random song selection feature. This would select a random song from the database regardless of the input of the user.

The app accomplishes its primary purpose of providing new music to users based on their emotional input and the custom nlp server functions suitably with minimal downtime per hour [14]. Adding playlist functionality would give users more song options to choose from without the need to send requests to the nlp server so frequently.

REFERENCES

[1] Madhavi, MrsKommineni, et al. "Music Recommendation System Based On Emotion Recognition." Journal of Algebraic Statistics 13.3 (2022): 2105-2112.

[2] Nair, Amrita, et al. "Emotion based music playlist recommendation system using interactive chatbot." 2021 6th international conference on communication and electronics systems (ICCES). IEEE, 2021.

[3] Xie, Guobo, and Wen Lu. "Image edge detection based on opencv." International Journal of Electronics and Electrical Engineering 1.2 (2013): 104-106.

[4] Farkash, Rula M. Ali, and Tengku ZatulHidayah Tengku Petra. "An AI Chatbot for Personalized Music Recommendations Based on User Emotions." Journal of Computing Research and Innovation 9.1 (2024): 197-213.

[5] Schäfer, Thomas, et al. "The psychological functions of music listening." Frontiers in psychology 4 (2013): 54458.

[6] Forsyth, Michael. Buildings for music: the architect, the musician, the listener from the seventeenth century to the present day. CUP Archive, 1985.

[7] Kaplan-Neeman, Ricky, Chava Muchnik, and Noam Amir. "Listening to music with personal listening devices: monitoring the noise dose using a smartphone application." International journal of audiology 56.6 (2017): 400-407.

[8] Bonneville-Roussy, Arielle, et al. "Music through the ages: Trends in musical engagement and preferences from adolescence through middle adulthood." Journal of personality and social psychology 105.4 (2013): 703.

[9] Saarikallio, Suvi H. "Music in mood regulation: Initial scale development." Musicae scientiae 12.2 (2008): 291-309.

[10] De Andrade, Paulo RM, et al. "Cross platform app: a comparative study." arXiv preprint arXiv:1503.03511 (2015).
[11] Chowdhary, KR1442, and K. R. Chowdhary. "Natural language processing." Fundamentals of artificial intelligence (2020): 603-649.
[12] Suthaharan, Shan, and Shan Suthaharan. "Support vector machine." Machine learning models and algorithms for big data classification: thinking with examples for effective learning (2016): 207-235.
[13] Walsh, Eamon. "Application of the flask architecture to the x window system server." Proceedings of the 2007 SELinux Symposium. 2007.
[14] Hemati, Wahed, Tolga Uslu, and Alexander Mehler. "TextImager: a Distributed UIMA-based System for NLP." Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations. 2016.
[15] Khawas, Chunnu, and Pritam Shah. "Application of firebase in android app development-a study." International Journal of Computer Applications 179.46 (2018): 49-53.