Allegheny College
Department of Computer Science

Senior Thesis

Music Recommendation by Mapping Music and Descriptive Paragraph

by

Xingbang Liu

ALLEGHENY COLLEGE
COMPUTER SCIENCE

Project Supervisor: **Janyl Jumadinova**
Co-Supervisor: **Gregory M. Kapfhammer**

May 1, 2020

Abstract

The current tools cannot help people find music based on user emotions efficiently. What they have are genre tags such as happy and sad. It takes extra time to listen to each song in the genre and find the desired one. The way they recommend music is by using the user browsing history or comparing the playlist between similar users. These recommendation methods cannot detect or predict the current user emotions. This project implemented a system to recommend music tracks based on user-submitted texts. The text can be the form of diary or video transcribes, which could contain users' current emotions. The recommended music can be found by matching the text with the song lyrics. By using the system, people can fill their emotional needs whenever they want. This system utilized Flask to be the front-end and back-end framework. Python implemented text rank was used to summarize the text. The Monge-Elkan similarity was used to match the user inputted text and lyrics. SQL database was used to store the necessary information. In this project, only English songs were kept, others were filtered. To test the efficiency of the system, participants were asked to use the system and assess the system in a survey. The result of the experiment was insignificant because the responses were insufficient. To improve the project, the experiment should be redesigned, and more participants should be invited.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

People use music to fulfill their emotional needs. However, it is normal to have the situation that people cannot find the right music track for certain emotional states by using the current tools. The proposed project tries to implement a system to recommend music based on users' emotional states.

1.1.1 Human Emotion and Music Emotion

Humans enjoy music for its capability to express emotions. A study has shown that music can evoke human emotions and influence human moods [28]. In the experiments mentioned by Przybysz, some sad or happy music can cause human physical reactions such as muscle tension and hormone release. This is one of the reasons for humans to have certain emotions toward music. However, sometimes music emotion and human emotion are separate. Human and music may carry the same emotion, for example, sadness, but human sadness and music sadness may be different. The human may be seeking temporary separation from the real world. Context outside of music could also arouse human emotions. For example, when people are at a funeral, they are expected to be sad. Similarly when song lyrics or song names are sad, people are expected to be sad.

As a result of the popularity of listening to music and the advent of the internet, music service companies started to provide people access to music content and convenient service bundles. Based on user demands, music services have different strategies to satisfy users. For example, some users want to access a large number of music tracks for a low cost, while other users want to discover new music content in a smart way [25]. Therefore, Spotify provides music streaming services, and users have the option to upgrade to Spotify premium. With Spotify premium, users can skip songs, make tracks offline with users' options, and more. With a good music service, users have access to music playlists published by others to play at coffee shops, when exercising, and at weddings.

1.1.2 Easy Music Discover

Due to the large amount of music content, it is often hard for users to curate playlists for certain occasions or events and discover new music [25]. Therefore, some music service providers developed music recommendation systems for users to find music content smartly. Spotify's Weekly Discovery is a good example for users to expand their playlist based on the music they saved previously. In industry, music recommendation systems usually work in two ways. The first way is to find new music content based on users' previous musical preferences. The second way is to match users who have similar actions and music preferences, then cross recommend songs to each other. Since different users could not possibly have the same music playlist, it would be efficient to share different songs to each other.

These music recommendation services are accurate, users may enjoy it, but this system has a limitation. It only digs deeper for user interests, but it does not expand them. The reason is that the information for music recommender systems to use only contains user interaction toward music tracks. The system can classify and rank music tracks that people like or dislike, but they cannot get information about the user's emotional state.

1.1.3 User Current Emotion State

As mentioned before, the relationship between music emotions and human emotions are complicated. Not only can some music evoke human emotions [38], in some situations, human emotions can also be evoked by the surroundings or lyrics. Humans tend to seek music that reflects their current emotional state. In situations when people are sad, they want corresponding sad music tracks to help them relieve mental anguish. The recommendation system will recommend a list of songs based on users' previously saved music. However, users not only have sad music in their playlist, but they also have happy music. It is hard for users to locate the songs they want. Although some music services have a sad music classification, users still have to explore more for the music that they want, often to match the context of the situation or emotion.

The important part here is to recognize the emotional state of the users. If music service recommends music content based on users' current emotional state, the results would be more accurate in terms of users' current demand. Therefore, it is better to let users express themselves. Allow users to define their own emotional states. This way, users can get more accurate music recommendations because users know their requirements the best. This solution can also provide a wider range of recommendations because it does not depend on user interaction history. Music content suggested by this theory would be potentially different from users' past preferences.

1.2 Current State of the Art

There are various music services, such as Spotify, Amazon Prime Music, Pandora, and YouTube Music, that people like to use. The most popular music service on the market would be Spotify. According to MIDiA [22], a company analyzes media and technology, Spotify owns 83 million subscribers and 36% of the market share [24] at H1 of 2018.

1.2.1 Spotify

Spotify has different clients on different platforms. For example, there are mobile clients, desktop clients, TV clients, Web player, and more. Users can choose freely between different platforms. In each Spotify platform, the user can choose their preferred music content to play [39]. As smart home technologies got more and more popular, Spotify added functions that can control playback devices on different platforms. For example, users can control Google Home to play music from Spotify using smartphones. Besides the basic music player function, users can also add other users as friends and share music content.

Spotify also has a strong music recommendation system to help people get more new music that they like. Users can get recommendations from Discover Weekly, Daily Remix, and song radios. There are three main techniques Spotify uses to get recommendations for the users, which are Collaborative Filtering, Natural Language Processing, and Audio Metadata Modeling. In other words, Spotify utilizes information from user interactive history and compares across different users, lyrics semantics, music patterns, and audio metadata, to get similar music content.

1.2.2 YouTube Music

YouTube Music is another popular music service on the internet. YouTube is famous for its video services. YouTube users can upload videos they produce. Users who want to enjoy personalized services have to register user accounts. Based on user interests, YouTube recommends videos on their homepage, and under each video that is played by the users, several related videos would be shown for users to pick. Under each video, users can write comments to share their thoughts. Video providers can also interact with other users. Youtube has built a platform for video producers to not only gain popularity through views, but also make money through advertisements on their videos. This is one of the reasons people are willing to upload videos to YouTube.

Youtube has increased in popularity in recent years. In 2013, Youtube announced that they had over 100 hours of video content uploaded every hour [21]. According to a user insight report in 2018, about 47% of users listen to music on-demand on Youtube [6]. With such a good user foundation, music producers and companies started to release music videos on YouTube. Based on a large number of licensed music videos, YouTube started music streaming services. With a good video recommendation system, YouTube combined content recommendation and social media functions. As a result, users can find content easier from similar users.

1.2.3 Netease Cloud Music

Netease Cloud Music is a Chinese music service provider. Most of the music licenses were held by Tencent Music at the time of inception. As a new music service provider, to compete with Tencent Music, Netease Cloud Music developed strategies to develop strong music recommendation systems and social media platforms.

They think the essence of a music service is to help users find the music they like efficiently. When users find the music they like, they would usually want to express themselves. Therefore, Netease Cloud Music provides comment sections for users to write their feelings and stories. Users can also share music between friends and other social media platforms. Because of the lack of music licenses, Netease Cloud Music encourages small and new producers to upload music to their platform. They also depend more on music recommendations between similar users. Due to Netease Cloud Music's good reputation among users, they got a great number of fundings recently [34].

1.3 Goals of the Project

A new system was proposed to match descriptive paragraphs with music tracks. This way, users can express their emotions whenever it is necessary, and a complex mechanism to detect the user's current emotion state can be avoided. Besides simple emotion detection, they can also expand their musical interest by exploring more random but relevant music content.

1.3.1 Scenarios

The project can be used in multiple scenarios. For example, a person who breaks up with his or her girlfriend or boyfriend would be very sad. People would listen to sad songs when that happens. To find the right music tracks to identify their exact feelings, the person who is experiencing negative emotions can write "descriptive" paragraphs. These paragraphs would contain the sentiment of that person, and the story behind the negative emotions. It would be the same for people experiencing positive emotions, such as falling in love with someone.

The scene does not limit to self-generated emotions. For example, a person who watches a film or television could find himself or herself related to the work. The person would listen to related music tracks to express appreciation. Or the filmmakers want to find relevant music that matches the scene of the film, even contains matching metaphor. They can use a description of the film or the film scripts as the descriptive paragraph. By mapping the key factors of the paragraph with the music key factors, the relevant music tracks would be identified.

1.3.2 General Workflow

Before the user uses the service, pre-processing is required to get extra music contexts, such as lyrics and author. The lyrics of the music track would be analyzed, and summarized, only details containing potential emotions would be kept. Finally, the result would be stored in the server's database.

After the pre-processing is done, this system would take user inputs. However, users are lazy, it always requires more to motivate people to generate sufficient inputs. Therefore, the format of the input has to be closely related to potential emotional states. These inputs can be a diary, personal blog, video script, or description of the video content.

After getting a sufficient amount of user input, the system would analyze the input, break them down into pieces and summarize them into fewer sentences. This extracted information would be used to compare with the language model in the music database. Specifically, the similarity would be checked between user description summarization and music lyrics summarization.

Finally, the system will return a list of relevant songs to users. The general workflow of the system is shown in figure 1.1.

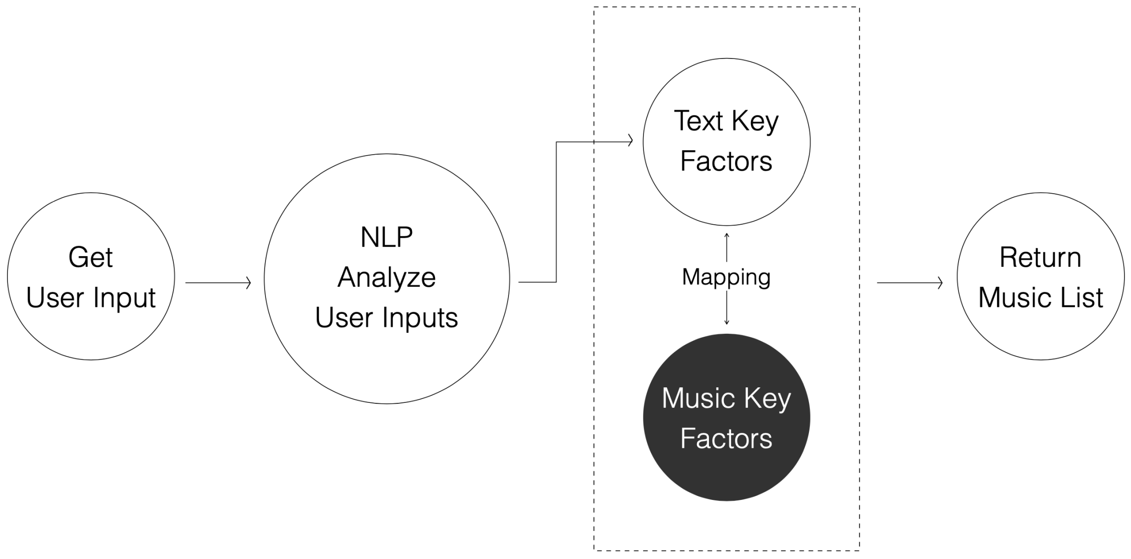


Figure 1.1: General Workflow

1.4 Thesis Outline

In this project paper, five sections will be used to introduce and analyze the proposed music recommendation system.

- The introduction will discuss the motivation of the project. Popular music services will also be introduced and analyzed. Besides motivation and popular music services, the expected goal of the music recommendation system and the general process steps will also be explained.

- The related work section will introduce similar studies. Each study will be introduced and analyzed toward their progress and limitations.

- The method section will describe how the system is build. Packages and tools used to build the system will also be introduced.

- The experiment section will discuss the evaluation of the system. The evaluation process and final results will be introduced and analyzed.

- Finally, the key findings and general ideas will be summarized in the conclusion section.

Chapter 2

Related Work

As a popular entertainment business, music services have always been a popular topic in the industry. Both the recommendation system and music emotions have many existing research projects.

2.1 Spotify Recommendation System

A recommendation system or recommender system is an algorithm that can suggest user-preferred items to a user by giving score or probability to items [33]. In the music recommendation system, the item to suggest would be a song.

Take Spotify as an example, it has three parts in their recommendation system, which are Collaborative Filtering, Natural Language Processing, and Audio Metadata Modeling. Spotify provided an effecient and abundent recommendation system in general. However, it does not provide current emotional state prediction. Users have to browse vague mood playlists and find their desired music. The proposed system actively detects user current emotional state and recommends music accordingly.

2.1.1 Collaborative Filtering

Collaborative Filtering [36] is an algorithm that recommends new content between similar users or items. The algorithm predicts user preferences by learning from users' past ratings for items [13].

Item-based

The first way to do collaborative filtering is to predict the relationship between items. Therefore, calculating item similarity is the first step. The common ways of calculating similarity for item i and j, sim(i,j), could be cosine similarity, Pearson correlation, adjusted cosine, or conditional probability.

Let the targeted users be the set U, let $r_{u,i}$ and $r_{u,j}$ be user's rating over item i and item j. Then the cosine similarity is calculated by equation (2.1).

$$\text{sim}(i,j) = \cos(\mathbf{i},\mathbf{j}) = \frac{\mathbf{i}\cdot\mathbf{j}}{\|i\| * \|j\|} = \frac{\sum_{u\in U} r_{u,i}r_{u,j}}{\sqrt{\sum_{u\in U} r_{u,i}^2}\sqrt{\sum_{u\in U} r_{u,j}^2}} \tag{2.1}$$

Let $\bar{r}_u$ be the average rating for user u. The adjusted cosine can be calculated by equation (2.2) [2].

$$\operatorname{sim}(i,j)=\frac{\sum_{u\in U}\left(r_{u,i}-\bar{r}_u\right)\left(r_{u,j}-\bar{r}_u\right)}{\sqrt{\sum_{u\in U}\left(r_{u,i}-\bar{r}_u\right)^2}\sqrt{\sum_{u\in U}\left(r_{u,j}-\bar{r}_u\right)^2}} \tag{2.2}$$

Let $\bar{r}_i$ be the average rating for item i. The Pearson correlation can be calculated by equation (2.3). One good thing about correlation similarity is that it can calculate how close are the items related [13].

$$\operatorname{sim}(i,j)=\frac{\operatorname{Cov}(i,j)}{\sigma_i\sigma_j}=\frac{\sum_{u\in U}\left(r_{u,i}-\bar{r}_i\right)\left(r_{u,j}-\bar{r}_j\right)}{\sqrt{\sum_{u\in U}\left(r_{u,i}-\bar{r}_i\right)^2}\sqrt{\sum_{u\in U}\left(r_{u,j}-\bar{r}_j\right)^2}} \tag{2.3}$$

Finally, the conditional probability can be calculated by equation (2.4).

$$\operatorname{sim}(i,j)=P(j|i)=\frac{f(i\cap j)}{f(i)} \tag{2.4}$$

After obtaining the similarity between items, a value of the item for the targeted user can be predicted. It means to obtain how the user rates similar items for specific item i, the record we obtained could be the history records.

User-Based

Another way to do collaborative filtering is to find the users that are similar. After knowing the similar users, and the item similarity score is obtained, the recommendations can be predicted.

Take Spotify as an example, after collecting the user interaction, say liking a song, a unique coordinate will be generated for each user. The collected fields are the dimensions. The item-based suggestion will be coming from analyzing the user playlists, and the user-based suggestion will be coming from comparing different users. Once the similar users are identified, the algorithm will simply recommend contents that one user has and others do not.

2.1.2 Natural Language Processing

Natural language processing is the study of the interaction between computers and human languages. Human language can be speeches or writings. This field has many sub-fields, for example, speech recognition, text summarization, and text generation. By using natural language processing techniques, the meaning of lyrics can be classified, which can be further used to identify the music genre. The sentiment of the lyrics can also be analyzed, which can be used to identify the mood of the music tracks.

Text Sentiment Classification

Text classification is a very important sub-field of natural language processing, many real-life applications depend on text classification. For example, web search, document classification, and information ranking all utilize text classification [18]. The classification can be based on different aspects, in music recommendation examples, sentiment can be a feature. Sentiment analysis is the contextual mining of text emotions, for instance, positive or negative. Happiness, joyful, and compliment emotions can be categorized as positive emotions. Sad, angry, and guilt can be categorized as negative emotions. The general text classification workflow is shown in figure 2.1.

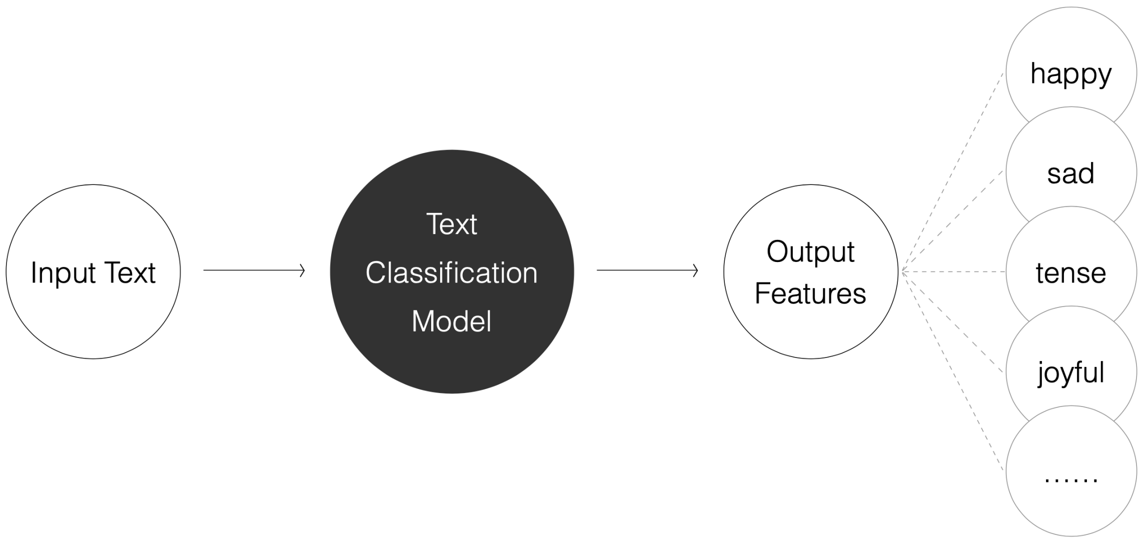


Figure 2.1: Text Classification Workflow

The first step to classify text is to treat text as a bag of words [11]. Bag of words is a model in natural language processing that treats texts as unordered words, and the frequency of each word is recorded. This process is usually conducted by tokenization, and followed by stemming and lemmatization. Stemming and lemmatization can delete the additional part of the word, remain only the stem part. For example, after stemming and lemmatization, the word "interesting" would become "interest". After stemming and lemmatization, stop words, which are words that have no actual meaning in a sentence, will be removed from the bag of words. An example of a stop word can be "and".

One way to classify texts is a Naive Bayes algorithm. Naive Bayes is based on Bayes' probability theorem [32]. In Bayes' probability theorem, for any event A and B, the conditional probability of A given B is equation (2.5).

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{2.5}$$

Multinomial naive Bayes is one of the ways to use the Naive Bayes algorithm to classify texts. It uses the term frequency, $tf(t,d)$, where t is term, and d is the active document. Let $tf(t,d)$ be the frequency of term t in document d, and n_d be the total number of terms in document. As the equation (2.6) shows.

$$\text{term frequency } = \frac{tf(t,d)}{n_d} \tag{2.6}$$

Spotify uses textual information from the internet to find key terms for music content. The terms will be used to build vectors for each song. A similar technique used in collaborative filtering will be applied to find similar content. This technique has a flaw that the textual information gets from the internet is too broad. It would be better if the information is from the comments under each song directly. That way, each song will have a direct correlation to the key terms. Except for comments, users' search history could also be used as part of the text [12].

2.1.3 Audio Metadata Modeling

Audio metadata is the information for the audio files, it usually contains names for artist, album, title, genre, track number, and more [7]. After sampling the audio, more information can be included in audio metadata, for example, loudness, tempo, and more. This information can be used to predict potential music or human emotions. Tempo represents the speed of the audio, it is related to human emotions according to studies. Fast tempo and emotions like joy and fear have correlations, slow tempo and emotions like sadness and tenderness have correlations [19].

Spotify has a database that contains the audio metadata. These aspects can be used to build vector representations. By applying a similar technique in collaborative filtering, similar content will be identified. Bogdanov et al. proposed a way of classifying music by using audio metadata [10]. They asked participants to choose their preferred music and categorized them based on their preferences. They then modeled the audio files and classified them. After that, they correlated the classified features with predefined categories. By using this model, they can suggest songs based on audio metadata.

2.2 Content-Based Music Information Retrieval

Besides collaborative filtering, content-based music information retrieval is another effective way of recommendation. This method collects information that describes the music, and uses the information to suggest new content. Audio metadata can be one of the sources of the information.

One application is to collect and classify audio signal features, then apply them to match based on different situations. In general, there are two ways of matching, which are query by example, and query by humming [20]. Query by example means getting the audio signal as input and return music metadata as output. This query method works on the specific music track, but not the variation tracks, for example, a cover version. Query by humming takes melody as input and returns similar tracks. This application is more of a searching or matching algorithm. The limitation of this recommendation method is as mentioned previously, lack of novelty [20]. It only expands on users' saved songs, and it cannot predict user situation.

The other application is to combine the information which describes the music with user preferences. The user preference does not mean the user ratings, content-based music recommendation does not rely on user ratings [13]. However, categorizing music content based on descriptive information requires experts of the fields to set rules. A similarity check is also required for the content-based suggestion. The way of calculating the similarity between two items is through calculating their distance. The common ways of calculating such distance are Euclidean distance (2.7), Manhattan distance (2.8), vector cosine distance(2.1), Mahalanobis distance (2.9), and Chebyshev distance (2.10).

$$d = |x - y| = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2} \tag{2.7}$$

Euclidean distance measures the straight line between two points in N-dimensional space [5].

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i| \tag{2.8}$$

Manhattan distance is also a measurement of the straight line between two points in N-dimensional space. Just imagine the street blocks in Manhattan, there are two paths from one intersection to the other one. However, the true distance between them is a straight line cut through the buildings [15].

$$d(x, y) = \sqrt{(x - y)^T S^{-1} (x - y)} \tag{2.9}$$

Mahalanobis distance is a measurement of the straight line distance between two points in N-dimensional space. Mahalanobis distance handles better with correlated points(≥ 2). In situations that there are more than 2 points that are correlated, the axes of the points are no longer orthogonal, and they cannot be plotted in 3-dimensional space [35].

$$d(x, y) = \max_{i=1 \ldots n} |x_i - y_i| \tag{2.10}$$

Chebyshev distance, also called maximum value distance, is similar to Euclidean distance and Manhattan distance. It calculates the straight line distance based on point coordinates.

There are limitations for this application when new users entered the system, they can't get the right suggestion because the system needs time to adjust to the user preferences. This situation is also called the cold-start problem [13]. Another issue is feature extraction. It is hard to extract high-level descriptions such as mood. These high-level descriptions are very important in accurate and personalized recommendations. Content-based music recommendation is different than the proposed system because it does not detect users' current emotional state. Content-based music recommendation is similar to collaborative filtering in some sense because they all depend on historical information, for example, listening history, ratings, and music meta information.

2.3 Contextual Music Retrieval

Contextual music information is any information, other than music content information, that describes the music content itself. Contextual music suggestion means to recommend music based on the user's actual situation, for example, emotional state. This method takes three sources of information, environment-related context, user-related context, and multimedia context [20]. Environment-related information can be the season, temperature, time, and weather because all these contexts can influence human emotions. User-related information can be user activities, such as exercising and driving, user profiles, such as social network statements, and emotional state. Multimedia can be text and images relevant to the music. This information can be used to profile and predict the user's actual situation. After the necessary information is gathered, similarity prediction, classification, and other techniques can be used to predict user preferences. This way of recommendation is similar to the proposed recommendation system. They both use side information to predict user emotional state. However, their intentions are different. The contextual music recommendation lets users passively get the recommended music, but the proposed system lets users actively require for music recommendations. Based on their different intentions, the scenario of usage can be different. For example, the contextual music recommendation works in users' daily usage, but the proposed system works only when users have the emotional demand.

A way of achieving contextual music retrieval is proposed by B. Han et al [16]. They proposed an emotional state transition model which models music-evoked human emotions, and content-based music recommendation ontology to profile user preference and context. This system provides music by mapping high-dimensional music features with the emotional state transition model. According to the paper, this method achieved 67.54% overall accuracy. The limitation of this method is that it needs a large amount of data, and the human emotional state can be changed based on the social environment. Once the social context is changed, more data would be required to get higher accuracy.

Recommendation based on contextual music retrieval is still new, and it has great potential to contribute to the music recommendation system.

Chapter 3

Method of Approach

This project is about providing a new concept that recommends users with their potentially preferred music. Although it is only a minimal viable prototype, it would be good to make it a web application hosted on a website. This chapter contains the tools utilized for this application and the workflow for how these tools work with each other.

3.1 Tools Utilized

3.1.1 Flask

Flask served as both front-end and back-end tool, it is a micro web application framework, which means it does not require other third-party libraries and services. It is a stand-alone framework that is easily extended [27]. Flask is implemented in Python, which is a great fit for this application because this project processes large amounts of textual data. Python is famous for its simple syntax and abundance of libraries, which means it is a good fit for big data processing. Simple syntax means the code would be easy to read, and the processing flow would be easily translated into actual code.

Flask Front-end

Flask uses HTML templates to render the front-end pages. Any values that want to be changed dynamically when running the website can be passed into HTML files by using variables. For example, in route functions, add the variable in the return statement as 3.1 shows. Then go to the rendered HTML file, add 3.2 in the corresponding place.

Listing 3.1: Flask Route Template Variables

```
return url_for(
  "html_file_name",
  outgoing_variable_name = incoming_variable_name,
  )
```

Listing 3.2: HTML Python Variables

```
{{outgoing_variable_name}}
```

In general, HTML templates are the frame of the pages, Flask controls the dynamic variables and rendering strategies. In HTML files, *form* were used to collect user answers. As the code 3.3 shows.

Listing 3.3: HTML Forms

```
<form method="POST">
   Title of text: <input type="text" name="title">
   <br>
   <textarea name = "text" rows="25" cols="80"></textarea>
   <br>
   <input type="submit" value="Submit">
</form>
```

Flask Back-end

In Flask, modules can be imported by using Python importing statements. All the setups can be put inside the *__init__.py* file for initialization. For example, in 3.4.

Listing 3.4: Python Project Initialization

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from flask_bcrypt import Bcrypt
from flask_login import LoginManager

# App Configurations
app = Flask(__name__)
# secret key to protect request and cookie
# use
# import secrets
# secrets.token_hex(16)
# to generate token.
app.config["SECRET_KEY"] = "__keys__"
# utilize db
app.config["SQLALCHEMY_DATABASE_URI"] = "sqlite:///site.db"
db = SQLAlchemy(app)
bcrypt = Bcrypt(app)
login_manager = LoginManager(app)

from music_main import routes
```

After importing and setting up all the necessary libraries, they can be used as normal Python programs. Other tools used in the back-end will be explained in other sections of this chapter.

3.1.2 SQLAlchemy

With such a big amount of data, the best way of saving and retrieving data is by using databases. Flask SQLAlchemy is a Flask extension that wraps the SQLAlchemy library for use with flask. The SQLAlchemy library simplifies the use of Structured Query Language (SQL) by creating object relational mapping [26]. To create a database using Flask SQLAlchemy, first, create the database then initialize the library 3.5, afterwards define the database as models 3.6. Use 3.7 to create tables corresponding to the models in the database.

Listing 3.5: Flask SQLAlchemy Utilization

```
app.config["SQLALCHEMY_DATABASE_URI"] = "sqlite:///site.db"
db = SQLAlchemy(app)
```

Listing 3.6: Flask SQLAlchemy Model

```
class User(db.Model, UserMixin):
    id = db.Column(
      db.Integer,
      primary_key=True,
    )
    username = db.Column(
      db.String(20),
      unique=True,
      nullable=False,
    )
    password = db.Column(
      db.String(60),
      nullable=False,
    )
    input = db.relationship("Input",
    backref="author",
    lazy=True,
    )

    def __repr__(self):
        return f"User('{self.username}')"
```

Listing 3.7: Flask SQLAlchemy Create Database

```
db.create_all()
```

Relational Database

This project used the relational database to manage user data, lyrics data, and processed data. The reason to choose the relational database over the non-relational database is that the data used in this project is strictly structured. When comes to retrieving results, it would be easier to use strictly defined tables. The data used in this project can be divided into three tables. The user table stores usernames, passwords, and their relation to other user data. The lyrics table stores the title of the song, the singer of the song, the lyrics of the song, and the Summarization of the lyrics. The input table stores the title of the user text, the user inputted paragraphs, the Summarization of user input texts, and the top five results of each text matching processing. The table design and their relations are as the graph 3.1 shows.

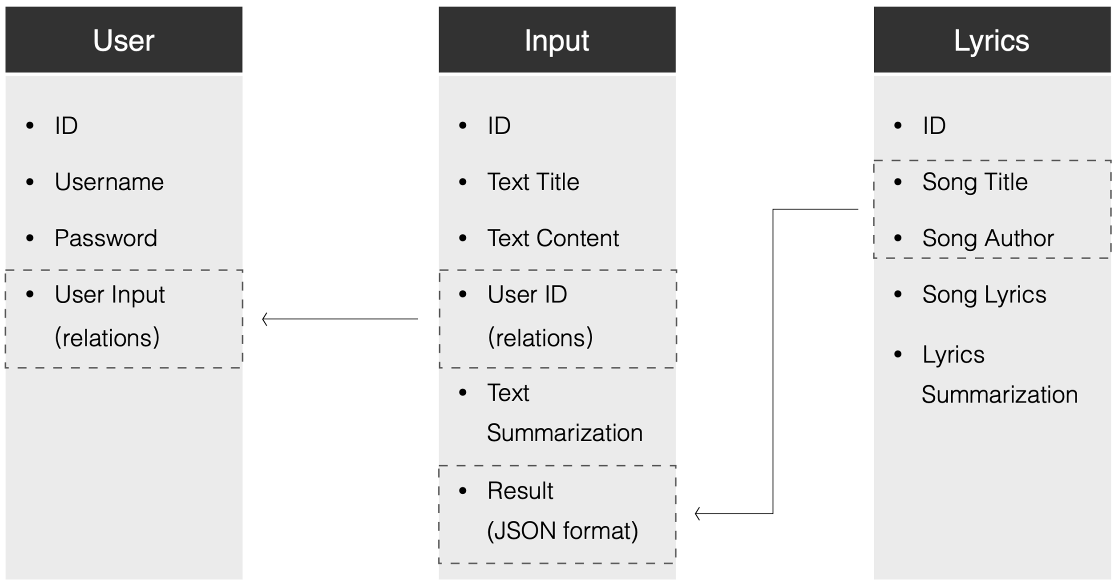


Figure 3.1: Database Design and Relation

3.1.3 Lyrics API

All the lyrics this project used were sourced from genius.com. Genius is a website provides music knowledge such as lyrics and story behind the music [1]. The developer website provides an API to access their music knowledge.

LyricsGenius

LyricsGenius is a tool to retrieve lyrics from genius.com on GitHub that was written in Python [17]. This tool utilized Genius API to retrieve song lyrics, but users have to register the Genius developer account and then register for application to get client access token. To get the lyrics, use the *search_song* function to get a Python object, then access the lyrics attribution. As the code 3.8 shows.

Listing 3.8: LyricsGenius Sample Code

```
import lyricsgenius

genius = lyricsgenius.Genius("my_client_access_token_here")
song = genius.search_song(song_title, song_artist)
```

```
song_lyrics = song.lyrics
```

Although this tool would return None when there is no song found according to the search term, the server would return vague results. Therefore, regular expression was used to filter out these vague results.

Playlists

To get a list of song titles and song authors, a website called playlist-converter.net [3] was used to convert saved Spotify playlist into CSV format files 3.9. Then the file was imported as Python list objects and they were further fed into the LyricsGenius tool. In this project, a total of 4 playlists with 593 pop songs was added to the database. The name of the Spotify playlists are: *Happy Songs (80s, 90s, 2000s, 2010s & 2020s _ Best Happy Hits & Oldies*, *POP Music Playlist - Best POP Hits of All Time (Updated in 2020)*, *Sad Songs*, and *Top Pop*.

Listing 3.9: Spotify Pliaylist

```
"track","artist"
"Twist And Shout - Remastered","The Beatles"
"Uptown Funk (feat. Bruno Mars)","Mark Ronson"
"A-Punk","Vampire Weekend"
"Hooked on a Feeling","Blue Swede"
"Come","Jain"

...
```

Language Detection

For now, the only language model used was English, so, song language other than English was also filtered. Language detection tool, Spacy-langdetect, based on Spacy, was used to detect whether the song lyrics are English [9]. To use the tool, Spacy and Spacy English language model should be installed. Then, use the Spacy and Spacy-langdetect pipeline to create the object. As the code sample 3.10 shows.

Listing 3.10: Spacy Language Detection

```
python -m spacy download en

import spacy
from spacy_langdetect import LanguageDetector

nlp = spacy.load('en')
nlp.add_pipe(
  LanguageDetector(),
  name='language_detector',
  last=True,
)
doc_check = nlp(text)
```

```
if doc_check._.language["language"] == "en":
    return True
```

3.1.4 Text Summarization

To filter out the redundant texts and retain only the useful texts, a Python library called Pytextrank was used to summarize each text entry. Pytextrank was implemented based on the Spacy pipeline, and Mihalcea 2004 paper [23], it can rank the phrases and sentences [14]. Based on the ranking, the top phrases were used to formulate the summarized sentences. To use Pytextrank, first install the dependencies, which are Spacy, NetworkX, and GraphViz. Then download the Spacy English model. After adding Textrank into the Spacy pipeline, import the text into the pipeline object. Finally, use the *summary* function to get the Summarization of the text. As the sample code 3.11 shows.

Listing 3.11: Pytextrank Sample Code

```
python -m spacy download en_core_web_sm

import spacy
import pytextrank

nlp = spacy.load("en_core_web_sm")
tr = pytextrank.TextRank(logger=None)
nlp.add_pipe(tr.PipelineComponent, name="textrank", last=True)
doc = nlp(text)
whole_sent = ""
sum = doc._.textrank.summary(
  limit_phrases=15, limit_sentences=5
)
for sent in sum:
  whole_sent = whole_sent + repr(sent).rstrip() + " "
```

3.1.5 Text Matching

There are many texts matching algorithms, for example, Levenshtein, Jaccard index, and cosine similarity. Although they all compare text entries, each algorithm only has good performance in certain areas. In situations like spelling check or keyword search, Levenshtein would be a good fit. Levenshtein algorithm used Levenshtein distance, it measures the minimum character edit steps from one string to another. Similar algorithms are Hamming and Jaro-Winkler. They are called edit-based similarities. Edit-based is simple and they are very efficient in short strings. However, they do not consider the meaning of the strings. Algorithms such as the Jaccard index and cosine similarity are called token-based similarities. The token here is a unit of the string, for example, n-gram. They are efficient in long strings, and they take the meaning of the text into account. However, they are not very efficient in short strings. These algorithms are a good fit for areas such as text mining and bioinformatics.

This project used a hybrid algorithm called Monge-Elkan. It combines both edit-based similarities and token-based similarities. It combined the advantages of edit-based similarities and token-based similarities. However, they can be slower than other similarity check algorithms. The tool this project used is called Py_stringmatching. It has a module to compute the Monge-Elkan similarity of two strings [37].

To use Py_stringmatching, first import the library. Then initialize the *MongeElkan* function. Finally, use the *get_raw_score* function to compare two string entries. As the sample code 3.12 shows.

Listing 3.12: Monge-Elkan Similarity

```
import py_stringmatching as sm

me = sm.MongeElkan()
return me.get_raw_score(user_sum, lyrics_sum)
```

3.2 Implementation and Workflow

This section explains how does the project organize and how were the tools used to construct the application. The workflow of the project is also explained in this section.

3.2.1 Project Implementation

This subsection explains the prerequisites, library management, and file structure of the project.

System and Python Versions

This project was implemented and deployed on the Linux system. Specifically Ubuntu 18.04.4 LTS 3.13.

Listing 3.13: System Version

```
cat /proc/version
Linux version 4.15.0-76-generic (buildd@lcy01-amd64-029)
(gcc version 7.4.0 (Ubuntu 7.4.0-1ubuntu1~18.04.1))
#86-Ubuntu SMP Fri Jan 17 17:24:28 UTC 2020
```

The Python version used was Python 3.7.6 3.14. To implement the project with the desired Python version, Pyenv was used to avoid version conflict between the installed version of Python with system versions of Python. Pyenv is a simple but powerful Python version management tool [29]. This tool can be used to download the desired versions of Python easily. The scope of the Python can be set to global, which means the desired version of Python can be used system-wise, or local, which means the desired version of Python can only be used in a specific project.

Listing 3.14: Python Version

```
pyenv versions
```

```
  system
* 3.7.6 (set by /home/sheldon/.pyenv/version)
```

Library Management

Many tools and plugins were used in this project. To avoid the dependency conflict, Pipenv was used to manage the packaging. Pipenv is a dependency management tool [30]. With pipenv, all the specifications can be recorded in *Pipfile* 3.15. For example, the Python version, dependencies, customized scripts.

Listing 3.15: Pipenv Pipfile

```
[[source]]
name = "pypi"
url = "https://pypi.org/simple"
verify_ssl = true

[dev-packages]

[packages]
lyricsgenius = "==1.8.2"
spacy = "==2.2.4"
networkx = "==2.4"
graphviz = "==0.13.2"
pytextrank = "==2.0.1"
numpy = "==1.18.2"
pip = "*"
six = "==1.14.0"
py-stringmatching = "==0.4.1"
spacy-langdetect = "==0.1.2"
attrs = "==19.3.0"
bcrypt = "==3.1.7"
beautifulsoup4 = "==4.6.0"
blis = "==0.4.1"
catalogue = "==1.0.0"
certifi = "==2019.11.28"
cffi = "==1.14.0"
chardet = "==3.0.4"
click = "==7.1.1"
coverage = "==5.0.4"
cymem = "==2.0.3"
decorator = "==4.4.2"
idna = "==2.9"
importlib-metadata = "==1.5.0"
itsdangerous = "==1.1.0"
langdetect = "==1.0.7"
more-itertools = "==8.2.0"
```

```
murmurhash = "==1.0.2"
plac = "==1.1.3"
pluggy = "==0.13.1"
preshed = "==3.0.2"
py = "==1.8.1"
pycparser = "==2.20"
pyparsing = "==2.4.6"
pytest = "==5.4.1"
requests = "==2.23.0"
srsly = "==1.0.2"
thinc = "==7.4.0"
tqdm = "==4.43.0"
urllib3 = "==1.25.8"
wasabi = "==0.6.0"
wcwidth = "==0.1.8"
zipp = "==3.1.0"
Flask = "==1.1.1"
Flask-WTF = "==0.14.3"
Flask-SQLAlchemy = "==2.4.1"
Flask-Bcrypt = "==0.7.1"
Flask-Login = "==0.5.0"
Jinja2 = "==3.0.0a1"
MarkupSafe = "==1.1.1"
SQLAlchemy = "==1.3.15"
Werkzeug = "==1.0.0"
WTForms = "==2.2.1"
gunicorn = "*"

[requires]
python_version = "3.7"

[scripts]
music = "./scripts/music.sh"
setup = "./scripts/setup.sh"
test = "./scripts/test.sh"

[pipenv]
allow_prereleases = true
```

With Pipenv, Python virtual environment can be set up and dependencies with users' required versions can be installed easily and recorded into Pipfile by using one line of the Bash command. If there is any conflict between dependencies, Pipenv can also help to diagnose the conflict and provides resolving advice. A complete list of requirements in the requirement file is as follows 3.16:

Listing 3.16: requirements.txt

```
-i https://pypi.org/simple
```

```
attrs==19.3.0
bcrypt==3.1.7
beautifulsoup4==4.6.0
blis==0.4.1
catalogue==1.0.0
certifi==2019.11.28
cffi==1.14.0
chardet==3.0.4
click==7.1.1
coverage==5.0.4
cymem==2.0.3
decorator==4.4.2
flask-bcrypt==0.7.1
flask-login==0.5.0
flask-sqlalchemy==2.4.1
flask-wtf==0.14.3
flask==1.1.1
graphviz==0.13.2
idna==2.9
importlib-metadata==1.5.0 ; python_version < '3.8'
itsdangerous==1.1.0
jinja2==3.0.0a1
langdetect==1.0.7
lyricsgenius==1.8.2
markupsafe==1.1.1
more-itertools==8.2.0
murmurhash==1.0.2
networkx==2.4
numpy==1.18.2
packaging==20.3
plac==1.1.3
pluggy==0.13.1
preshed==3.0.2
py-stringmatching==0.4.1
py==1.8.1
pycparser==2.20
pyparsing==2.4.6
pytest==5.4.1
pytextrank==2.0.1
requests==2.23.0
six==1.14.0
spacy-langdetect==0.1.2
spacy==2.2.4
sqlalchemy==1.3.15
srsly==1.0.2
thinc==7.4.0
```

```
tqdm==4.43.0
urllib3==1.25.8
wasabi==0.6.0
wcwidth==0.1.8
werkzeug==1.0.0
wtforms==2.2.1
zipp==3.1.0
```

File Structure

The project repository contains the application directory. This directory has the HTML templates, application Python files, and the database file. The playlist directory contains all the Spotify playlists. The scripts directory contains the script to set up the environment and download the dependencies automatically then start the service. The test directory contains all the testing program for the application. Texts directory contains all the input files wrote by the users. In the project root directory, there is app main function file, Pipfile, Piplock file, and requirements record text file.

The file structure of this project is as graph 3.2 shows.

3.2.2 Workflow

This section explains the deployment steps of this project, workflow of the process, and the user access flow.

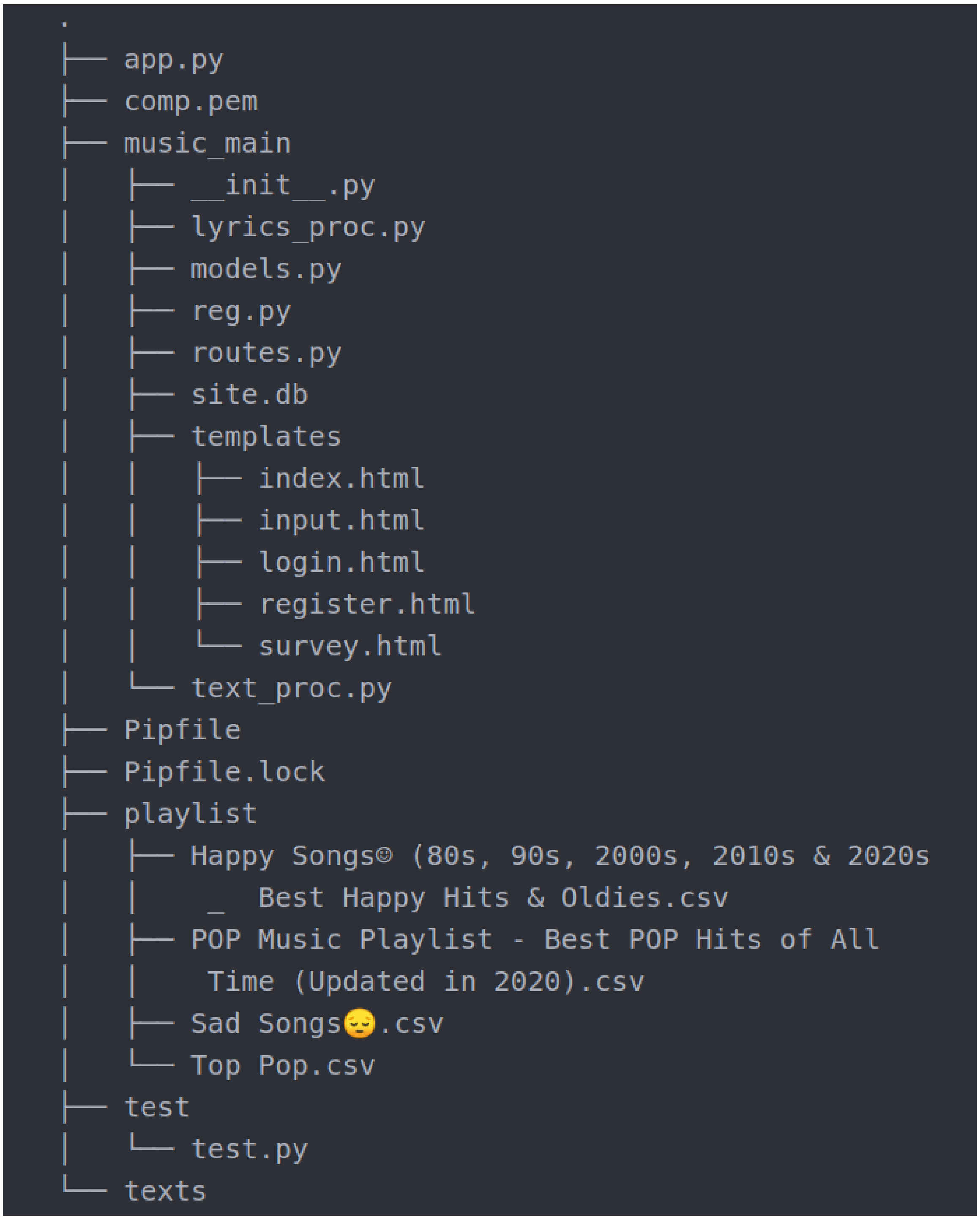
Deployment

After finishing the implementation, the project Pipfile was locked, which means all the dependency versions were locked. To automate the setup process on a new environment, a script was added to the project. Under the script section of Pipfile, add the command keyword and the path of the script, for example, *setup = "./scripts/setup.sh"*. The command to set up a new environment would be *pipenv run setup*. In the setup script, there was a command to download the two English language models, from Spacy, the command to create the databases, and the command to run the Lyrics model to update the Lyrics table with lyrics from Genius, and summarization of each lyrics. The Bash script is as follows 3.17:

Listing 3.17: setup.sh

```
#!/bin/bash

# Download language model
pipenv run python -m spacy download en_core_web_sm
pipenv run python -m spacy download en
# create database
pipenv run python -c "from music_main import db; \
from music_main.models import User, Input, Lyrics; \
db.create_all()"
```



```
.
├── app.py
├── comp.pem
├── music_main
│   ├── __init__.py
│   ├── lyrics_proc.py
│   ├── models.py
│   ├── reg.py
│   ├── routes.py
│   ├── site.db
│   ├── templates
│   │   ├── index.html
│   │   ├── input.html
│   │   ├── login.html
│   │   ├── register.html
│   │   └── survey.html
│   └── text_proc.py
├── Pipfile
├── Pipfile.lock
├── playlist
│   ├── Happy Songs☺ (80s, 90s, 2000s, 2010s & 2020s
│   │    _ Best Happy Hits & Oldies.csv
│   ├── POP Music Playlist - Best POP Hits of All
│   │     Time (Updated in 2020).csv
│   ├── Sad Songs😔.csv
│   └── Top Pop.csv
├── test
│   └── test.py
└── texts
```

Figure 3.2: File Structure Tree

```
# Create data for Lyrics table
pipenv run python -c \
"from music_main.lyrics_proc import *; lyrics_main()"
```

After the above preparations, ssh into the server and clone the repository from GitHub. Install the Pyenv and then the version of Python used to implement the Project. Install the Pipenv, and use *pipenv install* to create a Python virtual environment and install all the dependencies according to the Pipfile. Nginx and Gunicorn were used as web servers during deployment on an Amazon AWS server. The supervisor was used to monitor the application process on the Amazon AWS Linux server.

Workflow

When everything is ready and the application is up and running on the server, the URL of the website can be shared with people who are interested in participating in the experiment.

When the lyrics model was run during the deployment process, the playlist was first combined into one list. Then the list was looped through and was used to get the lyrics from Genius by using Lyricsgenius library. After making sure the result was real lyrics and the language was English by using regular expression and Spacy-langdetect, the lyrics were summarised by calling the summarization function powered by Pytextrank from the text model. Finally, the corresponding information was added and committed to the database table.

After the user filled in their response, the text was saved as text files, and the summarization function was called from the text model. Then the text matching function powered by Py-stringmatching was also called from the text model. The top 5 results were kept, and all the corresponding information was saved into the database table.

The complete system design is as graph 3.3 shows.

Access Flow

When users access the website to use the application and to participate in the experiment, the landing page will contain the brief project Introduction, data usage, how to quit the experiment anytime, compensation lottery participation, experimenter contact information, login link, log out link, and register link. As the graph 3.4 shows. After the user registered and logged in, the user would be landed in the user input page 3.5. Here the user should read the instruction and fill out the blanks. In the result page, a list of no more than 5 songs will be shown to the user, followed by the link to the survey and compensation participation sign up link 3.6. The sitemap of this application is as graph 3.7 shows. The Walk-Through section under Experiments also talked about the access flows.

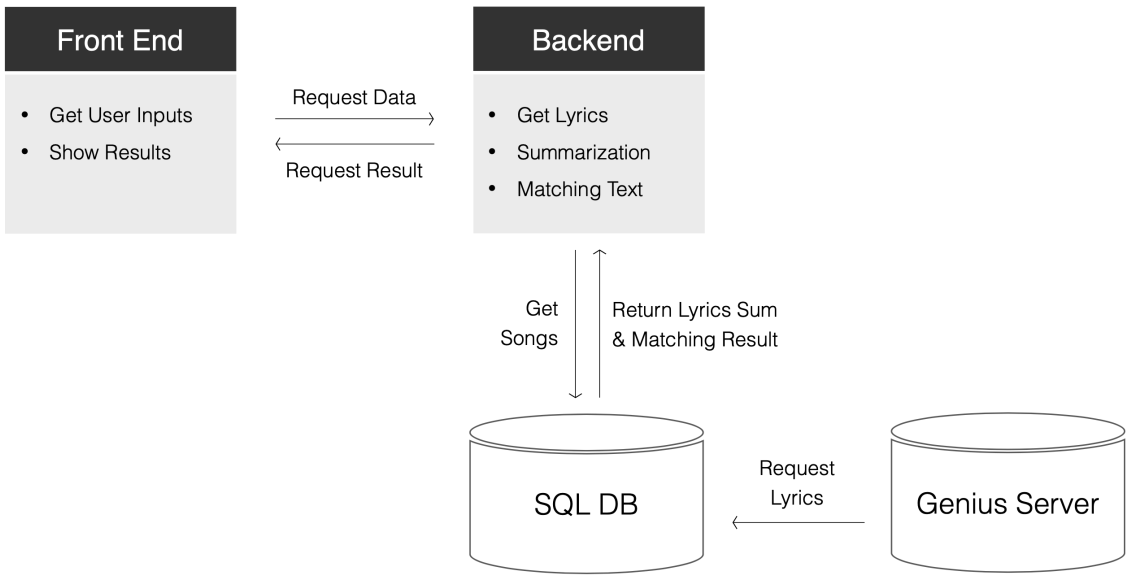
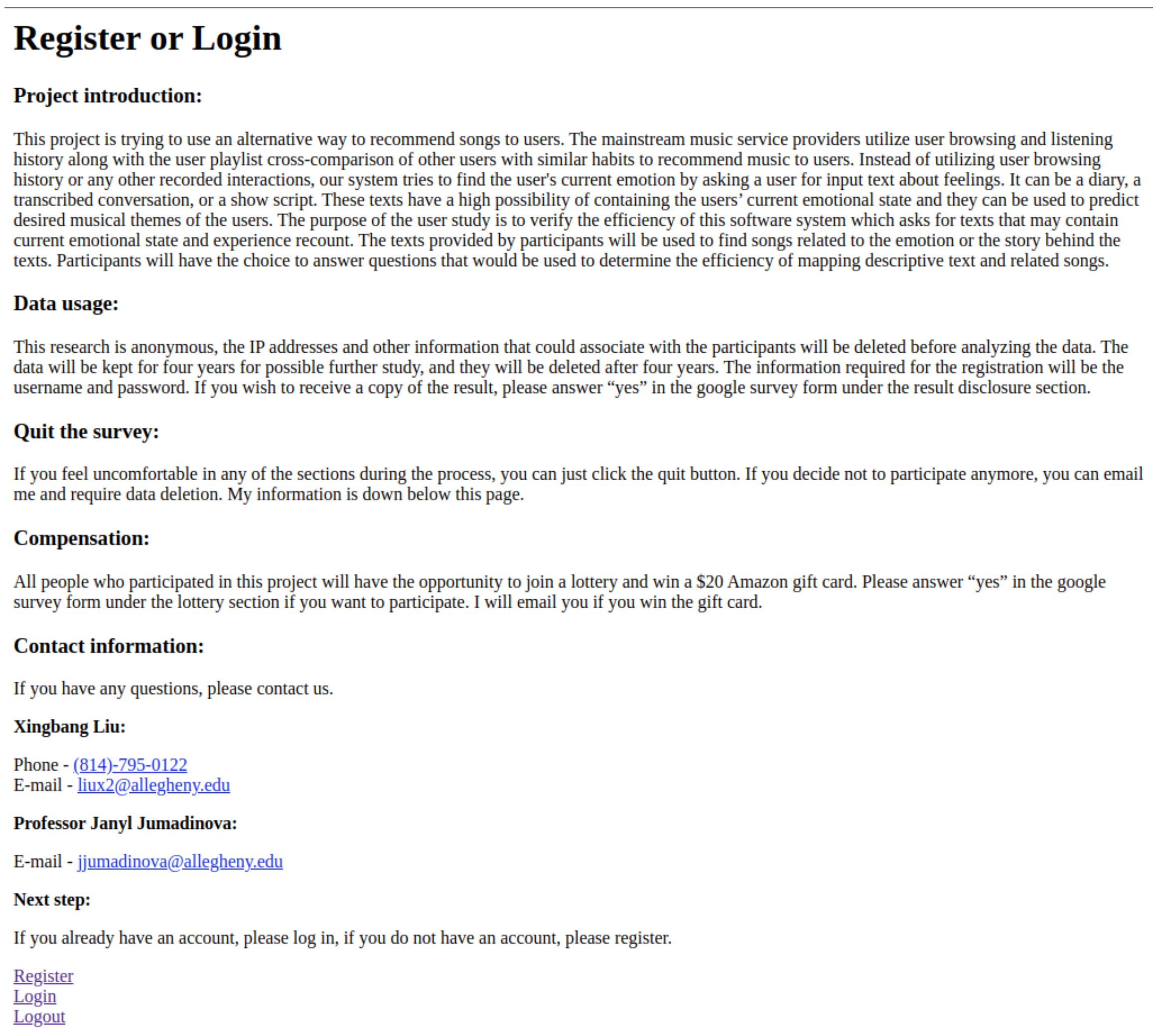


Figure 3.3: System Design



Register or Login

Project introduction:

This project is trying to use an alternative way to recommend songs to users. The mainstream music service providers utilize user browsing and listening history along with the user playlist cross-comparison of other users with similar habits to recommend music to users. Instead of utilizing user browsing history or any other recorded interactions, our system tries to find the user's current emotion by asking a user for input text about feelings. It can be a diary, a transcribed conversation, or a show script. These texts have a high possibility of containing the users' current emotional state and they can be used to predict desired musical themes of the users. The purpose of the user study is to verify the efficiency of this software system which asks for texts that may contain current emotional state and experience recount. The texts provided by participants will be used to find songs related to the emotion or the story behind the texts. Participants will have the choice to answer questions that would be used to determine the efficiency of mapping descriptive text and related songs.

Data usage:

This research is anonymous, the IP addresses and other information that could associate with the participants will be deleted before analyzing the data. The data will be kept for four years for possible further study, and they will be deleted after four years. The information required for the registration will be the username and password. If you wish to receive a copy of the result, please answer “yes” in the google survey form under the result disclosure section.

Quit the survey:

If you feel uncomfortable in any of the sections during the process, you can just click the quit button. If you decide not to participate anymore, you can email me and require data deletion. My information is down below this page.

Compensation:

All people who participated in this project will have the opportunity to join a lottery and win a $20 Amazon gift card. Please answer “yes” in the google survey form under the lottery section if you want to participate. I will email you if you win the gift card.

Contact information:

If you have any questions, please contact us.

Xingbang Liu:

Phone - (814)-795-0122
E-mail - liux2@allegheny.edu

Professor Janyl Jumadinova:

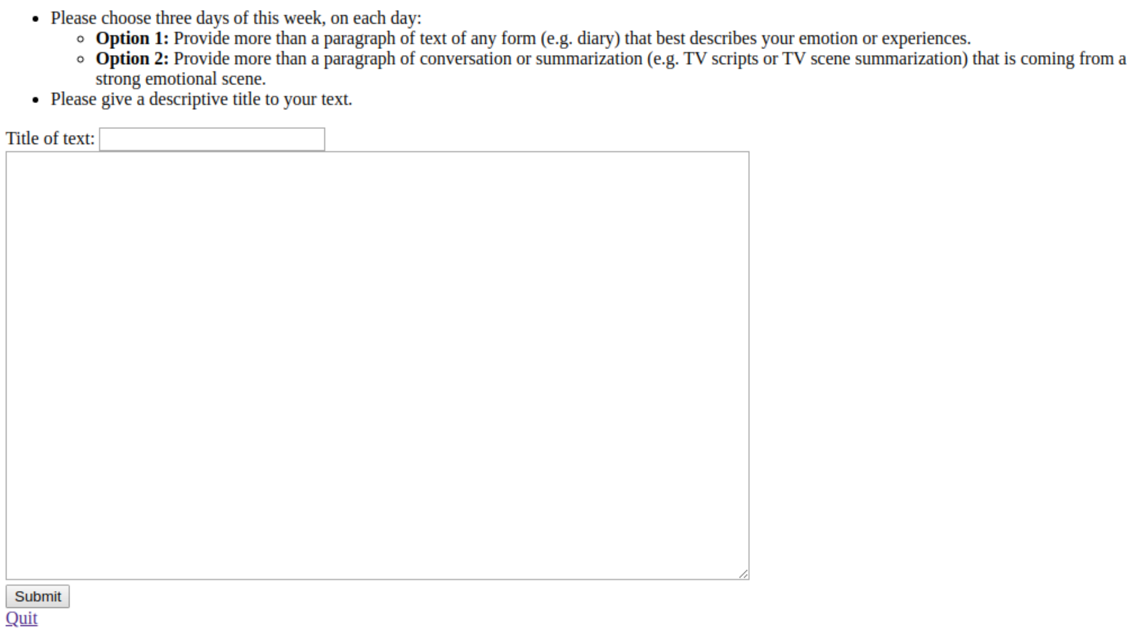
E-mail - jjumadinova@allegheny.edu

Next step:

If you already have an account, please log in, if you do not have an account, please register.

Register
Login
Logout

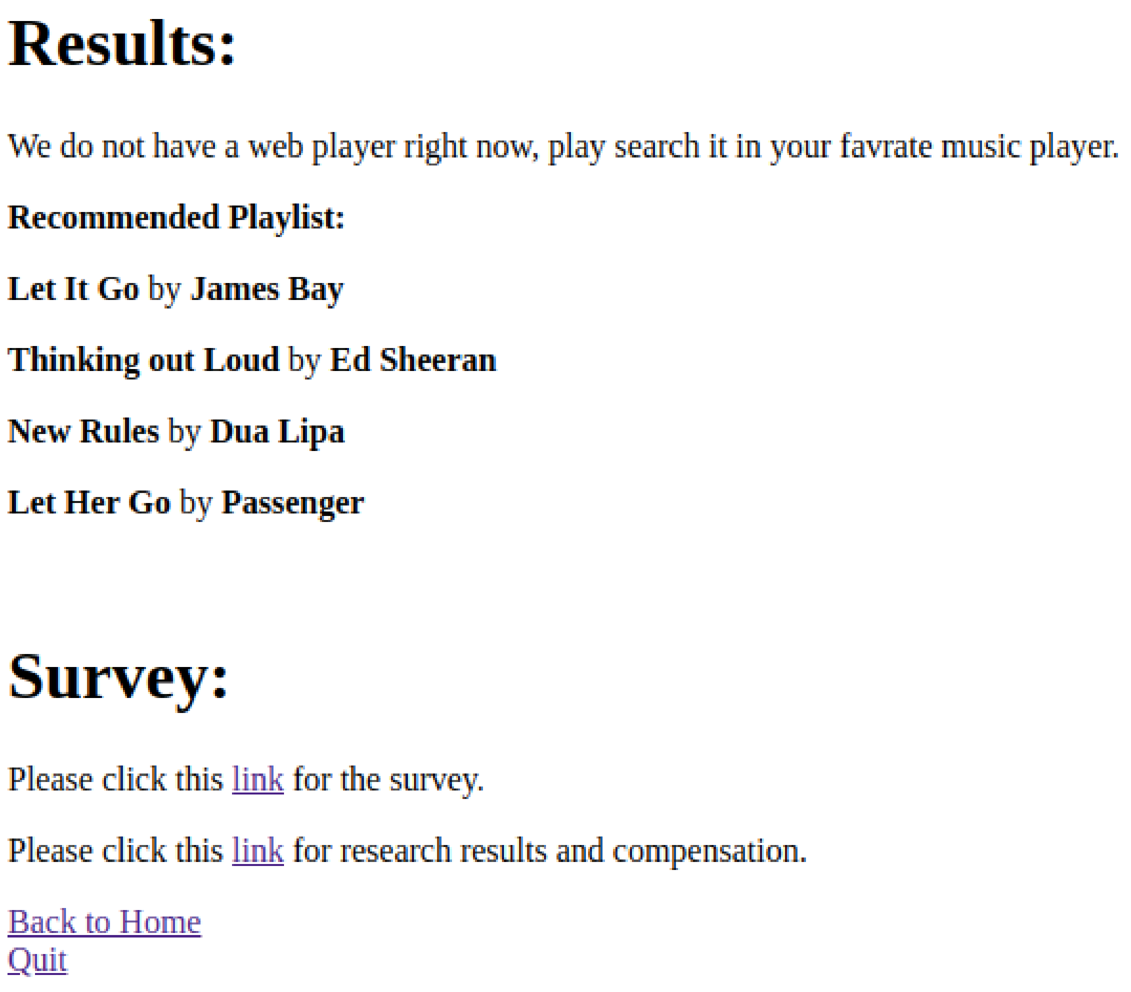
Figure 3.4: Landing Page



- Please choose three days of this week, on each day:
  - **Option 1:** Provide more than a paragraph of text of any form (e.g. diary) that best describes your emotion or experiences.
  - **Option 2:** Provide more than a paragraph of conversation or summarization (e.g. TV scripts or TV scene summarization) that is coming from a strong emotional scene.
- Please give a descriptive title to your text.

Title of text:

Figure 3.5: Input Page



Results:

We do not have a web player right now, play search it in your favrate music player.

Recommended Playlist:

Let It Go by **James Bay**

Thinking out Loud by **Ed Sheeran**

New Rules by **Dua Lipa**

Let Her Go by **Passenger**

Survey:

Please click this link for the survey.

Please click this link for research results and compensation.

Back to Home
Quit

Figure 3.6: Result Page

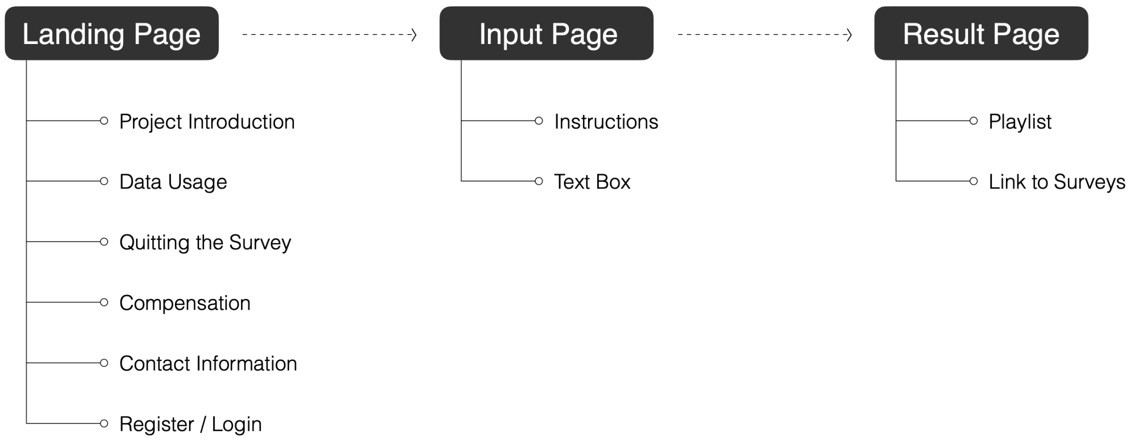


Figure 3.7: Sitemap

Chapter 4

Experimental Results

4.1 Experimental Design

There are two parts in the experiment designed to test the result of this project. The first part is the program test. This part of the experiment tests if there are any bugs in the program. The second part is the user experiment. This part of the experiment tests for the accuracy of the algorithm.

4.1.1 Program Tests

The tool used in the program test is Pytest. Pytest is a framework that can help to build tests for Python programs [31]. The programs tested by Pytest are the lyrics model and text model. The Flask based programs were tested by going through all the pages and the bugs were detected by using Flask debug mode. The reason for not using Pytest-flask to test the Flask based code is that this is not a complex website and it does not affect the accuracy of the algorithm. The core of the text processes is lyrics summarization, user-entered text summarization, and summarization matching.

Lyrics Model Test

The lyrics model is responsible for merging the playlists and downloading the lyrics from Genius server as string datatype. The Language detection function detects the language of the lyrics, all the non-English lyrics were filtered. The data committing function committed the data into the database. The pipeline function was the controller of the lyrics process. There were three functions tested in the lyrics model, the CSV playlist merging function, the lyrics getting function, and the language detection function. The data committing function was tested by manually check the data table in the database. After calling the function, the function object was checked for whether the merged list was empty. The lyrics getting function was tested by using a parametrized test, which three song titles and their authors 4.1 were passed into the function. Finally, the returned object was tested if they were *string* or *None*. To test the language detection function, three English sentences 4.2 were passed into the function to check if the function returns true. The other three German sentences 4.3 were passed into the function to check if the function returns anything other than true.

Listing 4.1: Parametrized Song Title and Author

```
@pytest.mark.parametrize(
    "song_title,␣song_artist",
    [
        ("Country␣House", "Blur"),
        ("On␣Top␣Of␣The␣World", "Imagine␣Dragons"),
        ("Scared␣To␣Live", "The␣Weeknd"),
    ],
)
```

Listing 4.2: Parametrized English Sentences

```
@pytest.mark.parametrize(
    "check_text",
    [
        "This␣is␣English.",
        "This␣is␣not␣English.",
        "Good␣morning!",
    ],
)
```

Listing 4.3: Parametrized German Sentences

```
@pytest.mark.parametrize(
    "check_text",
    [
        "Das␣ist␣Deutsch!",
        "Das␣ist␣auch␣Deutsch!",
        "Wie␣geht's␣dir?",
    ],
)
```

Text Model Test

The text model is responsible for summarizing the string, and the string matching function will calculate the closeness of two summarizations. To test the summarization function, three sets of sentences 4.4 were passed into the function, and the returned object was tested to check if it was not empty. To test the string matching function, three pairs of sentences 4.5 were passed into the function, and the returned object was tested to check if it was floating-point number.

Listing 4.4: Parametrized Summarization Sentences

```
@pytest.mark.parametrize(
    "org_text",
    [
        "This␣is␣English.␣This␣is␣not␣important.",
        "This␣is␣not␣English.␣This␣is␣important.",
        "Good morning! How are you doing?",
    ],
)
```

Listing 4.5: Parametrized Matching Sentences

```
@pytest.mark.parametrize(
    "user_sum, lyrics_sum",
    [
        (
            ["This is English. This is not important."],
            ["I'm speaking English."],
        ),
        (
            ["This is not English. This is important."],
            ["I'm speaking German."],
        ),
        (
            ["Good morning! How are you doing?"],
            ["How do you do?"],
        ),
    ],
)
```

4.1.2 User Tests

Institutional Review Board

The form of user tests is surveying. To conduct the user survey, the designed survey has to be approved by Allegheny College Institutional Review Board (IRB). The Institutional Review Board is an organization that reviews and supervises experiments to make sure the human rights of participants do not get hurt [8]. Before the application starts, experimenters have to finish a Collaborative Institutional Training Initiative (CITI Program) [4]. The program required by the Allegheny College Institutional Review Board is Social & Behavioral Research. To get approved by the IRB, a proposal must be submitted. The proposal contains review questions to help the experiment designer and organizers to identify the type of the experiment because certain types of experiment can get exemption or expedition. The questions are, for example, does the experiment involve humans, does the experiment involve individuals under the age of 18. This experiment required an exempted review. Other than the review type, a brief introduction, procedure, and walk-through should be included.

Description and Rationale of Project

The description and rationale of the project included in the Institutional Review Board proposal is as follows:

This develops a novel software system to recommend songs to users. The mainstream music service providers utilize user browsing and listening history along with a comparison of the user's playlist to that of other users with similar habits to recommend music to users. In this project, instead of standard data, our system uses machine learning to identify the user's current emotion by asking a user to enter a text that contains some indication of their current emotions and feelings. The text entered by the user can be their diary, a transcribed conversation, or a script from a show that their current emotional state relates to. Using this text, our system then learns about the users' current emotional state and uses that information to predict desired musical themes and songs to the user. The purpose of the user study is to verify the efficiency of this software system, where the participants are asked to provide a short body of text, and following the review of the recommended song list, they are invited to provide feedback on the outcome they received.

Methods and Procedures

The methods and procedures of the experiment included in the Institutional Review Board proposal is as follows:

The participation for this study will be advertised on My Allegheny, and via departmental announcements (posters, club emails, Slack channel). The individuals who agree to participate in the study will be emailed a link to the website on which our system operates. Once on the landing page of the website, the participants will be provided information about the experiment and they will be informed that their consent is received if they proceed with the next steps of the study. In the next steps, the participants will create an account in our system and then log in to their unique account. The walk-through section contains a detailed sequence of steps that the participants will be involved in and the pages on our website that they will be directed to.

The website will be hosted on a private cloud-based server (Amazon Cloud Server, AWS). The IP addresses and other information that could associate with the participants will be deleted before analyzing the data. The data will be kept for four years for possible further study, and it will be deleted after four years. AWS's data centers and network architecture is designed to protect information, identities, and applications stored in their cloud by following core security and compliance requirements. The information required for the registration will be the username and password. After logging in, the participants will be asked to follow the instructions on the website and provide a body of text that reflects their emotions on three different days. The following instructions will be provided:

> Please choose three days of this week, and for each day:
>
> **Option 1:** Provide more than a paragraph of text in any form (e.g. diary) that best describes your emotions or experiences on that day.
>
> **Option 2:** Provide more than a paragraph of a conversational or summarization text (e.g. TV scripts or TV scene summarization) that describes an emotional scene.
>
> Please provide a descriptive title to each body of text that you enter.

After receiving the bodies of text from the user, our system will automatically provide a list of recommended songs. After getting the recommended music playlist, participants will be invited to participate in the survey by answering the following three questions:

1. Describe the body of text that you provided. What emotions did you want to express in this text?
2. Please choose the best description for the recommended playlist.
   (a) This playlist has nothing to do with my text
   (b) I can see a rare relation to my text
   (c) This playlist is acceptable, but I do not have a strong emotional connection to it
   (d) I have a fair amount of an emotional connection to this playlist
   (e) I have a strong emotional connection to this playlist
3. If you feel an emotional connection to this playlist, why do you think the music is related to your provided text? If you do not feel a connection to this playlist, what do you think the playlist is missing?

The goal of the questions in the survey above is to assess the efficiency of the algorithm and the participants' interpretation of their feelings. The answers will be collected on the same website of our system and the answers will be connected to specific user accounts. The user account is the only connection between the answers and the provided bodies of text by the user. Again, all the user information, including IP address, browser information, usernames, and passwords will be deleted before the analysis of the collected data.

After the survey, the data will be analyzed using the responses in user satisfaction from question 2. Responses in the range from 1 to 3 will be marked as "need for further analysis". Question 1 and question 3 will be used to interpret the unsatisfied playlist. The summary of the analyzed results will be reported in a senior thesis document that will be stored on the cloud-based version-control system, called GitHub. All participants who indicate an interest in receiving the results of the study will be emailed a link to the GitHub page containing the senior thesis document.

The potential risks of this study are that the participants may spend a long time writing texts. They may also have to think about sad memories. If the participants do not wish to proceed, they may quit any time during the process by pushing the *quit* button on the page.

Walk-Through

Participants would receive a link to begin their participation in this user study. When participants click on the link, they would land on the "Information" page.

Information Page After entering the website, users would see an information page that contains the following message:

Project introduction:

This project develops a novel software system to recommend songs to users. The mainstream music service providers utilize user browsing and listening history along with comparison of the user's playlist to that of other users with similar habits to recommend music to users. In this project, instead of standard data, our system uses machine learning to identify the user's current emotion by asking a user to enter a text that contains some indication of their current emotions and feelings. The text entered by the user can be their diary, a transcribed conversation, or a script from a show that their current emotional state relates to. Using this text, our system then learns about the users' current emotional state and uses that information to predict desired musical themes and songs to the user. The purpose of the user study is to verify the efficiency of this software system, where the participants are asked to provide a short body of text, and following the review of the recommended song list, participants are invited to provide feedback on the outcome they received.

Data usage:

This research is anonymous, the IP addresses and other personal information that could be associated with the participants will be deleted before analyzing the data. The data will be kept securely on Amazon Web Services for four years for possible further study, and they will be deleted after four years. The information required for the registration is the username and password. If you wish to receive a copy of the results of the study, please answer "yes" in the google survey form under the result disclosure section. A copy of the report of the study in the form of the senior thesis project document will be sent to you upon the completion of the analysis.

Exiting the study:

The potential risks of this study is that the participants may spend a long time writing texts. They may also have to think about sad memories. If you feel uncomfortable in any of the sections during the process, you can click the quit button. If you decide not to stop the participation, you can email the principal investigator of the project at *liux2@allegheny.edu* to request immediate deletion of any data you have created.

Compensation:

All participants will have the opportunity to join a drawing for a chance to win a $20 Amazon gift card. Please answer "yes" in the google survey form under the lottery section if you want your name to be entered into the drawing. All participants will receive a notification of the results of the drawing.

Contact information:

If you have any questions, please contact us.

Xingbang Liu:

Phone: (814)-795-0122

E-mail: liux2@allegheny.edu

Professor Janyl Jumadinova:

E-mail: jjumadinova@allegheny.edu

If you consent to the participation in this study, please continue with the next step.

After participants logged in, they would be taken to the "Text Entering" page.

Text Entering Page After participants entered their text according to the instruction, they would be directed to the "Result" page containing their recommended playlist, below which information about the survey will be included.

Result Page After listening to the songs recommended, users can proceed to the servey.

Survey 1: Please click on this link for the survey to provide feedback on your recommended playlist.

Survey 2: Please click on this link to indicate your interest in receiving the results of this study and your interest in entering the drawing.

4.2 Evaluation

4.2.1 Program Test Results

The program passed the test with only two warnings 4.6. The first warning does not affect the application, thus, it can be ignored. The second warning does not affect the application in version 3.7 of Python. Therefore, both of them can be ignored.

Listing 4.6: Pytest Terminal Output

```
==========================================================================
test session starts
==========================================================================
platform linux -- Python 3.7.6, pytest-5.4.1, py-1.8.1,
pluggy-0.13.1
plugins: cov-2.8.1
collected 16 items

test/test_lyrics_proc.py .Searching for "Country House" by
Blur...
Done.
.Searching for "On Top Of The World" by Imagine Dragons...
Done.
.Searching for "Scared To Live" by The Weeknd...
```

```
Done.
.......
test/test_text_proc.py ......

================================================================
warnings summary
================================================================
FSADeprecationWarning: SQLALCHEMY_TRACK_MODIFICATIONS adds
significant overhead and will be disabled by default in the
future. Set it to True or False to suppress this warning.
    'SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead
    and '

DeprecationWarning: Using or importing the ABCs from
'collections' instead of from 'collections.abc' is deprecated
since Python 3.3, and in 3.9 it will stop working
    if (isinstance(value, str) or isinstance(value,
    collections.Callable) or hasattr(value, 'match')

-- Docs: https://docs.pytest.org/en/latest/warnings.html
================================================================
16 passed, 2 warnings in 11.27s
================================================================
```

4.2.2 User Test Results

Case Study

In this section, a text from user was submitted and the result was analyzed manually.
The first text describes a sad love scene from a TV drama.

> Shen Wei and Zhao Yunlan are characters from drama Zhenhun. In the drama, Shen wei is one of the most powerful man charging the hell and he masks as a college professor to live on earth. Shen Wei loves Zhao Yunlan. The reason he works as a college professor on earth is to look for Zhao.He had met Zhao 10000 years ago when he was young but Zhao disappeared and left a promise that they would meet again.
>
> After Shen finnally found Zhao, Zhao had no idea who Shen is, but they stil fall in love. As story goes on, they found that the Zhao 10000 years ago was the Zhao in the present travled through time. Anyway, Shen waited him for 10000 years.
>
> In the end of the story, they were trying to save the world and Zhao was endanger. Shen sacrificed himself to confront the boss. However, after Shen's scrifice, Zhao also used his life to light up the world.

They saw each other before their sprits disappear. They made another promise, which is to meet each other again sometime in the future, no matter how long it takes.

The scene was so visual impact due to the superb acting of two actors. Shen always stares Zhao with overloaded emotion. The scene of them saying good bye was very impressive when Shen finally can't hold himself crying.

In a few words, this text described the love story between two characters which entangled for thousands of years. However, every time they meet with each other, they have to separate for different reasons. After submitting the text to the system, the result shows:

Recommended Playlist:

Too Good At Goodbyes by **Sam Smith**

Bad Liar by **Imagine Dragons**

When I Was Your Man by **Bruno Mars**

7 rings by **Ariana Grande**

One Match Fire by **Andrew Paley**

The key emotions in these songs can be summarized as lovers separate too easy; a guy wants his lover to believe that everything he said was a lie, so that the girl is free to leave; a guy regrets that he did not treat his lover better; a girl encourages herself when she is sad; a guy encouraging people to have faith. Form the rough interpretation, 3/5 of the recommended songs are about sad love. 2/5 of the songs are about having faith.

The result of the survey is not significant since only 3 people participated in the survey. The graph 4.1 shows that all people participated thought they had a fair amount of an emotional connection to their playlist.

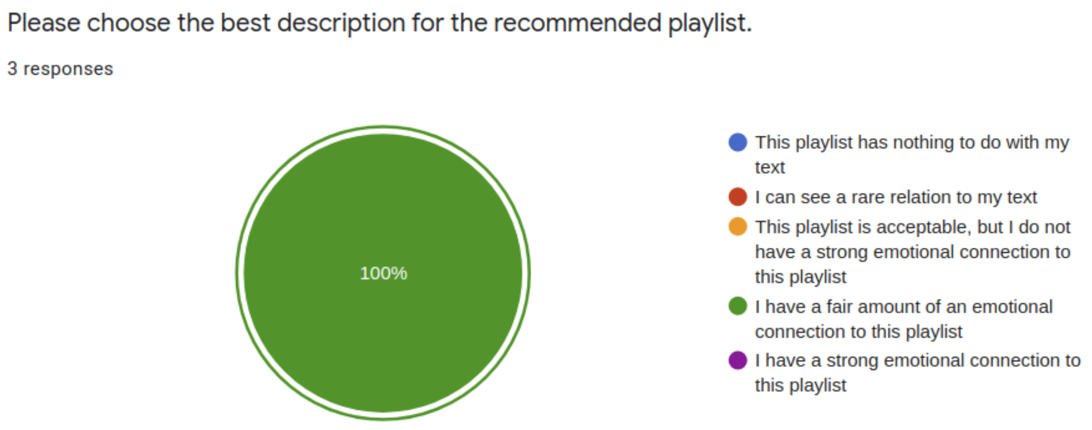


Figure 4.1: Participants Conclusion

4.3 Threats to Validity

There are multiple factors that can affect the validity of this experiment. First of all, the number of people who participated and answered the survey in this experiment was not significant. Therefore, the interpretation of the responses could not show any trends. Secondly, the lack of a web player could make participants lose the emotional connection between their text and songs. Finally, there is a big flaw in the experiment design.

The experiment contains three variables that are hard to control. It would be better if the experiment has only one variable. The current experiment design requires participants to generate a text that contains emotions. After the recommendation, participants have to assess their text, song emotion, and associations. Therefore, the variables that the experiment contains are text emotion, song emotion, and the perception of participants' association between text and song. Only limiting the scene, so that the variables can be limited as one, can the accuracy of the result be increased.

Chapter 5

Discussion and Future Work

5.1 Summary of Results

The proposed project is to build a system that can match descriptive paragraphs with music tracks. By using this system, users do not have to explore music without a target, and they can find music based on their current emotions.

System Summary

The system built takes user text and matches the text with the lyrics of the music tracks. The top results will be recommended to users.

The web interface allows participants to read the instructions and their rights. Participants can also register a unique account and log into the system. After logging in to the system, participants can read the requirements and submit their descriptive text. Finally, users can get the recommended playlist, the links to the survey, and participation for the compensation lottery from the web interface. The web interface was visualized by the front-end. The front-end of this system was built by using Flask and HTML templates. The back-end consists of a database, Flask routes, lyrics model, and text model. The back-end allows the data of the website to be processed.

Experiment Summary

The designed experiment asks participants to use the system, and finish a survey to assess their experiences each time they submit a text to the system. The proposed survey has three questions for the participants:

1. Describe the body of text that you provided. What emotions did you want to express in this text?
2. Please choose the best description for the recommended playlist.
   (a) This playlist has nothing to do with my text
   (b) I can see a rare relation to my text
   (c) This playlist is acceptable, but I do not have a strong emotional connection to it

   (d) I have a fair amount of an emotional connection to this playlist
   (e) I have a strong emotional connection to this playlist

3. If you feel an emotional connection to this playlist, why do you think the music is related to your provided text? If you do not feel a connection to this playlist, what do you think the playlist is missing?

The purpose of the three questions is to assess the emotion that participants expressed in the text, their satisfaction for the recommended playlist, and their self-assessment on the associations between the text and music tracks.

The survey of user study was collected with Google Forms. Due to the complexity and long time required for this experiment, only half of the surveys were collected. People would drop the experiment after they submitted the text. Therefore, the survey result is not significant. The efficiency of the system can not be assessed.

Although the number of responses collected was not significant, based on the submitted responses, most of the people explained that they can not listen to the music right after getting the result. Since this is a minimum viable product, the music player was not implemented. However, the participants wanted to relate themselves with the music right after the text submission, so that they can be related to the emotions. There were also responses that the emotion they provided are simple, thus, the result should be easy to get. The responses show a blind trust and self-justification of the recommended playlist. The reason could be that participants did not get the accurate meaning of the music track. According to the responses, 100% of the participants thought that they had a fair amount of an emotional connection to this playlist.

5.2 Future Work

Many aspects of this project can be improved.

Web Player

First of all, according to the result of the survey, one participant thought it would be better if the music could be played right after the text submission. That way, participants can have a better connection between text and songs. All the songs were collected from Spotify playlists, it would be convenient if a Spotify Web player can be implemented.

Flask Project Structure and Front-End

As for the website, the code can be modulized. For now, all the features are all in one directory, it would add extra work if the website needs more functions. It would also be good if a certain style and theme can be added to the front-end. A designed website not only makes reading easier but also has the ability to lead the user based on visual cue according to Fitts's law in the area of human–computer interaction.

The Lyrics Model

The lyrics model in the project does not have the ability to determine duplicated entries. Therefore, to add new playlists, the old data table has to be deleted. To improve that, a function to determine the new entry would be helpful to decrease the data upload time and to automate the playlist update. It would also be helpful to add language support to other languages. This would increase the number of songs and the amount of non-English speaking users.

The Text Model

The summarization and text matching algorithms can also be improved. For the summarization algorithm, it will rank the key phrases. However, the lyrics were not preprocessed. Therefore, the lyrics contain duplicated sentences. In this case, only the duplicated sentences would be kept because the algorithm thinks the repeated sentences are important. To improve the summarization algorithm, besides text ranking, other features can be learned and added to the summarization. As for the matching algorithm, the Monge-Elkan algorithm was not evaluated in terms of two hybrid string matching algorithms. By evaluating the different combinations of the edit-based algorithm and token-based algorithm, the Monge-Elkan algorithm can be optimized.

The Experiment

Finally, the experiment design can be improved by decreasing the number of variables. For example, by letting participants read a given text, then listen to the recommended songs, can the perception of association be tested. By letting participants listen to certain songs and describe their possible emotions, the song's emotion can be tested. Besides the controlling of variables, the scale of emotions can also be optimized. The emotions are hard to scale, for example, the sadness of a person can be different in different time slots because of environmental cues or other reasons. It would be helpful to scale the emotions of a person so that the emotion detection accuracy can be increased.

5.3 Conclusion

In this project, a system was implemented to match the descriptive text with the music tracks. The descriptive text contains the emotions of the participants, and it was summarized with a Python implemented text ranking system. The lyrics were also summarized with the same Python implemented text ranking system, and the two summarized texts were matched by using the Monge-Elkan similarity algorithm. The implemented system can help users to find music tracks by detecting their current emotions. This way, users do not have to broadly search for music by emotional tags such as sad and happy.

The result of the experiment designed to test the efficiency of the system was not significant due to the number of responses received. Therefore, the system could not be evaluated efficiently. The experiment design was also not optimized because the variables were not limited. Thus, a better-designed experiment and a larger amount of participants are required to check if the implemented system is efficient.

In the future, a web player should be implemented for an instant association between participants submitted text and recommended playlist. The structure of the project and the web interface should be optimized for the convenience of developer implementation and participants' browsing. The lyrics model should be optimized by having the ability to update the lyrics database in an efficient manner. The text model should be optimized, so that more important details are kept, and the matching of two texts are more accurate. Finally, the experiment should be redesigned, so that the variables are controlled.

Bibliography

[1] About genius. https://genius.com/Genius-about-genius-annotated, last accessed on 04/29/20.

[2] Adjusted Cosine Similarity. http://www10.org/cdrom/papers/519/node14.html, last accessed on 04/29/20.

[3] Convert your music playlists. https://www.playlist-converter.net/, last accessed on 04/29/20.

[4] Mission and history. https://about.citiprogram.org/en/mission-and-history/, last accessed on 04/29/20.

[5] 'n'-dimensional euclidean distance. https://hlab.stanford.edu/brian/euclidean_distance_in.html, last accessed on 04/29/20.

[6] IFPI releases 2018 Music Consumer Insight Report, Oct 2018. https://www.ifpi.org/news/IFPI-releases-2018-music-consumer-insight-report, last accessed on 04/29/20.

[7] Metadata in digital audio files – what it is, where it is and how to tidy it up., Jan 2018. https://www.cambridgeaudio.com/usa/en/blog/metadata-digital-audio-files-\T1\textendash-what-it-where-it-and-how-tidy-it, last accessed on 04/29/20.

[8] Institutional review board, Sep 2019. https://sites.allegheny.edu/committees/institutional-review-board/, last accessed on 04/29/20.

[9] abhijit2592. A fully customisable language detection pipeline for spaCy, May 2019. https://github.com/Abhijit-2592/spacy-langdetect, last accessed on 04/29/20.

[10] Bogdanov, D., Haro, M., Fuhrmann, F., Xambó, A., Gómez, E., and Herrera, P. Semantic audio content-based music recommendation and visualization based on user preference examples. *Information Processing & Management 49*, 1 (2013), 13–33.

[11] Brilis, S., Gkatzou, E., Koursoumis, A., Talvis, K., Kermanidis, K. L., and Karydis, I. Mood classification using lyrics and audio: A case-study in Greek music. In *IFIP Advances in Information and Communication Technology* (2012).

[12] Bu, J., Tan, S., Chen, C., Wang, C., Wu, H., Zhang, L., and He, X. Music Recommendation by Unified Hypergraph: Combining Social Media Information and Music Content. In *Proceedings of the 18th ACM international conference on Multimedia* (2010).

[13] Celma, Ò. *Music Recommendation and Discovery.* 2010.

[14] ceteri. Python impl for TextRank, June 2017. `https://github.com/DerwenAI/pytextrank`, last accessed on 04/29/20.

[15] Craw, S. Manhattan distance, Jan 1970. `https://link.springer.com/referenceworkentry/10.1007/978-0-387-30164-8_506`, last accessed on 04/29/20.

[16] Han, B. J., Rho, S., Jun, S., and Hwang, E. Music emotion classification and context-based music recommendation. *Multimedia Tools and Applications* (2010).

[17] johnwmillr. LyricsGenius: a Python client for the Genius.com API, Dec. 2019. `https://github.com/johnwmillr/LyricsGenius`, last accessed on 04/29/20.

[18] Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759* (2016).

[19] Kamenetsky, S. B., Hill, D. S., and Trehub, S. E. Effect of tempo and dynamics on the perception of emotion in music. *Psychology of Music 25*, 2 (1997), 149–160.

[20] Kaminskas, M., and Ricci, F. Contextual music information retrieval and recommendation: State of the art and challenges, 2012.

[21] Liikkanen, L. A., and Salovaara, A. Music on YouTube: User engagement with traditional, user-appropriated and derivative videos. *Computers in Human Behavior* (2015).

[22] MIDiA-Research. About Us, 2019. `https://www.midiaresearch.com/about-us/`, last accessed on 04/29/20.

[23] Mihalcea, R., and Tarau, P. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing* (2004).

[24] Mulligan, M. Mid-Year 2018 Streaming Market Shares, Sep 2018. `https://www.midiaresearch.com/blog/mid-year-2018-streaming-market-shares/`, last accessed on 04/29/20.

[25] Palaniappan, V. Spotify Product Analysis, Jan 2018. `https://medium.com/@vigneshp_/spotify-product-analysis-e76a59b5591`, last accessed on 04/29/20.

[26] pallets. Adds SQLAlchemy support to your Flask application, Sept. 2019. `https://github.com/pallets/flask-sqlalchemy`, last accessed on 04/29/20.

[27] PALLETS. The Python micro framework for building web applications, Apr. 2020. `https://github.com/pallets/flask`, last accessed on 04/29/20.

[28] PRZYBYSZ, P. Music and emotions. *Avant* (2013).

[29] PYENV. Simple Python Version Management: pyenv, Jan. 2019. `https://github.com/pyenv/pyenv`, last accessed on 04/29/20.

[30] PYPA. Pipenv: Python Development Workflow for Humans, Nov. 2018. `https://github.com/pypa/pipenv`, last accessed on 04/29/20.

[31] PYTEST DEV. The pytest framework makes it easy to write small tests, yet scales to support complex functional testing, Mar. 2020. `https://github.com/pytest-dev/pytest`, last accessed on 04/29/20.

[32] RASCHKA, S. Naive bayes and text classification I - introduction and theory. *CoRR abs/1410.5329* (2014).

[33] RICCI, F., ROKACH, L., AND SHAPIRA, B. Introduction to recommender systems handbook. In *Recommender systems handbook*. Springer, 2011, pp. 1–35.

[34] RUSSELL, J. China's NetEase raises $600M for its music streaming business, Nov 2018. `https://techcrunch.com/2018/11/13/netease-cloud-music-raises-600-million/`, last accessed on 04/29/20.

[35] STEPHANIE. Mahalanobis distance: Simple definition, examples, Aug 2018. `https://www.statisticshowto.datasciencecentral.com/mahalanobis-distance/`, last accessed on 04/29/20.

[36] SU, X., AND KHOSHGOFTAAR, T. M. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence* (2009).

[37] TEAM, W. M. A comprehensive and scalable set of string tokenizers and similarity measures in Python, Feb. 2019. `https://github.com/anhaidgroup/py_stringmatching`, last accessed on 04/29/20.

[38] THOMPSON, W. F., AND ROBITAILLE, B. Can composers express emotions through music? *Empirical studies of the arts 10*, 1 (1992), 79–89.

[39] ZHANG, B., KREITZ, G., ISAKSSON, M., UBILLOS, J., URDANETA, G., POUWELSE, J. A., AND EPEMA, D. Understanding user behavior in Spotify. In *Proceedings - IEEE INFOCOM* (2013).