

MRUSA - SENTIMENT-DRIVEN MUSIC RECOMMENDER

PROJECT REPORT

Submitted by

Udita Kausakhi U (RA2111047010102)
Tarush Kumar Goyal (RA2111047010110)
Charvi Jain (RA2111047010113)
Karan Raghavan (RA2111047010119)
Mohnish Ramachandran (RA2111047010121)

Under the Guidance of

Dr. Poongothai E.

Assistant Professor, Department of Computational Intelligence

In partial satisfaction of the requirements for the degree of

**BACHELOR OF TECHNOLOGY
in
ARTIFICIAL INTELLIGENCE**



SCHOOL OF COMPUTING

**COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR - 603203**

OCTOBER 2023



COLLEGE OF ENGINEERING & TECHNOLOGY SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
S.R.M. NAGAR, KATTANKULATHUR – 603 203
Chengalpattu District

BONAFIDE CERTIFICATE

Certified that this mini project report “**MRUSA - Sentiment-Driven Music Recommender**” is the bonafide work of “**Udita Kausakhi U (RA2111047010102), Tarush Kumar Goyal (RA2111047010110), Charvi Jain (RA2111047010113), Karan Raghavan (RA2111047010119), Mohnish Ramachandran (RA2111047010121)**” who carried out the project work for **18AIC301J – DEEP LEARNING TECHNIQUES** under my supervision at **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**, Kattankulathur during the academic year 2023 – 2024.

SIGNATURE

Faculty In-Charge

Dr. Poongothai E.

Assistant Professor

Department of Computational Intelligence

SRM Institute of Science and Technology

SIGNATURE

HEAD OF THE DEPARTMENT

Dr. Annie Uthra

Professor and Head,

Department of Computational Intelligence

SRM Institute of Science and Technology

TABLE OF CONTENTS

S. NO	TITLE	PAGE NO
1	ABSTRACT	1
2	INTRODUCTION	2
3	OBJECTIVE	3
4	ARCHITECTURE	4
5	MODULES	5
6	INPUT/OUTPUT	6
7	SOCIETAL BENEFITS	12
8	REFERENCES	13

ABSTRACT

A dual-part deep learning project is presented, combining sentiment analysis and a music recommendation system. The sentiment analysis model, built using two Kaggle datasets, accurately predicts emotions in text, and the accompanying Python application provides users with numeric outputs corresponding to the detected emotions. The recommendation system, powered by the Spotify Web API, delivers personalized music suggestions through a CLI application. The project showcases the potential for enhanced human-computer interaction and personalized music experiences, offering users the opportunity to explore their emotions and discover music tailored to their preferences. The GitHub repository (https://github.com/Mohnish4533/DL_Project) contains detailed resources for further exploration and development of this innovative project.

INTRODUCTION

In a world where technology continues to reshape how we interact with information and entertainment, this project represents an innovative convergence of sentiment analysis and music recommendation. We enter a realm where deep learning algorithms decipher the emotions embedded in text and subsequently guide users to discover the perfect soundtrack for their state of mind. This dual-part endeavor not only illuminates the capabilities of artificial intelligence but also exemplifies how it can elevate our human experiences. As we embark on this journey, we venture into the intriguing intersection of emotion, language, and music, offering a unique and transformative perspective on how AI can redefine the way we connect with technology.

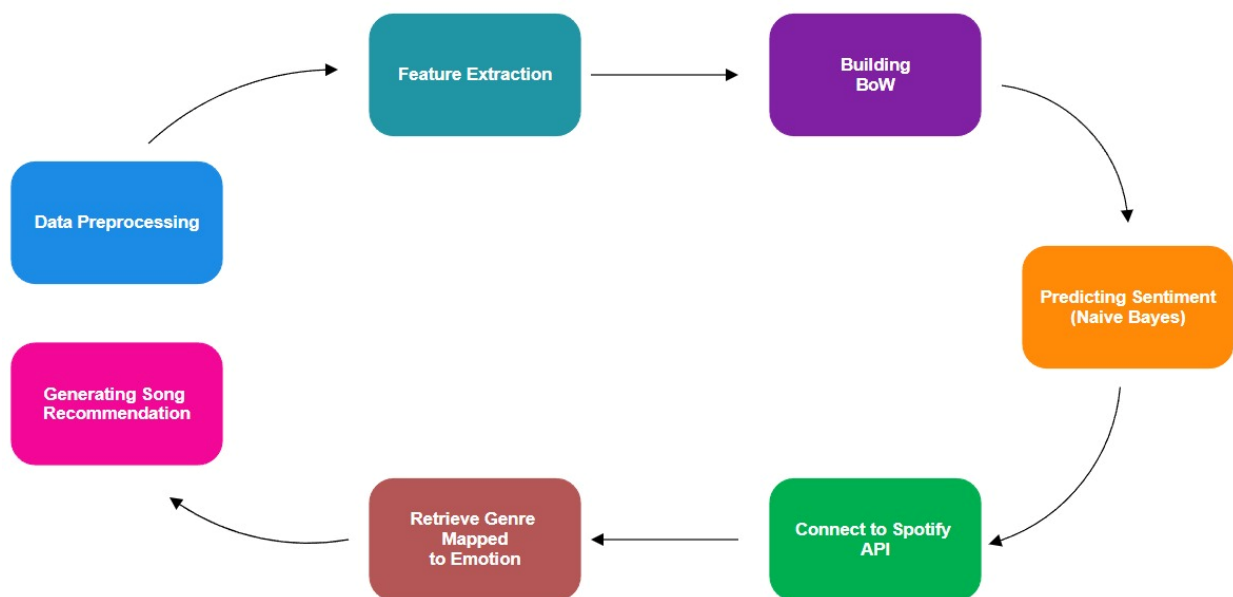
OBJECTIVE

The primary objective of this dual-part deep learning project is to create a seamless and engaging user experience by combining sentiment analysis and music recommendation. Specifically, the project aims to achieve the following:

- **Sentiment Analysis:** Develop an accurate and efficient sentiment analysis model that can comprehend and categorize emotions expressed in text, providing users with valuable insights into the emotional content of their input.
- **Emotion Mapping:** Establish a mapping system that categorizes emotions into main categories (sadness, joy, love, anger, fear) and their corresponding sub-emotions, enabling a more nuanced understanding of human emotional states.
- **User Interaction:** Create a user-friendly Python application that allows users to interact with the sentiment analysis model, receiving numeric outputs that correspond to detected emotions. This application aims to enhance human-computer interaction and foster more empathetic digital communication.
- **Music Recommendation:** Develop a music recommendation system that leverages the Spotify Web API and OAuth 2.0 client credentials flow to provide users with personalized music suggestions based on their preferences and moods.
- **Integration:** Seamlessly integrate the sentiment analysis and music recommendation components to offer users a holistic experience, where their detected emotions influence the music recommendations, enhancing personalization and engagement.
- **Enhanced User Experience:** Improve the quality of human-computer interaction by allowing users to explore their emotions and discover music that resonates with their feelings, resulting in a more enjoyable and meaningful digital experience.

ARCHITECTURE

The system architecture of this project seamlessly combines sentiment analysis and music recommendation. It begins with the user input, typically a concise text describing their emotions. Each word in the input is converted into a numeric value based on an existing vocabulary, creating a Bag of Words (BoW) representation. This BoW list is then used to predict the sentiment or emotion using a Naive Bayes model. The predicted emotion is mapped to a relevant music genre. The Spotify API is employed to search for music within the identified genre, and a tailored list of music recommendations is generated for the user. This integrated architecture provides a holistic user experience by bridging sentiment analysis with personalized music discovery.



MODULES

PART 1: Sentiment Analysis Module

- **Data Collection:** This module uses two Kaggle datasets to collect text data for sentiment analysis. The data includes text samples with associated emotion labels.
- **Data Preprocessing:** This module involves tasks such as text cleaning, tokenization, feature extraction, and preparing the text data for sentiment analysis.
- **Sentiment Analysis Model:** The core component that includes building, training, and fine-tuning the deep learning model to classify text into different emotions and sub-emotions.
- **Emotion Mapping:** This module categorizes the emotions detected into main emotions (sadness, joy, love, anger, fear) and their associated sub-emotions, providing a more comprehensive emotional analysis.

PART 2: Music Recommendation Module

- **Spotify API Integration:** This module connects to the Spotify Web API using OAuth 2.0 client credentials, allowing the system to access and retrieve music data.
- **User Preferences:** Capturing and understanding the user's music preferences and moods, which are used to inform music recommendations.
- **Recommendation Algorithm:** Developing an algorithm that suggests tracks, artists, or playlists based on the user's preferences and emotional state.
- **CLI Application:** Creating a command-line interface (CLI) application that enables users to interact with the music recommendation system and receive personalized music suggestions.

PART 3: Integration Module

- **Seamless Integration:** This module connects the sentiment analysis model's output with the music recommendation system to ensure that the user's detected emotions influence the music recommendations, providing a unified and tailored user experience.

INPUT/OUTPUT SCREENSHOTS

PART 1: Sentiment Analysis Module

Data Preprocessing:

```
In [23]: print(df2['sentiment'].unique())
print(df2['sentiment'].value_counts().unique())

['empty' 'sadness' 'enthusiasm' 'neutral' 'worry' 'surprise' 'love' 'fun'
'hate' 'happiness' 'boredom' 'relief' 'anger']
[8638 8459 5209 5165 3842 2187 1776 1526 1323 827 759 179 110]
```

Sentiment Analysis Model:

- **Creating Model:**

```
In [39]: # Create Bag Of Words Model
vect = CountVectorizer()
BoW = vect.fit_transform(df['text'])
BoW_list = BoW.toarray()
```

```
In [40]: #vect.vocabulary_
```

```
In [41]: # length of vocabulary of BoW
len(vect.vocabulary_)
```

```
Out[41]: 40334
```

```
In [42]: # preparing column name for dataframe
col_name = []
for i in range(len(vect.vocabulary_)):
    c = "x" + str(i)
    col_name.append(c)

print(col_name)
```

```
In [44]: # split into train and test dataset
X_train, X_test, y_train, y_test = train_test_split(X, Y, train_size= 0.8, random_state= 42)
```

```
In [45]: print("X train shape : {} \nX test shape: {} \nY train shape: {} \nY test shape: {} \n".format(X_train.shape, X_test.shape, y_train.shape, y_test.shape))

X train shape : (47340, 40334)
X test shape: (11836, 40334)
Y train shape: (47340,)
Y test shape: (11836,)
```

```
In [48]: # Creating Sentiment predicting model
model1 = MultinomialNB()
model1.fit(X_train, y_train)
```

```
Out[48]: MultinomialNB()
```

- **Model Testing:**

```
In [49]: # testing model
y_pred = model1.predict(X_test)

print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.57	0.56	0.57	2751
1	0.55	0.84	0.67	4563
2	0.46	0.18	0.26	1055
3	0.68	0.22	0.33	1300
4	0.47	0.30	0.36	2167
accuracy			0.55	11836
macro avg	0.55	0.42	0.44	11836
weighted avg	0.55	0.55	0.51	11836

- **Prediction Using Custom Input**

```
In [ ]: # ERROR ----- NOT ABLE TO USE CUSTOM INPUT (CHECK) (FORGET ABOUT THIS FOR NOW)
text_input = input("Enter a text about how u feel: ")

# create BoW for the input text
t_Bow = vect.transform([text_input])
t_Bow_array = Bow.toarray()

# predict sentiment
print(model1.predict(t_Bow_array))
```

- **Exporting Model:**

```
In [53]: import pickle
```

```
In [54]: file = open("./BoW_Vocab.txt", 'wb')
saved_vocab = pickle.dump(vect.vocabulary_, file)
file.close()
```


```
In [55]: file = open("./Sentiment_Prediction_Model.txt", "wb")
saved_model = pickle.dump(model1, file)
file.close()
```

Emotion Mapping:

Code

Blame

11 lines (10 loc) · 269 Bytes

 Code 55% faster with GitHub Copilot

```
1      # main app only works for 1 genre at a time (one-to-one mapping)
2  def get_mappings():
3      mapping = {
4          '0' : ['classical'],
5          '1' : ['pop'],
6          '2' : ['romance'],
7          '3' : ['metal'],
8          '4' : ['thriller']
9      }
10
11      return mapping
```

PART 2: Music Recommendation Module

Recommendation Algorithm:

```
# import required dependencies
# make sure these are present in the main application as well
import os
import pickle
from sklearn.feature_extraction.text import CountVectorizer

def Get_Sentiment():
    # load Bag Of Words Vocabulary
    BoW_Vocab_path = "./requirements/BoW_Vocab.txt" # change path if required

    if os.path.getsize(BoW_Vocab_path) > 0:
        with open(BoW_Vocab_path, 'rb') as vocab_file:
            unpickler = pickle.Unpickler(vocab_file)
            vocab = unpickler.load()
    else:
        print("Empty")
        return 11

    # load the sentiment prediction model
    Prediction_Model_path = "./requirements/Sentiment_Prediction_Model.txt" # change path if required

    if os.path.getsize(Prediction_Model_path) > 0:
        with open(Prediction_Model_path, 'rb') as model_binaryfile:
            unpickler = pickle.Unpickler(model_binaryfile)
            model = unpickler.load()
    else:
        print("Empty")
        return 22

    vocab_size = len(vocab)
    #print(vocab_size)

    # create a CountVectorizer Object
    vect= CountVectorizer(max_features=vocab_size)
    vect.vocabulary_ = vocab

    # get user input
    text_input = input("Enter a text about how u feel: ")

    # create BoW for the input text
    BoW = vect.transform([text_input])
    BoW_array = BoW.toarray()

    # predict sentiment
    senti = model.predict(BoW_array)[0]

    return senti
```

CLI Application:

```
1  # dependencies for App
2  from dotenv import load_dotenv
3  import os
4  import base64
5  from requests import post, get
6  import json
7
8  # dependencies for sentiment prediction
9  # import os (repeat so commenting out)
10 import pickle
11 from sklearn.feature_extraction.text import CountVectorizer
12 import Model_Application      # importing sentiment predictor
13
14 # other dependencies
15 import genre_mapping
16
17 # load client environment info
18 path = "./requirements/.env"
19 load_dotenv(path)
20
21 # get client info
22 client_id = os.getenv("CLIENT_ID")
23 client_secret = os.getenv("CLIENT_SECRET")
24
25 # generating token to access api
26 ✓ def get_token():
27     auth_string = client_id + ":" + client_secret
28     auth_bytes = auth_string.encode("utf-8")
29     auth_base64 = str(base64.b64encode(auth_bytes), "utf-8")
30
31     url = "https://accounts.spotify.com/api/token"
32     headers = {
33         "Authorization": "Basic " + auth_base64,
34         "Content-Type": "application/x-www-form-urlencoded"
35     }
36
37     data = {"grant_type": "client_credentials"}
38     result = post(url, headers= headers, data= data)
39
40     json_result = json.loads(result.content)
41     token = json_result['access_token']
42
43     return token
44
```

```

45     # get headers
46     def get_auth_header(token):
47         return {"Authorization": "Bearer " + token}
48
49     # search query
50     def get_genre_based_tracks(token, g):
51         url = "https://api.spotify.com/v1/recommendations/"
52         headers = get_auth_header(token)
53         query = f"?seed_genres={g[0]}&limit=10"
54
55         query_url = url + query
56
57         result = get(query_url, headers=headers)
58
59         json_result = json.loads(result.content)['tracks']
60
61         return json_result
62

```

```

64     # MAIN CODE
65
66     #get token to run spotify api
67     token = get_token()
68
69     # run sentiment predictor
70     emotion = Model_Application.Get_Sentiment()
71
72     # handle errors
73     if emotion == 11:
74         print("ERROR: Bag Of Words Vocabulary error. Please get the right vocabulary file and save it as \"BoW_Vocab.txt\"....")
75     elif emotion == 22:
76         print("ERROR: No Sentiment analysis model exists. Please have a file \"Sentiment_Prediction_Model.txt\" with the exported model....")
77     else:
78         # get mapping of emotion-to-genre
79         mapping = genre_mapping.get_mappings()
80
81         required_genre = mapping[f"{emotion}"]
82
83         result = get_genre_based_tracks(token, required_genre)
84         if len(result) == 0:
85             print("No song recommendations could be found.")
86         else:
87             # show output in readable format
88             for idx, song in enumerate(result):
89                 print(f"{idx + 1}. {song['name']}")

```

PART 3: Integration Module

Seamless Integration (Emotion to Numeric):

- (0) -- sadness, boredom, surprise:

```

Enter a text about how u feel: A close relative of mine died recently and its heartbreaking
0
['classical']
1. Main Title/John Williams/Jaws - From The "Jaws" Soundtrack
2. Passion Week: Of Your Mystical Supper
3. Songs My Mother Taught Me (from "Gypsy Songs", Op. 55, No. 4)
4. Fugment No. 2, "Remembering Mr. Ives"
5. Symphony No.2 in C minor - "Resurrection": 5d. "Aufersteh'n, ja aufersteh'n wirst du" (Langsam. Misterioso) - "Auferstehung"
6. Where Is My Mind
7. Vaughan Williams: The Lark Ascending: Opening (Excerpt)
8. Klarinettenkonzert Nr. 1 f-Moll: II. Adagio ma non troppo
9. Sonata for Althorn and Piano: IV. Das Posthorn (Zwiegespräch) - Lebhaft
10. Shawshank Prison - Stoic Theme

```

- (1) -- joy, enthusiasm, happiness, fun, neutral:

```
Enter a text about how u feel: I finally won a competition. My first ever competition!
1
['indian']
1. Allah Jaane (From "Teri Meri Kahaani")
2. Satisfya
3. Aaja Nachle [Come on dance!]
4. Abhi Na Jao Chhod Kar
5. Mera Yaar Dildar
6. Rangeela Re (From "Rangeela")
7. Jag Ja
8. Bahon Mein Chale Aao
9. Dum Maro Dum Mit Jaye Gham
10. Mundian to Bach Ke - Beware Of The Boys - Jay-Z Remix
```

- (2) – love:

```
Enter a text about how u feel: I fall more in love with her every day. I love love her
2
['romance']
1. She Doesn't Mind
2. Come On Get Higher
3. Fall For You
4. Irreplaceable
5. Lovebug
6. Knock You Down
7. Little Talks
8. Feel This Moment (feat. Christina Aguilera)
9. Sparks Fly
10. A Thousand Miles
```

- (3) -- hate, anger, empty, relief:

```
Enter a text about how u feel: I hated the event
3
['chill']
1. Happy Home
2. Someone Like You
3. It Girl
4. Through Glass
5. Rocket Man (I Think It's Going To Be A Long Long Time)
6. No One
7. Just A Dream - Main
8. Upside Down
9. White Winter Hymnal
10. Come On Get Higher
```

- (4) -- fear, worry:

```
Enter a text about how u feel: im worried cause my results are going to be released
4
['comedy']
1. The Salt and Pepper Diner
2. Trick Or Treating
3. Walking With Dinosaurs
4. Stupid In School - Live
5. Over 40 and Dating
6. Open Water
7. I Miss Michael Jackson
8. Women's Movement
9. World Tour
10. Ireland
```

SOCIETAL BENEFITS

- **Mood Upliftment:** Recommending upbeat and happy tunes through sentiment analysis can improve users' moods and foster a positive atmosphere.
- **Personalized Music Enjoyment:** Music recommendation systems help individuals discover songs that align with their emotional state, leading to a more enjoyable listening experience
- **Stress Reduction:** Sentiment analysis can recommend calming or soothing music to reduce stress and anxiety, contributing to overall mental well-being
- **Enhanced Productivity:** Music recommendations based on sentiment analysis can boost productivity by providing music that matches the user's work or study mindset.
- **Improved Emotional Health:** Music can be used as a therapeutic tool, with sentiment-based recommendations helping individuals manage and enhance their emotional health.
- **Cultural Exploration:** Sentiment analysis-driven recommendations introduce users to music from different cultures and languages, fostering cross-cultural understanding and appreciation.
- **Social Connection:** Music recommendations can encourage social interactions and bonding among individuals who share similar musical tastes.
- **Cognitive Development:** Exposure to a diverse range of music through sentiment-based recommendations can stimulate cognitive development, especially in children.
- **Music Discovery:** Users can explore new genres and artists they might not have encountered otherwise, broadening their musical horizons.
- **Support for Independent Artists:** Sentiment-based recommendations can help independent or lesser-known artists gain recognition and support, contributing to a vibrant and diverse music ecosystem..

REFERENCES

- [1] SentiSpotMusic: a music recommendation system based on sentiment analysis. (2022, February 11). IEEE Conference Publication | IEEE Xplore.
<https://ieeexplore.ieee.org/document/9781862>
- [2] Radhika, N. (2017). Music Recommendation System Based on User's Sentiment.
<https://www.semanticscholar.org/paper/Music-Recommendation-System-Based-on-User%E2%80%99s-Radhika/a8d623d60b99118eab3f73f8a1a0d70180944517>
- [3] Music recommendation system based on user's sentiments extracted from social networks. (2015, August 1). IEEE Journals & Magazine | IEEE Xplore.
<https://ieeexplore.ieee.org/document/7298296>
- [4] Hossain, A. B., Hasan, W. U., Zaman, K. T., & Howlader, K. (2023, January 1). Integrated Music Recommendation System Using Collaborative and Content Based Filtering, and Sentiment Analysis. https://doi.org/10.1007/978-3-031-34622-4_13
- [5] Rosa, R. L., Rodríguez, D. Z., & Bressan, G. (2015, August 1). Music recommendation system based on user's sentiments extracted from social networks. IEEE Transactions on Consumer Electronics; Institute of Electrical and Electronics Engineers.
<https://doi.org/10.1109/tce.2015.7298296>