

MASTER OF TECHNOLOGY PROJECT REPORT

YOUR SONG Intelligent Music Recommender Robot

Team members

Lin Qianqian

Liu Jing

Yang Mingrun

Zhu Junkun

CONTENTS

1 EXECUTIVE SUMMARY.....	
2 PROBLEM DESCRIPTION.....	
2.1 PROJECT OBJECTIVE.....	
3 KNOWLEDGE MODELING.....	
3.1 GENERAL DESCRIPTION.....	
3.2 DIALOGFLOW CHATBOT.....	
3.2.1 KNOWLEDGE IDENTIFICATION.....	
3.2.2 KNOWLEDGE EXTRACTION.....	
3.2.3 KNOWLEDGE DEFINITION.....	
3.3 COLLABORATIVE FILTERING RECOMMENDATION.....	
3.4 INFORMED SEARCH TREE.....	
3.4.1 KNOWLEDGE REFINEMENT.....	
4 SOLUTION OUTLINE	
4.1 SYSTEM ARCHITECTURE.....	
4.2 PROJECT SCOPE	
4.3 SYSTEM'S FEATURES	
4.4 FUTURE IMPROVEMENTS.....	
Appendix 1: Project Proposal.....	
Appendix 2: Mapped System Functionalities against the knowledge.....	
Appendix 3: Installation and User Guide.....	
Appendix 4: Individual Report – Lin Qianqian.....	
Appendix 5: Individual Report – Liu Jing.....	
Appendix 6: Individual Report – Yang Mingrun.....	
Appendix 7: Individual Report – Zhu Junkun.....	

1 EXECUTIVE SUMMARY

There is no doubt that music is an indispensable spiritual companion in modern People's Daily life. Music is something that can inspire people, gives them hope, makes them believe in something greater, or lets them escape for a moment. It can be your best friend in time of need and is always there when you need it. The quote, "play the music, pause the memories, stop the pain, rewind the happiness." This is what music can do for you in simplest terms. Music is a tool that can bind all cultures together. Every culture makes music and it is neat to see how cultures can bond over music.

It is the great desire and demand of modern people for music that leads to the emergence of a large number of music and related music apps. In this era of information explosion, the vast network resources we have access to greatly facilitate our life, so that we can listen to a variety of music at will. Whatever it's pop, rock, folk, electronic, jazz, absolute music, rap, metal, world music, new age, classical, indie, ambient music, etc. which divided by genre. However, sometimes the more choices provided, the more difficult it is for users to find the favorite songs among the numerous songs. Many music apps have realized this and implemented intelligent personalized recommendations in order to help users discover their own interests and preferences. However, the existing recommendation function in the market is not that perfect, and the playlist acquired by users is often limited to a single style, which will form the information cocoon effect in the long run.

Our project team wanted to help music apps solve the problem of flexibility in recommendations. The intelligent recommendation robot developed by us can increase users' participation in playlist customization by allowing them to talk with the robot. After obtaining the information provided by users, our system can quickly customize different playlists in real time. This playlist customization is unlimited and can be adjusted at any time.

We used the third-party library Spotipy to scrape a large number of songs from Spotify to build a database. After obtaining a large amount of information from various dimensions of songs, we selected several song features that we thought were most suitable for subsequent data analysis: 'Genre', 'Energy', 'Mode', 'Danceability'. In terms of music recommendation, we chose Item-based Collaborative Filtering Algorithm. In order to increase the diversity

of our system, we combined this algorithm with informed search. In terms of the construction of the chatbot, we designed a simple and fresh chat interface that allows users to chat with the robot by vue.js. Besides, we used the python post method to resolve user preference data sent back by the front-end, and derived preference information by the rule engine, and got the music list.

Our team works very happily on this project and hopes that the project can be used and receive feedback by more uses. This project can be more international if we can cooperate with music companies and ask them to add our bots to the app as a complement to the recommendation system.

2 PROBLEM DESCRIPTION

A wide variety of music not only meets people's needs for different kinds of music, but also causes the problem that users are not clear about their own needs, it is difficult to summarize their interests and hobbies, and it is difficult to find songs in line with their own music aesthetics. Many music apps have realized this and implemented intelligent personalized recommendation in order to help users discover their own interests and preferences. If music is a dish, the music streaming platform's song recommendations are a savvy cook who pays attention to the dishes you like best, but more carefully considers your personal tastes. The importance of a strong and satisfying song recommendation module in a music app is self-evident, and it can even be said that this is the key factor for many users to choose an app and become its long-term users.

However, many music apps today, such as Spotify and netease Cloud, offer recommendations based on users and songs, respectively. The problem with such a recommendation system is that it makes recommendations subjectively based on the songs and artists users have heard. Therefore, the recommended song list obtained by users is often made for You song list with similar styles, so that users will be constantly solidified by their own interests, and their vision will become narrower and narrower over time, even the phenomenon of group polarization. And for users with strong subjective initiative, they may want to increase their participation in playlist customization and want to get some playlists that break the convention. For example, for user Ruby, she often listens to some Rock music, and the music

app she often uses knows her taste very well. Her song list is full of Hardcore Punk, Blues Rock, Britpop, Psychedelic Rock and so on, But one day, Xiao A suddenly wanted to listen to A quiet classical music, and it was very difficult to wait for her recommendation system to recommend A classical music.

2.1 PROJECT OBJECTIVE

Our music recommendation robot allows users to propose music styles they are interested in in real time through human-computer interaction. Whether you suddenly want to switch from traditional folk to rap or switch from electronic to world music, our robot can create a playlist tailored to your preferences in real time. This recommendation method will increase the degree of users' flexible and independent choice of music on the original recommendation.

We hope our project can be added to those existing Music Apps as a complement to the recommendation system. In today's market, our system can meet the following requirements of users and music software:

1. Retain new users

The daily recommendation function needs to collect the user's favorite song list and listening record, but when the new user just uses the website, the system does not have his behavior data, so it is not enough to obtain the user's music preference resources, and the prediction accuracy is low. At this time, the use of our flexible recommendation function can allow new users to choose and try different styles independently, while helping the system to collect and remember users' needs

2. Expand the scope of users

Our flexible and free recommendation system can satisfy users with different music preferences, and also greatly mobilize the enthusiasm of users to create playlists. It also creates a good social atmosphere that attracts other users who are not already on the social network.

3. Experience from shallow to deep

Our recommendation system enables users to enjoy a VIP customized feeling of music, and the conversation with the robot enhances the fun of using the app. The unlimited number of recommendations can also enable users to have a richer music experience.

3 KNOWLEDGE MODELING

3.1 GENERAL DESCRIPTION

As we know in CGS, we can embed our technology(dialogflow chatbot /recommendation algorithm) into our system to provide recommendation work.

Briefly speaking, our project consists of three different parts. And each part applied one technology included in our course.

We will separately explain how we build our knowledge m model in three parts of our projects.

3.2 DIALOGFLOW CHATBOT

3.2.1 KNOWLEDGE IDENTIFICATION

In this part,We represent our knowledge by using two things. Intents and slots.Every sentence sent by users will be detected as different intents and entities.

Intent detection is a classification problem. That our robot needs to classify users' sentences into different existing intents according to context.

3.2.2 KNOWLEDGE EXTRACTION

Once a sentence was classified as a kind of intent , the robot will then try to recognize those entities belonging to that intent. It will extract the essential information from the entities and then pass them to our backend system.

However , our backend system is using rule-based NER. According to different intents detected by robots, it will react differently : Reply with designed sentences or call our API to get recommendations.

3.2.3 KNOWLEDGE DEFINITION

We use users' past chat records to help our chat robot to get familiar with each entity and intents.

This work is to work with Google cloud NLP algorithm.

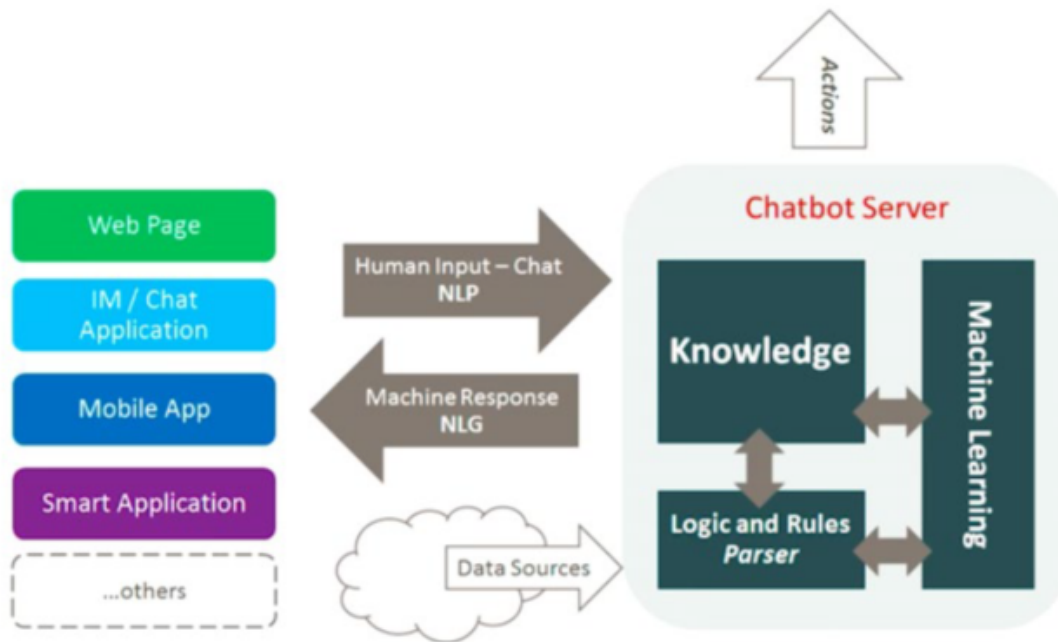


Fig1 explain the structure of our chatbot

3.3 COLLABORATIVE FILTERING RECOMMENDATION.

Knowledge specification:

Our original knowledge is our songs and their features. We use some scripts to scrap those songs' information from a musical website.

Raw data is usually not suitable for direct input into a model, thus we need to transform these data into a modeling dataset.

After that we take a collaborative filtering algorithm to deal with the dataset and do the recommendation.

We use this formula to calculate the similarity of different songs:

$$\text{sim}(\mathbf{i}, \mathbf{j}) = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_u)(r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_u)^2}}$$

Fig2 r means the feature's value of different songs.

3.4 INFORMED SEARCH TREE

3.4.1 KNOWLEDGE REFINEMENT

We design a rule to find a list of songs which have the maximum sum of similarity.

We take the similarity calculated from the collaborative filtering system as the model's input. Based on that we build a binary search tree and using greedy algorithm to find optimal solutions

We will further explain our algorithm in the next part.

4 SOLUTION OUTLINE

4.1 SYSTEM ARCHITECTURE

The system architecture diagram illustrates how the application in the front end interfaces with the python service, chat bot server and the recommendation engine. Below I will describe how our subsystems cooperate with each other from two typical user scenarios.

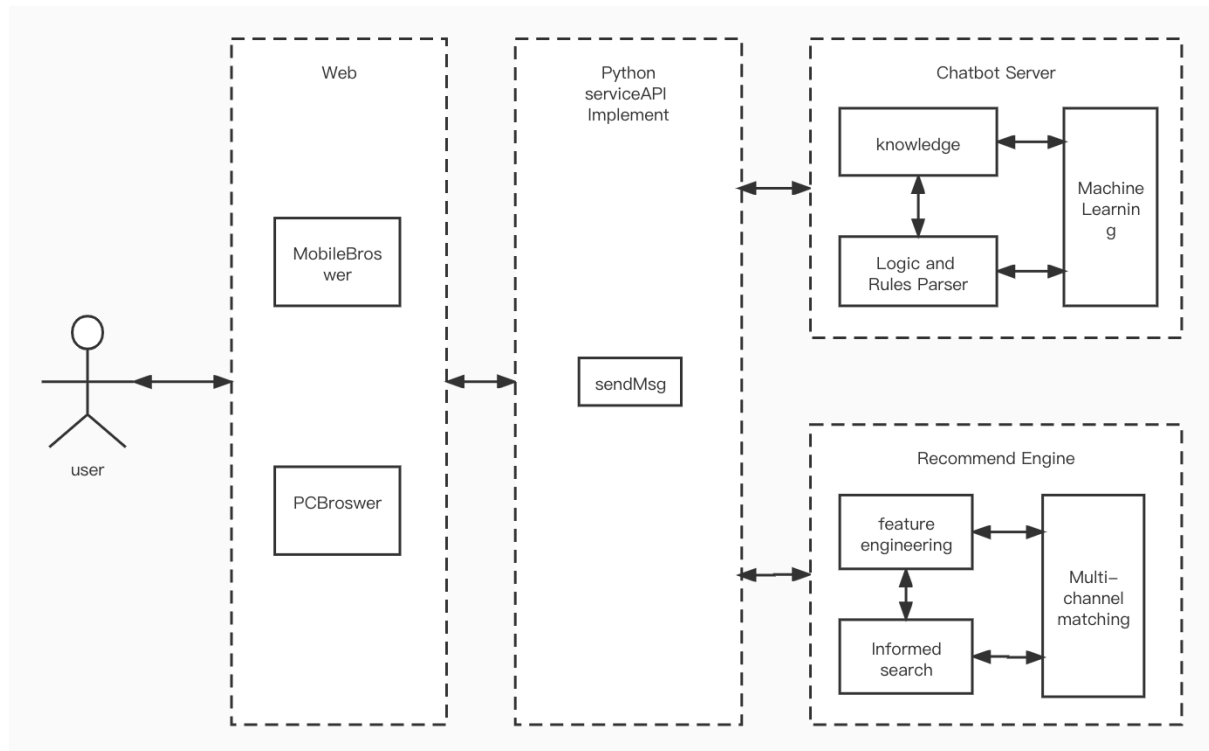


Fig3: system architecture

A typical user usage scenario is as follows. The user entered a sentence to the robot on the web page to ask the robot to recommend a few rock songs because he was in a bad mood and needed to vent, and suggested that he likes the Linkin Park band. The sentence was forwarded to the chatbot service (dialogflow) by python, and through Logic and rules parser inferred that rock is the song genre, Linkin Park is the user's favorite band, and returned information to the python service.

After the python service obtains the song genre and the artists, it inputs the two information into the recommendation engine. The recommendation engine searches and matches in the song library and gives ten rock-style songs back. Considering the diversity of recommended content, we ask that the first three of them come from the Linkin Park band, and the last seven are randomly selected from the same style of songs.

Another typical scenario is that the user likes a song and wants to hear more similar songs. Same as the above scenario, when the user tells the chatbot what song he likes, the chatbot service recognizes the song's name, returns it to the python service, and inputs the song name into the recommendation engine. The recommendation engine finds the similarity of all songs from the

item-item similarity matrix and sorts them. After getting the first two songs that are most similar and not the song inputted, we recursively search for the most similar songs of each of these two songs until we get ten songs. What is finally returned to the user is a song playlist with both similarity and diversity.

4.2 PROJECT SCOPE

The chatbot

Since our chatbot training is a continuous and highly time-consuming task, we provide a limited amount of corpus and train chatbot through tagging in the background. At present, the chatbot cannot recognize all song names and genres, and only supports ten genres and a small number of song name recognition.

The song library

The recommendation of the robot depends highly on the richness of the library. At present, our data comes from crawling the song data on the spotify platform including 22 features of the song. The last update of the district library was in October 2021.

The recommendation engine

Based on the data we obtained, we only selected four features (genre, mode, danceability, energy) to construct our similarity matrix. But it is very easy to expand.

4.3 SYSTEM'S FEATURES

From the user's perspective, we have implemented three functions in total: have daily conversation with the chatbot, specify the genre and artist to request the chatbot to recommend music, and specify one favorite song to request the robot to recommend more music.

Daily conversations with robots

You can just start a daily conversation and chat with the robot. The daily conversations currently supported include greetings and say bye.

Specify the genre and artist to recommend songs

You can describe your favorite genre and artist to the chatbot, and ask the chatbot to recommend related songs. The genre is mandatory, and the artists are optional. This is not a technical limitation, just product design.

Recommend more similar songs based on a song

You can tell the robot a song you like, and he will recommend more songs similar to this song. But there is a prerequisite, the song must be in our song library.

4.4 CORE IMPLEMENT

The chatbot

The chatbot solution we use is the dialogflow which is mentioned in the course. Dialogflow provides us with a very complete SDK. We only need to install the SDK and call the API to easily build a chatbot on the project. Simple engineering implementation allows us to focus on how to evaluate user scenarios, build corpus based on the scenarios, and finally train the robot as a training set.

First we designed a use case. The user greets the robot first, then proposes his favorite genre and favorite singer, and asks the robot to recommend some music. After getting the playlist, the user said that he liked one of them very much and hoped to hear more similar songs.

Based on the user case above, we divide the question into four Intents, one for identifying style, one for identifying song names, and two for daily conversations. In the genre intent, we set up two slots, one is used to identify the genre, the other is used to identify the artist name.

After preparation, we start the service and send the chatbot to friends to get enough corpus. In the end, we collected hundreds of data as our training set within an hour. After manually tagging, our chatbot gradually became smarter.

Item-based collaborative filtering recommender

There are many mature recommendation algorithms but how to choose a recommendation algorithm that suits our scenario? Our choice is based on the following three conditions: fast inference speed, no need for user privacy information, and better cold start performance.


Since our system is a real-time question and answer scenario, we have higher requirements on the time cost of the algorithm when selecting the recommendation algorithm. Users don't want to receive a reply from a robot after two minutes. This is not a chat bot, but a message board. The similarities between items change much less frequently than between users and all item-item distances can be pre-computed. These characteristics make the item-based collaborative filtering algorithm have a high inference speed.

What's more, since it is difficult for us to legally obtain user data, all algorithms that require user data have been eliminated. Another advantage of using item-based algorithms is that since the features of each song are already existing, in a cold start situation, the algorithm will perform better than algorithms that rely on history data.

Data acquirtance

We first found a song data set provided by Spotify on kaggle, but the data lacks artist information and the song genre is not popular enough. After technical pre-research, we decided to use the data crawling interface provided by Spotify to get the data ourselves.

The library we use is Spotipy which is a lightweight Python library for the Spotify Web API. With Spotipy we can get full access to all of the music data provided by the Spotify platform.



2.19.0

Welcome to Spotipy!

Features

Installation

Getting Started

Authorization Code Flow

Client Credentials Flow

IDs URIs and URLs

Customized token caching

Examples

API Reference

client Module


oauth2 Module

Support

Contribute

License

Indices and tables

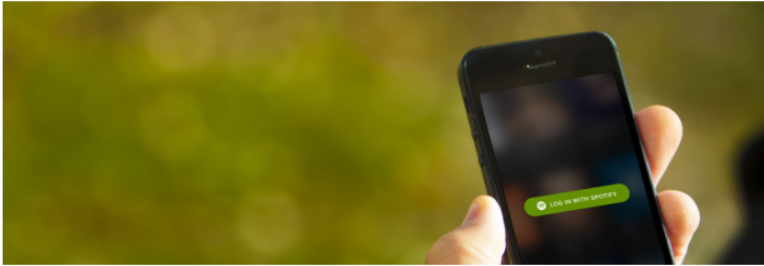


Real-Time, Intraday & Historical
Market Data API Free forex data for
hundreds of currencies **Get Now**

Ad by EthicalAds · Monetize your site

Docs » Welcome to Spotipy!

Edit on GitHub



Welcome to Spotipy!

Spotipy is a lightweight Python library for the [Spotify Web API](#). With Spotipy you get full access to all of the music data provided by the Spotify platform.

Assuming you set the `SPOTIFY_CLIENT_ID` and `SPOTIFY_CLIENT_SECRET` environment variables, here's a quick example of using *Spotipy* to list the names of all the albums released by the artist 'Birdy':

```
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials

birdy_uri = 'spotify:artist:2WX2uTcsvV50nS0inACecP'
spotify = spotipy.Spotify(client_credentials_manager=SpotifyClientCredentials())

results = spotify.artist_albums(birdy_uri, album_type='album')
albums = results['items']
while results['next']:
    results = spotify.next(results)
    albums.extend(results['items'])

for album in albums:
    print(album['name'])
```

Here's another example showing how to get 30 second samples and cover art for the top 10 tracks

Fig4: Document of Spotipy

Informed Search

The implementation of our first version is based on the similarity and directly returns the N songs that are most similar to the input song. But after we used the application, we have found that the returned songs are highly similar, which can easily cause aesthetic fatigue. So we thought about a new way to increase the diversity of songs while maintaining a high degree of similarity.

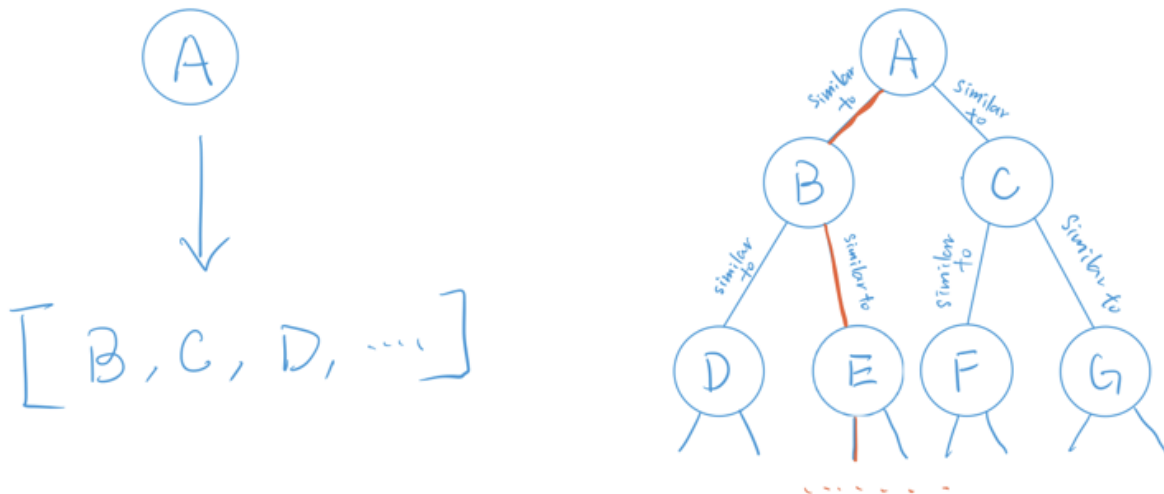


Fig5: comparison between top-n and informed search

We build a similarity tree in which the value of a node is the similarity value with its parent. And the height of the tree equals the number of songs we need plus 1, which is 11 in the app. After building the tree, we look for the path with the largest sum of similarity, and return all nodes on the path.

4.4 FUTURE IMPROVEMENTS

The update of song library

At present, our music library needs to manually run the script and specify the playlist id to crawl the data. In the future, we can use a timed task to update the music library regularly according to customer needs.

Train the chatbot with high efficiency

At present, the way we obtain training corpus and tagging is manual. In the future, we can automatically generate labeled training corpus through scripts and continue to optimize our model.

Improve algorithm efficiency

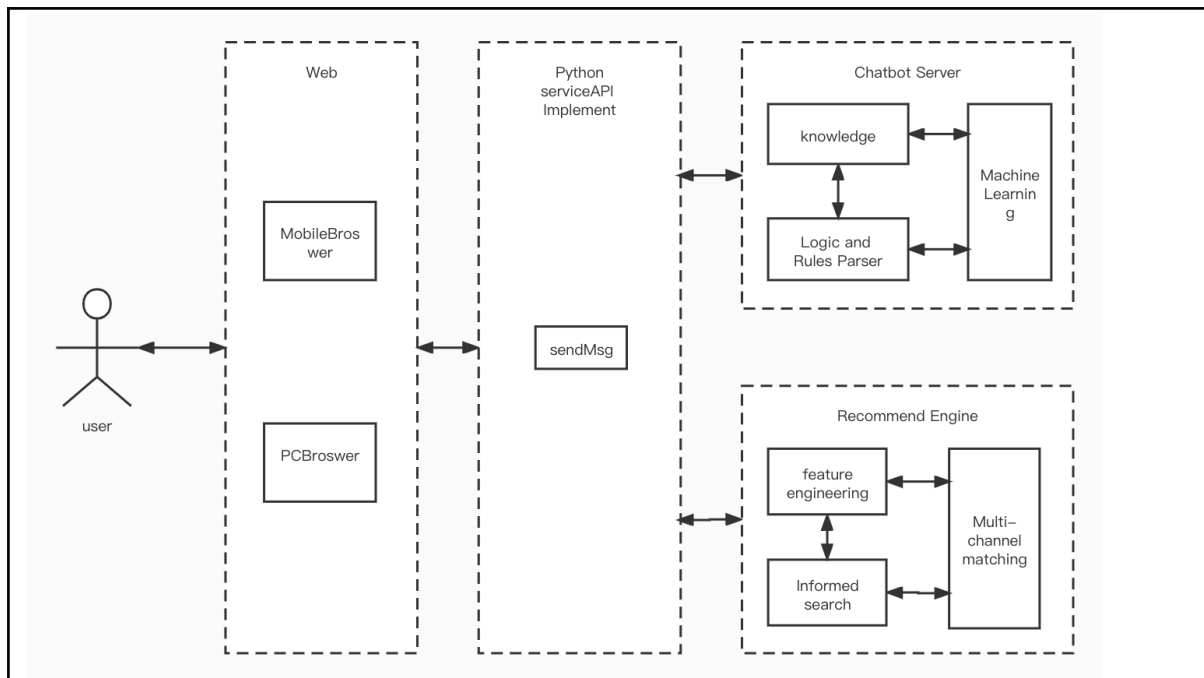
Currently, we support two matching methods: top-n matching and informed search matching. The default is informed search. But if the number of recommendations increases, the time complexity of the algorithm will

increase exponentially. In the future, we can specify rules according to the number of recommendations we need, and combine these two strategies to formulate new rules to recommend.

Appendix 1: Project Proposal

PROJECT PROPOSAL

Date of proposal: 4/11/2021
Project Title: Your Song—— Intelligent songs recommender robot
Group ID (As Enrolled in LumiNUS Class Groups): Group 15 Group Members (Name , Student ID): LIN QIANQIAN A0231438J LIU JING A0231403Y YANG MINGRUN A0231344R ZHU JUNKUN A0231471N
Sponsor/Client: <i>(Company Name, Address and Contact Name, Email, if any)</i> Institute of Systems Science (ISS) at 25 Heng Mui Keng Terrace, Singapore NATIONAL UNIVERSITY OF SINGAPORE (NUS) Contact: Mr. GU ZHAN / Lecturer & Consultant Telephone No.: 65-6516 8021 Email: zhan.gu@nus.edu.sg
Background/Aims/Objectives: The proposed Intelligent Songs Recommender Robot will make use of various songs information and features to custom songs list, users can use the chat function to communicate with the robot and share his music preference.
Project Overview: System architecture design



Our system mainly include following three parts:

The chatbot, since our training of robots is a continuous and highly time-consuming task, we provide a limited amount of corpus and train chatbot through tagging in the background. At present, the chatbot cannot recognize all song names and genres, and only supports ten genres and a small number of song name recognition.

The song library. The recommendation of the robot depends largely on the richness of the library. At present, our data comes from crawling the song data on the spotify platform, and the features of the song. The last update of the district library was in October 2021.

The recommendation engine. Based on the data we obtained, we only selected four features (genre, mode, danceability, energy) to construct our similarity matrix. But this is very easy to expand.

Data acquisition

Spotify is not only the world's largest legal streaming music service platform, but also provides powerful Web API analysis functions for users to use, so our database is built on the Spotify music library.

I used the third party library python-Spotipy to obtain the token to establish a connection with the server. I then got playlists for different genres and information about each song in the playlist

Resource Requirements (please list Hardware, Software and any other resources)

To use our system

Hardware: any device has a browser.

Software: a browser.

To deploy the service

Software: server and python 3.6 env; GPU is not mandatory.

Correlation model algorithm

We use an item-based collaborative filtering algorithm to build an item-item similarity matrix for all songs in the library. This calculation is off-line, so in real-time inference, the time-consuming is very low.

Appendix 2: Mapped System Functionalities against the knowledge

Business resource optimization

We use Informed search to do the business optimization of our recommended result. When the recommendation algorithm returns the recommended list, we construct a similarity tree and return the list searched from the tree instead of the top-n similarity list. This optimization method provides diversity for algorithm results while maintaining high similarity.

Knowledge Discovery & data mining

We construct a recommendation engine to do the music recommendation. With the dataset crawled by ourselves, we use the item-based collaborative filtering algorithm to do the data mining which is finding the similarities among each song in the dataset.

System designed with cognitive techniques

We use a DialogFlow to support our user-interface of human-mode communication and train the model by constructing a corpus with a manual label. The cognitive system uses four intents designed by ourselves to support specific music context conversation.

Appendix 3: Installation and User Guide

Download the code from the Github repository

https://github.com/twinkletwinklelittlestar70/your_song

Import dialogue chatbot

<https://github.com/twinkletwinklelittlestar70/IRS-PM-2021-07-05-IS21FT-YourSon-YourSong/tree/master/SystemCode/algorithms>

1. Open the dialogflow

2. To import an entity:

Click the more  icon.

Click Upload entity and choose the file.

Create a Google Cloud account to get the key (private key)

Step by step according to the official document:

<https://cloud.google.com/dialogflow/es/docs/quick/setup>

The Google Cloud can provide public keys to provide open services to enterprise users.

Installation:

Environment Requirement and run(frontend):

1. Install NodeJS and npm

2. Checking the Installation

->node -v

->npm -v

3. Go to the frontend directory and run:

->npm install

->npm run serve

4. Open your browser and type <http://localhost:8080/>

Congratulations! See the front end!

Environment Requirement and run(backend):

1. python 3.6

2. Go to the Backend directory, run

->pip install -r requirements.txt (It contains all the dependencies we need)

3. Create the static folder in the backend directory

4. Go to the frontend directory and run

->*npm run build*

5. Check the Static folder

6. Change the contents of the key in the code to yours

->`os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = "key"`

7. Starting the Flask Service(CMD)

->*set FLASK_APP=hello*

->*flask run*

* Running on <http://127.0.0.1:5000/>

8. Open your browser and type <http://127.0.0.1:5000/>

Congratulations! Use your Song now!

User Guide:

Once you start the service, you can open Your Song to recommend songs!

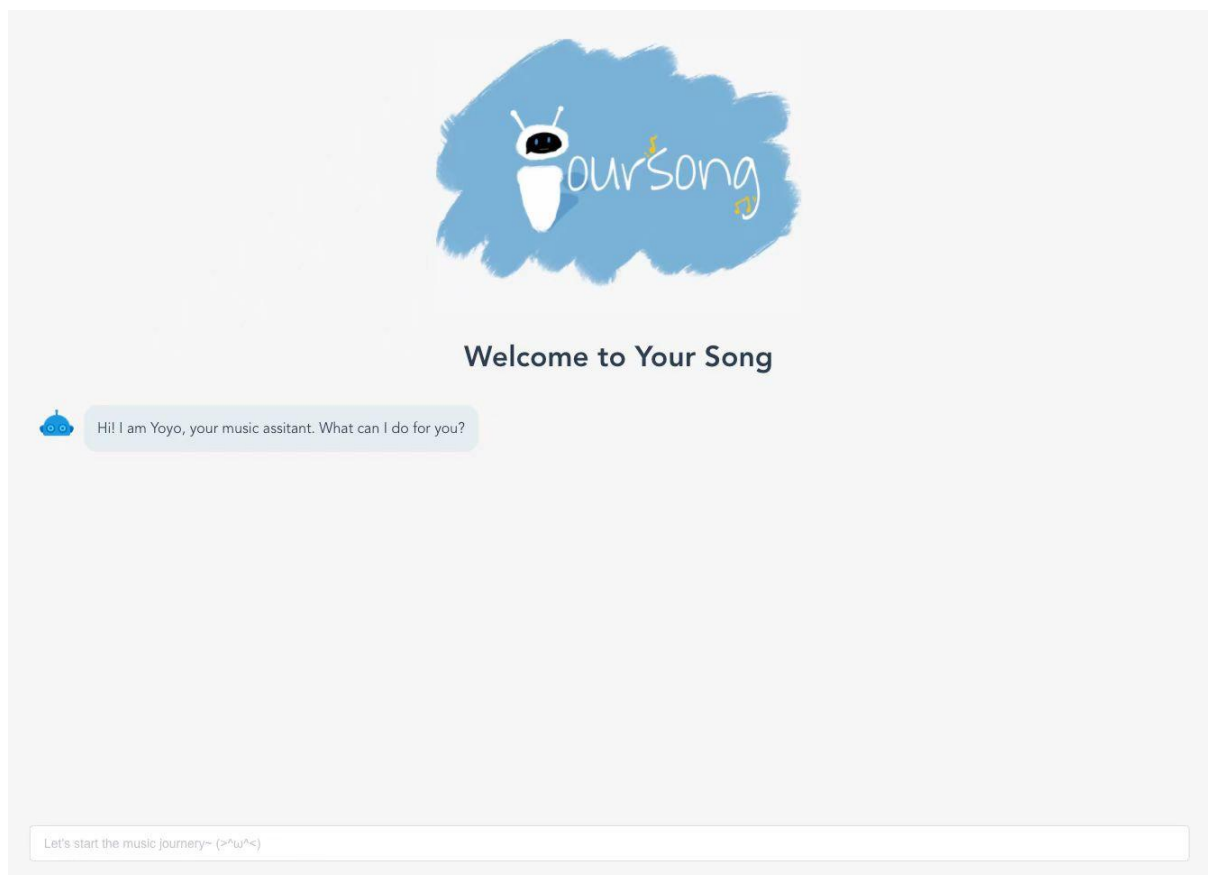


Fig6: The user interface

1. Just entered the interface, you can simply say hello, relax yourself
try : "Hi" "How are you?"

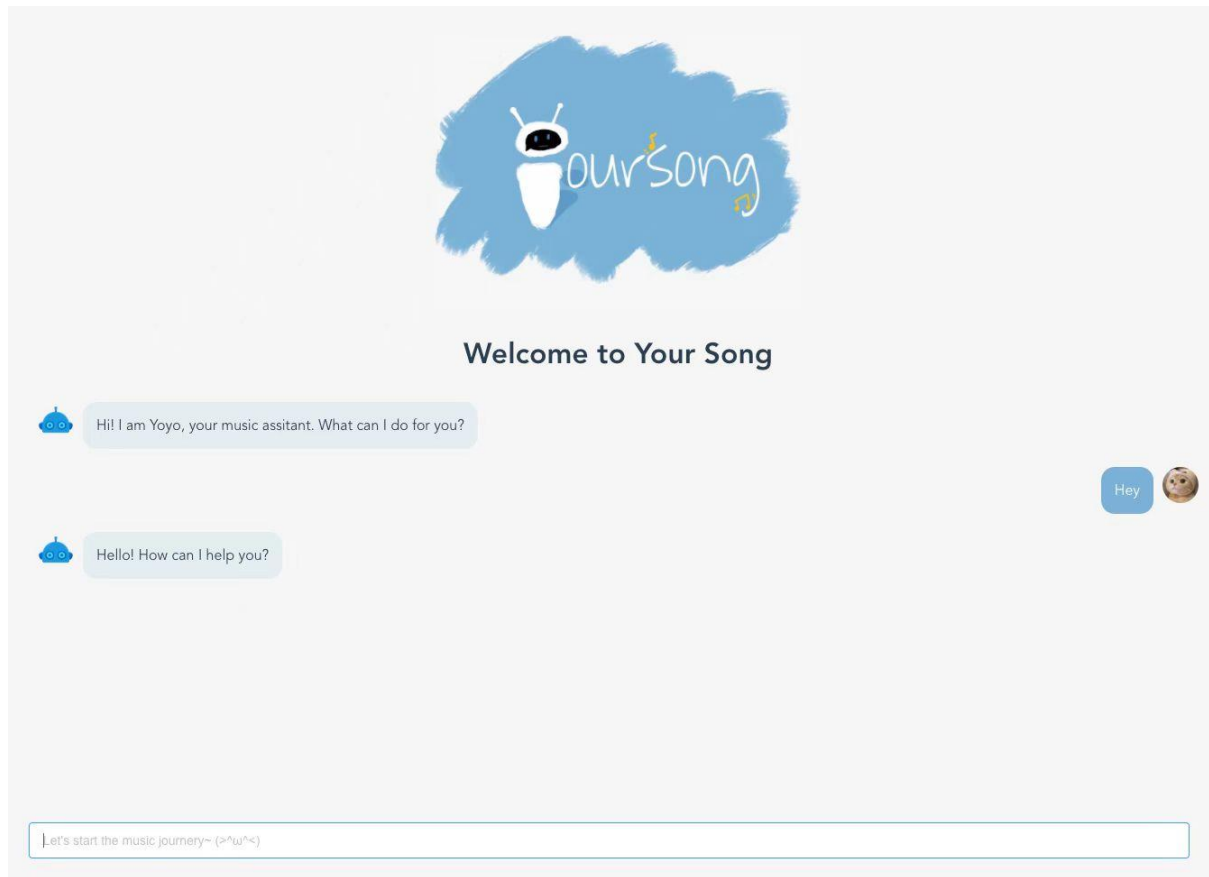


Fig7: Say "hello"

2. When you want to listen to music, which is the core of it, you can get it to recommend it in the following ways

a) First of all, it can be recommended according to the genre of music

Try : "Recommend me some popular music"

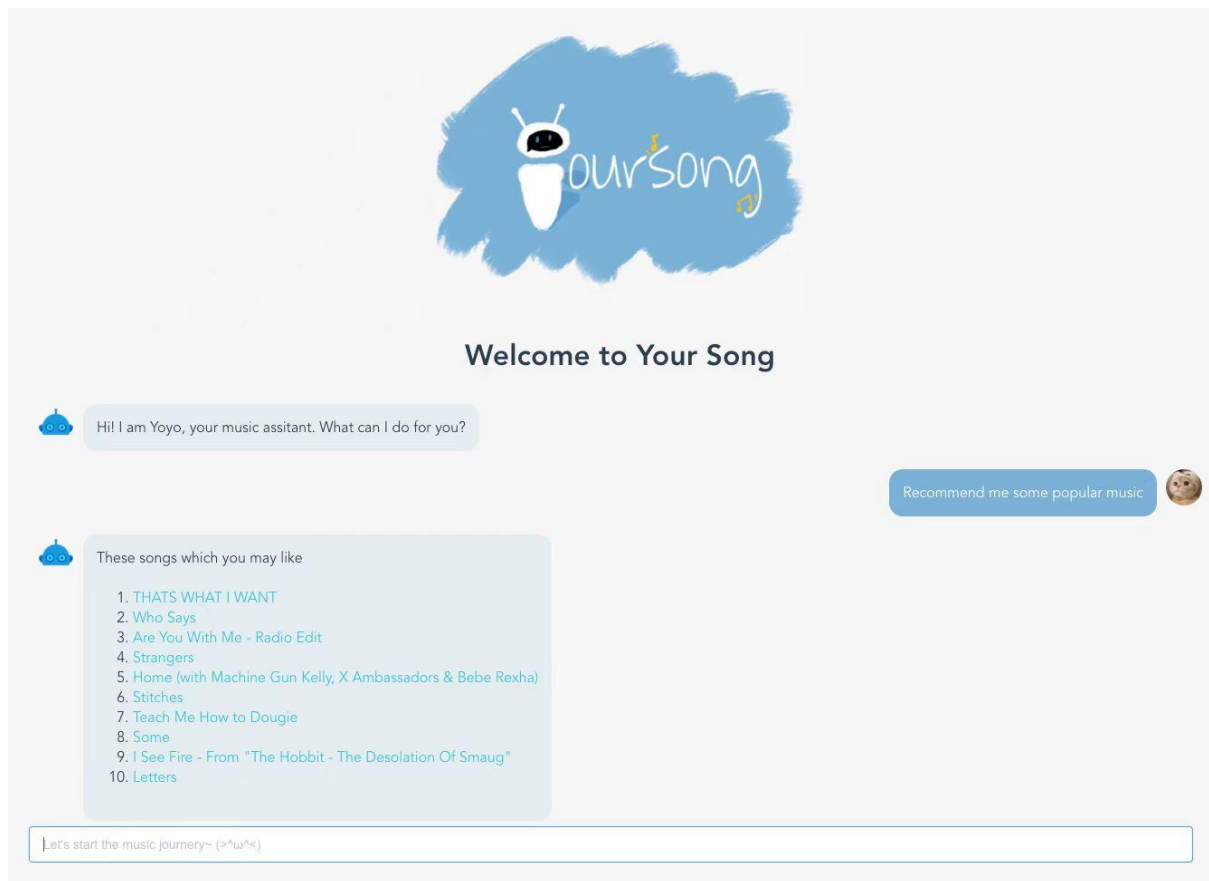


Fig7: Recommended according to the genre of the music

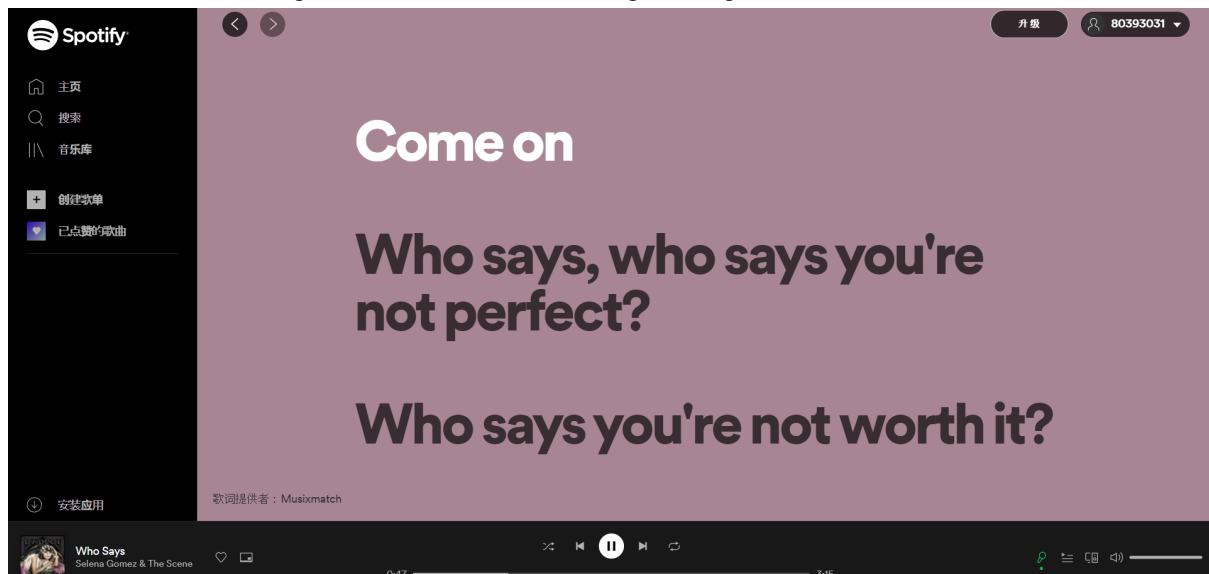


Fig8: Click the link

Of course, you can also want other styles !

Notice : Currently we support nine popular styles, They are :

popular, rock, folk, hiphop, rnb, jazz, electronic, classical, absolutemusic

b) Then it can also recommend according to your favorite songs, try to say some song names

Try: "I like something similar to We Are The World"

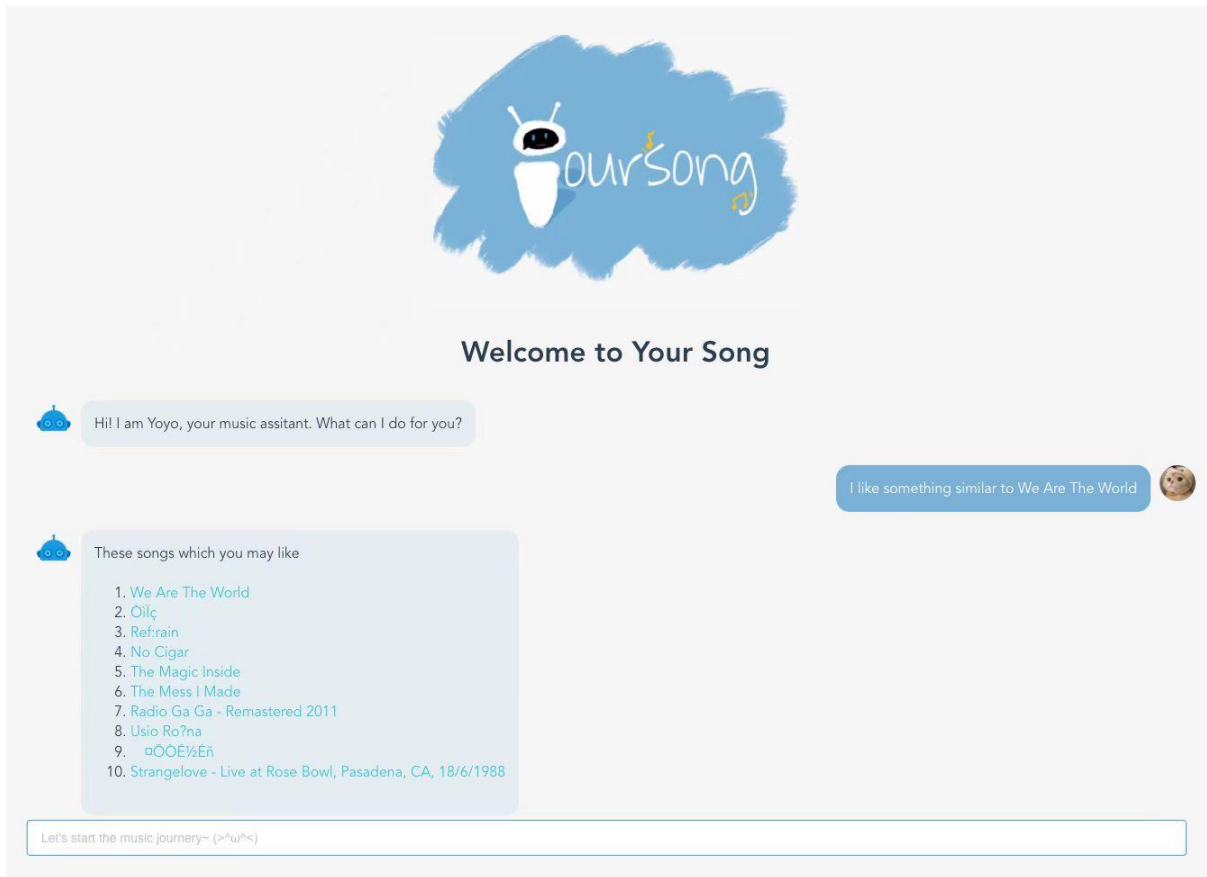


Fig9: Recommended by the song name

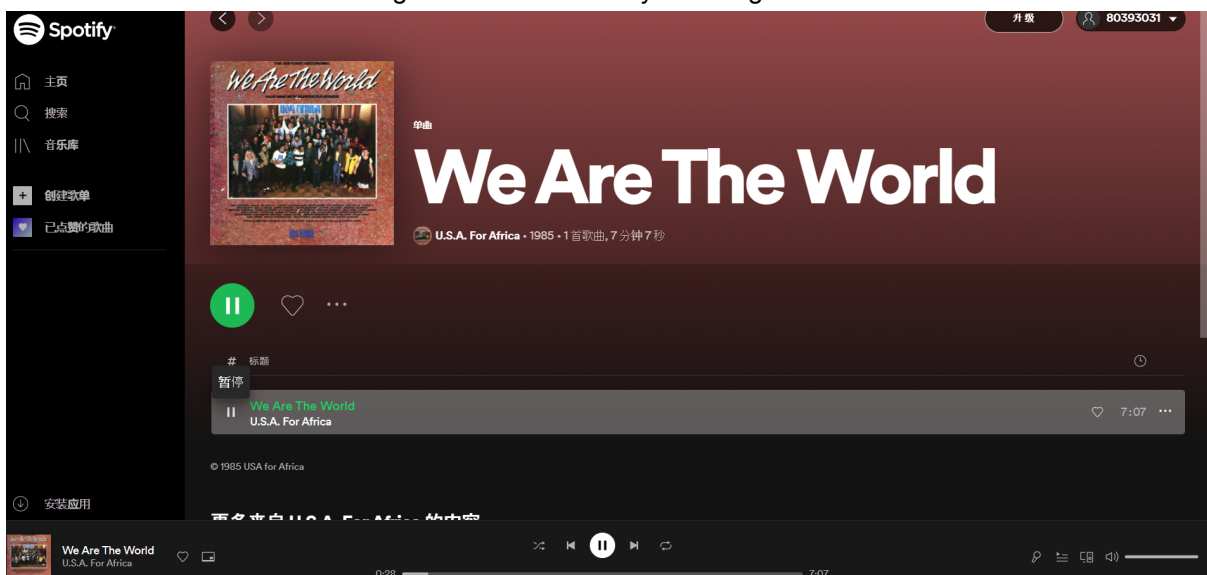


Fig10: Click the link

Notice:Capitalize the first letter of each word when typing the song title.

c) It can also be recommended when you want to listen to similar styles of certain singers, give it a try.

“Recommend me some popular music and I like Linkin Park”.

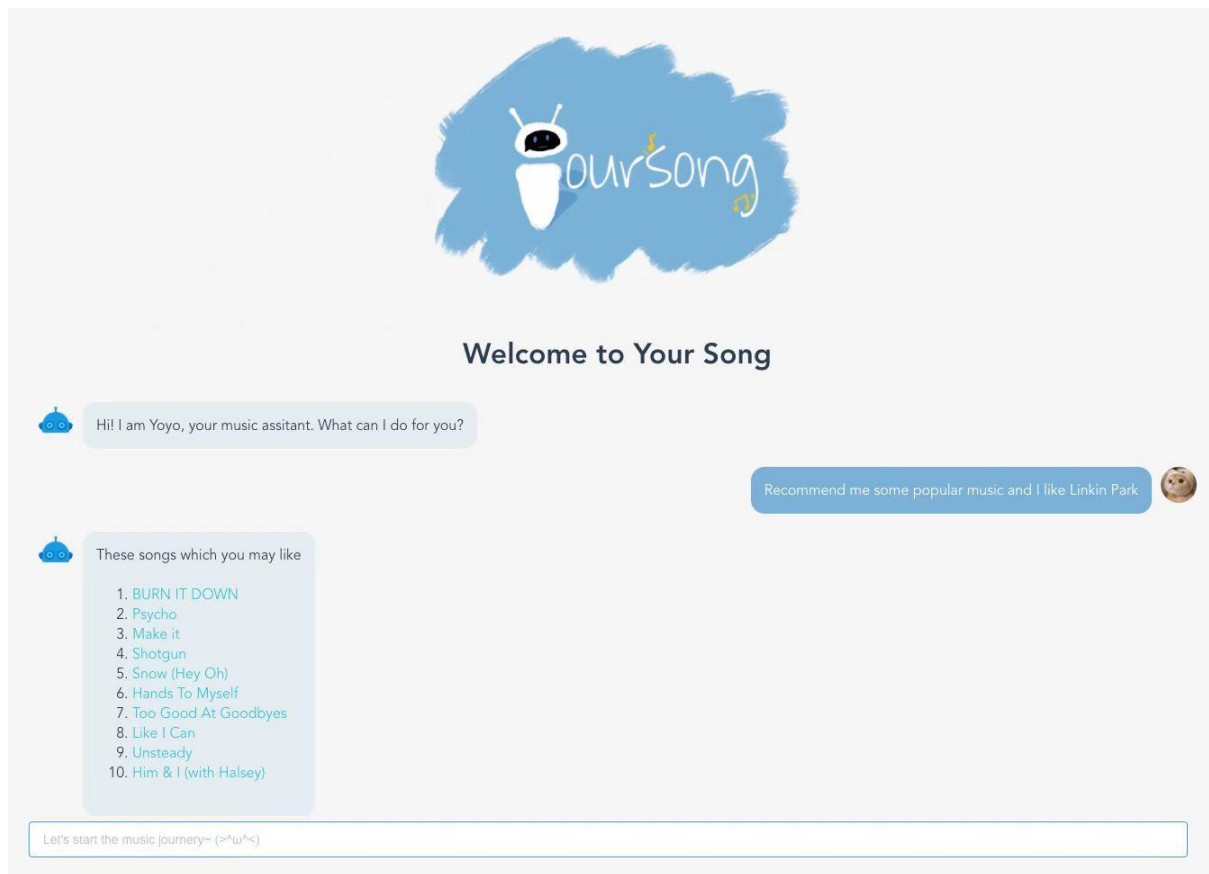


Fig11: Recommended by the name of the singer

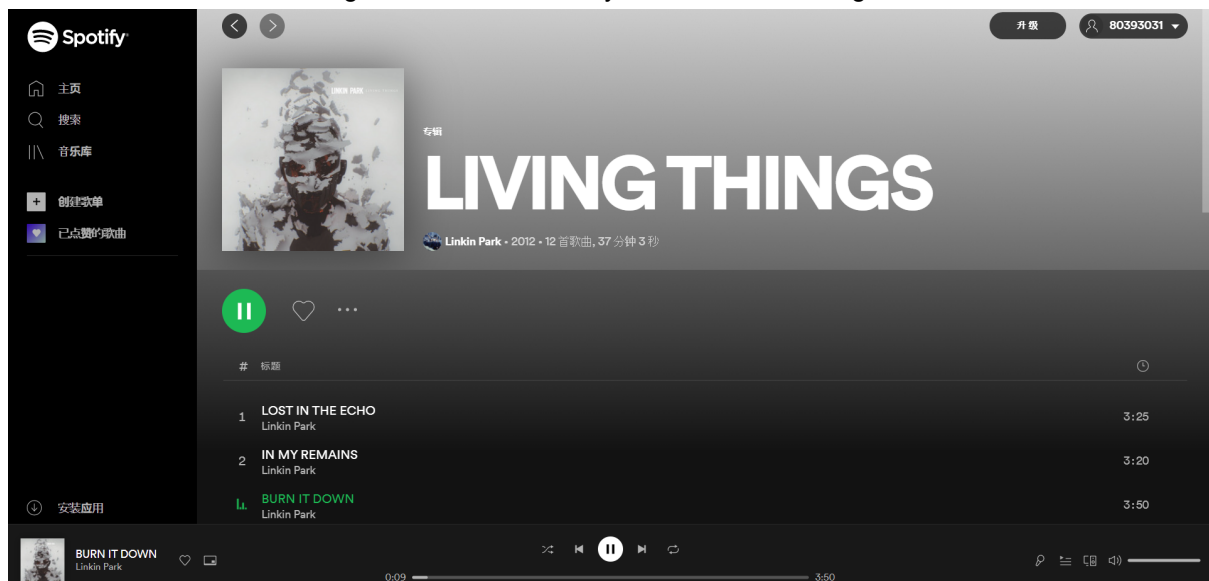


Fig12: Click the link

3.Recommend 10 songs each time, from top to bottom, in order of high similarity to low.

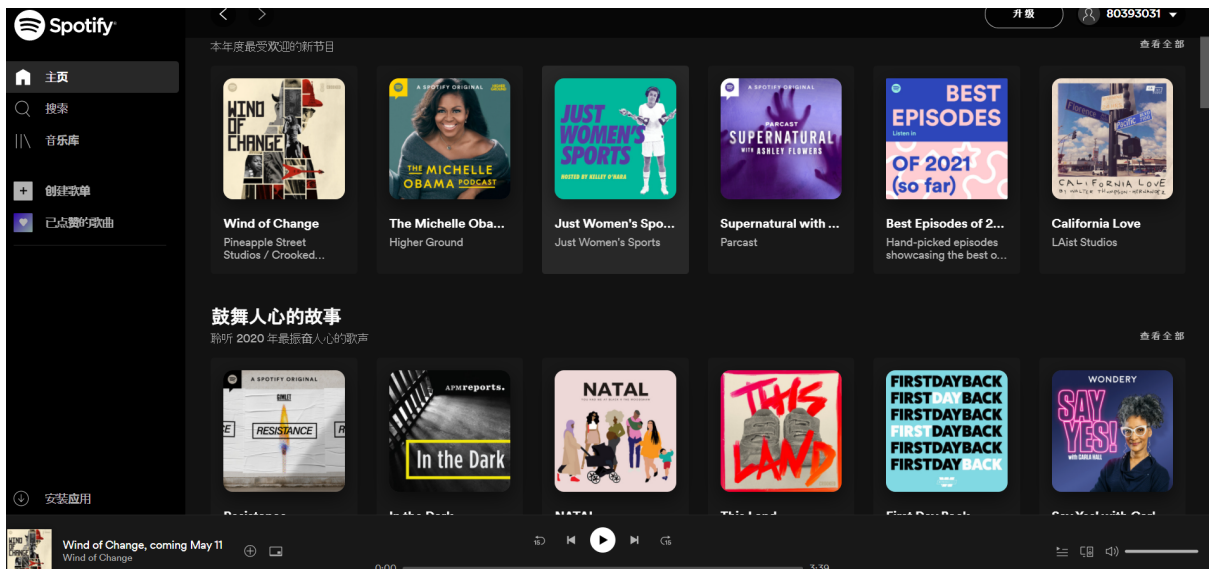


Fig13: Music source



Fig14: Recommended Songs

Notice: All song data comes from Spotify, Because songs are written in different languages around the world, the titles of songs are occasionally garbled.

Appendix 4: Individual Report – Lin Qianqian

Lin Qianqian A0231438J

1. Personal contribution

I contribute to the project in the following four parts.

First of all, I did a pre-research on the overall technical plan of the project. I collected technical reference materials including existing research and code. And then we all discussed together to determine the overall technical solution and the typical user scenarios.

Secondly, I was responsible for the recommendation part and I finished technical research, evaluation and the final implementation. I do the pre-research for the recommendation part, collecting reference articles and code implementations, and discuss with Liu Jing to determine our final solution. And I do the coding for our recommendation engine in python.

Thirdly, after the completion of the similarity calculation algorithm development, I optimized the top-n matching, discussing with Zhu Junkun, by using greedy search to improve diversity in our return recommended playlist.

At last, I completed the web framework initialization, the front-end development and back-end interface implementation in engineering work.

2. What is learnt?

Recommendation Algorithm

At the stage of program research, I learned some popular solutions online, such as collaborative filtering, word2vec method to convert a song into a vector to calculate similarity, and algorithms such as NMF and SVD. In order to choose one, I learned the advantages and disadvantages of each algorithm to help me decide the final solution. I also found some reference codes and articles.

In the process of project implementation, I first use colab to write and run our recommendation algorithm. In this process, I learned the basic usage of Dataframe in the panda library and became familiar with the python language.

After completing the algorithm demo, I started to integrate the algorithm in the flask framework. I first designed and documented two API functions that my

recommendation engine needs to expose. Then based on the API, I started to develop. I encountered some problems, but I gradually solved them.

Informed Search

The idea of using Informed Search instead of TopN matching came from Zhu Junkun, and I implemented it. In the process of algorithm design, I reviewed uninformed search and informed search algorithms on PPT, and finally chose the simplest greedy algorithm because of its best time performance among search algorithms such as greedy and A*.

When implementing the algorithm, based on the principle of "do not reinvent the wheel", I used a library called anynode to help me build faster. But there are still some details that need to be considered. For example, the ancestors of each node cannot be repeated, because we need ten songs that are not repeated. Although this is not a very difficult task, it still took me some time to implement the algorithm.

During this process, I became more familiar with search algorithms and binary tree data structures.

Web Development

Since I am very familiar with web development, this part of the work progressed smoothly without any problems.

3. How can you apply the knowledge and the skill gained?

I think the most important thing in this process is that I have learnt how to get started in an AI area quickly. How to do technical research in AI, how to read papers, and how to compare different solutions. I believe these abilities can help me in the direction of AI learning in the future.

In addition to the ability to learn, I have also improved in technical understanding. When I reviewed the PPT, some points I missed in class were also noticed and had better understanding.

In addition, I am very interested in word embedding algorithms, and I am about to learn a new field of NLP. After learning the field of NLP, I will try to use the song2vec algorithm to improve our products.

Appendix 5: Individual Report – Liu Jing

Liu Jing A0231403Y

1. Personal contribution

As for me, this project is meaningful and interesting. I had a great time working with the team members, and I really appreciate their help and encouragement during the project.

In actual development, I was mainly responsible for the construction of the database and the implementation of the music recommendation algorithm with Lin Qianqian. Besides, I was involved in evaluating the recommendation system solution and evaluating the feasibility of the overall solution.

2. What is learnt?

1) Grasp data

In the database construction, Spotify is not only the world's largest legal streaming music service platform, but also provides powerful Web API analysis functions for users to use, so our database is built on Spotify music library. The steps to build the database are as follows: First, using Spotify for Developers to create your own app and get clientID and Client Secret. Secondly, Obtain the token through the third-party library python-spotipy to establish a connection with the server. Third, I took playlists from different genres like Pop, Rock, Folk, Electronic, Jazz, Absolute Music, Rap, Metal, World Music, New Age, Classical, Indie, Ambient Music. Eventually, I obtained information and features from each song in the playlist.

Considering the curse of high dimensionality and overfitting, I also learnt different feature selection methods like filter method, wrapper method, embedded method to choose those features which can describe the similarity between songs best.

2) Implementation of recommendation Algorithm

In order to implement the recommendation algorithm, I checked many references and tried the baseline algorithm, Item-based Collaborative Filtering Algorithm and matrix factorization-based(SVD, PMF, SVD++)algorithm. During the process of learning, I learnt the features and application situation of different models.

3) Development ability

Our project has complete front and back end development, I learned a lot about vue.js based front-end development and Python-flask backend

development in the process. Practice is the best way to learn and it is important to combine artificial intelligence with practical applications.

3. How can you apply the knowledge and the skill gained?

The skills I learned through this project will be of great help to my future study and work.

First of all, in the future, data set problems will not be so difficult for me, I will be able to make my own data sets that meet my needs in a formal and reasonable way, and I will be able to apply this skill appropriately for any tricky or niche problem.

Secondly, I will learn to evaluate algorithms in different ways when facing different problems, compare the characteristics of different models and select the most appropriate solution for the problem.

At the same time, I hope to be more involved in the development of projects in the future. Users can intuitively feel and use the front and back end, so I think this is also a very important skill.

Appendix 6: Individual Report – Yang Mingrun

Yang Mingrun A0231344R

1. Personal contribution

The experience of completing this project with Lin Qianqian, Liu Jing, and Zhu Junkun has been a lot for me. We spent a lot of time with these project members from setting the theme to the end, and putting our ideas into practice bit by bit. This made me feel a sense of accomplishment. My project members are also very good at communicating, helping each other, and helping each other. Discuss and solve problems with each other, it is an honor to work with such a team.

In actual development, I was mainly responsible for the backend development and dialogflow. At the same time, I also take the initiative to participate in our weekly project team meeting to share the research on dialogue robots and the new work progress in the project, and discuss new division of labor with project members. It is very pleasant to discuss with the team and exchange new knowledge.

2. What is learnt?

During this project, I learned something about dialogue bots and back-end development.

(1) DialogFlow

DialogFlow, its predecessor was API.ai, which was later acquired by Google. Its framework is based on Google Cloud's natural language understanding.

Before we decided what to use to build our chat bot, I first looked for and researched many different ways to build a chat bot, and finally decided to use dialog flow. Because it incorporates Google's machine learning expertise and Google Cloud Speech-to-Text and other products, and it is a Google service that runs on the Google Cloud Platform and allows you to scale to hundreds of millions of users, it is very suitable for beginners, intuitive, and has a complete and clear article.

And I also learned how to deploy a real chatbot. First, understand the concepts (intent, entity, context...) through the first gentle method of the graphical user interface, and then understand a more "coded" approach. Method, use the Python client to make RESTful API calls to reproduce the same agent but use code (instead of a click in the GUI). Finally, when we can use Python API calls to build agents, we will use them to create end-to-end chatbot projects that

you will deploy on multiple channels (Slack, Facebook, Telegram...).After you understand this, I can try to build my own chatbot.

(2)Back-end development

In the whole project, I completed the development of the back-end function. The python web framework I chose is Flask, because this framework is quite convenient to build an API for front-end. But I haven't used it before, so it is a bit of a challenge for me.Fortunately, this framework is simple and easy to learn.

The function I implemented is to recommend songs according to the genre when the intent is recognized. When recognizing the intent, the singer name recommends the song according to the singer, and when the song name is recognized, it is recommended according to the song name type. When nothing is recognized At that time, we will return to our default reply. In the process of implementing these functions, I learned the two HTTP request methods GET and POST. The GET method gets the information in the chatbot, and the POST method submits the intent to the recommendation algorithm and corresponding result.

(3)Teamwork

Because it is a project cooperation, So I discussed with Lin Qianqian who is responsible for the recommendation algorithm to determine the front-end and back-end API, which made the development efficiency higher. Similarly, I communicated with Zhu Junkun, who is responsible for building the dialogue robot, in time to determine the required intention and legality. When I have a problem in the development process, the project members will share their experience and knowledge to help me, so the efficiency in the development process is very high, and this experience also helps me learn how to communicate with the team.

3. How can you apply the knowledge and the skill gained?

First of all, I gained the skills and experience to build a chatbot. I have a better understanding of natural language processing. Now chatbots are in demand in all walks of life. I can apply the skills of this project to participate. Continue to use it in other similar projects.

Secondly, I have also understood the process of back-end development and how the front and back end interact, which is an essential skill and experience for my future work. Combined with the call to the database in the future, I believe I can apply this knowledge in many development projects.

Third, I learned how to cooperate and communicate in the team. Regular meeting summary is very necessary, the document can help us to arrange work processes, clearly see everyone's division of labor, and also can be summarized to sort out what we have done all the work of some links, data set position, etc. Details can be recorded, convenient to everyone in view when necessary. In teamwork, I developed good code habits. In order to facilitate others to read, the code should have certain norms.

Appendix 7: Individual Report – Zhu Junkun

Zhu Junkun A0231471N

1.personal contribution to group

I give our team members this idea according to my own demand.

Luckily, they accepted this idea and worked on this project together with me.

After discussion, I designed the whole structure of our project as three parts : Dialog Flow , Collaborative filtering algorithm and informed search algorithm.

In the coding part, I'm in charge of designing the Dialog flow agent and work with another team member to implement the back-end service.

I also participate in designing the special informed search algorithm to make our recommendation more interesting.

2. what learnt is most useful for you

I used to do some deep learning work in my Undergraduate laboratory, but we only deal with some algorithm problems.

This is the first time for me to successfully transform the relevant technology of artificial intelligence into a project that can be applied to life scenes.

The most useful thing I learnt is how to understand the different artificial intelligence technologies and put them into the right place.

The technology and knowledge can be learnt everywhere, but how to apply them into solving problems is the most important part to me.

Also, understanding how to work with others as a team is essential to me. In this project, everyone has their own advantages, like Lin qianqian's work experience, LIU jing's creative ideas and YANG mingrun's hardworking.

Thus after several discussions with our team. We finally design a structure for the application and also choose the right technology to implement our functions.

3.how you can apply the knowledge and skills in other situations or your workplaces

Now I have a basic understanding of building an intelligent agent to do recommendations.

In the process, I may have the skill to find different API services , and understand them by reading their help document, especially the google cloud.

When I need to build another application in the future to solve a different artificial intelligence problem, I will firstly recall what I learned from the class and then search on the internet to see whether I can get help from an AI service provider like google.

I would also find people with the same interests to work together.

With the various knowledge from the team and the help of tools and APIs, it's easier for me to build a great application when I have a great ideal in the next time.