

Music Recommendation System

This report focus on building a recommendation engine that suggests songs a user has not listened to yet. The system is designed to take a user ID as input and recommend songs tailored to their preferences. The goal is to combine personalization and diversity in the recommendations.

To achieve this, the system uses a hybrid approach that combines;

1. Collaborative Filtering (CF). Suggests songs by finding users with similar preferences.
2. ContentBased Filtering (CBF). Recommends songs based on the details of songs a user has already listened to like the artist or title.

By blending these two methods the system ensures accurate and diverse recommendations.

Approach

The provided code utilizes the following steps to generate recommendations

1. Installing Surprise

First, a library called Surprise was installed in Google Colab environment.

Code snippet

```
!pip install surprise
```

1. Importing Libraries:

Then imports of all the necessary libraries like pandas for data manipulation, sklearn for some calculations, ipywidgets to help make interactive function, and components from surprise like SVD (Singular Value Decomposition) for the recommendation algorithm was done.

Code snippet

```
import pandas as pd
from surprise import SVD, Dataset, Reader
from surprise.model_selection import train_test_split
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer
import ipywidgets as widgets
from IPython.display import display
```

2. Data Loading and Preprocessing

The step will load the CSV dataset into Python for processing.

Code

```
df_songsDB = pd.read_csv('song_dataset.csv')
df_songsDB
```

Result. Here is a snapshot of the dataset.

	user	song	play_count	title	release	artist_name	year
0	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOAKIMP12A8C130995	1	The Cove	Thicker Than Water	Jack Johnson	0
1	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOAPDEY12A81C210A9	1	Nothing from Nothing	To Die For	Billy Preston	1974
2	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBBMDR12A8C13253B	2	Entre Dos Aguas	Flamenco Para Niños	Paco De Lucia	1976
3	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBFNSP12AF72A0E22	1	Under Cold Blue Stars	Under Cold Blue Stars	Josh Rouse	2002
4	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBFOVM12A58A7D494	1	Riot Radio (Soundtrack Version)	Nick & Norah's Infinite Playlist - Original Mo...	The Dead 60s	0
...
102622	21f4ac98aa1665bd42027ba12184a939ff435f59	SOKAKHH12AF72A5BAF	3	87	Hopeless Romantic	Bouncing Souls	1999
102623	21f4ac98aa1665bd42027ba12184a939ff435f59	SONPOXM12A8C1440C2	4	Space Olympics	Incredibad	The Lonely Island	2009
102624	21f4ac98aa1665bd42027ba12184a939ff435f59	SOPREHY12AB01815F9	8	I'm On A Boat	Incredibad	The Lonely Island / T-Pain	2009
102625	21f4ac98aa1665bd42027ba12184a939ff435f59	SOQXKUV12A6D4FB4C9	3	Amityville	The Marshall Mathers LP	Eminem / Bizarre	2000
102626	21f4ac98aa1665bd42027ba12184a939ff435f59	SOSJRJP12A6D4F826F	18	Master Of Puppets	Master Of Puppets	Metallica	1986

102627 rows × 7 columns

Result: The SVD model is trained and ready to predict user preferences.

3. Collaborative Filtering

Collaborative Filtering works by comparing users. If two users like the same songs, it assumes they might enjoy other songs that the other has listened to. This method uses a technique called **Singular Value Decomposition (SVD)** to predict how much a user might like a song.

Code

```
reader = Reader(rating_scale=(1, df_songsDB['play_count'].max()))
surpriseData = Dataset.load_from_df(df_songsDB[['user', 'song', 'play_count']], reader)

trainset, testset = train_test_split(surpriseData, test_size=0.2, random_state=20)

algo_SVD = SVD()
algo_SVD.fit(trainset)
```

Result: The SVD model is trained and ready to predict user preferences.

4. Content Based Filtering

Content Based Filtering focuses on song details, such as the artist's name, the song title, or the release info. It uses a technique called TF-IDF (Term Frequency-Inverse Document Frequency) to find similar songs

Code

```
def content_score_calculator(uesr_songs, unlistened_songs):
    df_songsDB['combined_features'] = (
        df_songsDB['artist_name'] + " " +
        df_songsDB['release'] + " " +
        df_songsDB['title']
    )

    uesr_songs_features = df_songsDB[df_songsDB['title'].isin(uesr_songs)][['combined_features']]
    unlistened_song_features = df_songsDB[df_songsDB['song'].isin(unlistened_songs)][['combined_features']]

    tfidf = TfidfVectorizer()
    tfidf_matrix = tfidf.fit_transform(df_songsDB['combined_features'])

    selected_matrix = tfidf.transform(uesr_songs_features)
    unlistened_matrix = tfidf.transform(unlistened_song_features)
    similarity_scores = cosine_similarity(selected_matrix, unlistened_matrix)

    avg_similarity = similarity_scores.mean(axis=0)

    return dict(zip(unlistened_songs, avg_similarity))
```

Result: The system identifies songs with similar features. For example If User 1 listens to "Song A" by "Artist X," it might recommend another song by the same artist.

7. The Hybrid Engine.

The hybrid approach combines the strengths of Collaborative Filtering and Content Based Filtering. It uses a parameter called alpha to decide how much weight to give each method. For example, if $\alpha = 0.5$, both methods contribute equally

Code

```
def hybridRecommendationEngine(user_id):
    alpha=0.5
    user_songs=list(df_songsDB[df_songsDB['user']==user_id]['title'].unique())
    listened_songs = df_songsDB[df_songsDB['user'] == user_id]['song'].unique()
    all_songs = df_songsDB['song'].unique()
    unlistened_songs = set(all_songs) - set(listened_songs)

    cf_scores = collaborative_score_calculator(user_id, unlistened_songs)
    content_scores = content_score_calculator(user_songs, unlistened_songs)

    final_scores = {}
    for song_id in unlistened_songs:
        cf_score = cf_scores.get(song_id, 0)
        content_score = content_scores.get(song_id, 0)
        final_scores[song_id] = alpha * cf_score + (1 - alpha) * content_score

    scores = list(final_scores.values())
    min_score = min(scores) if scores else 0
    max_score = max(scores) if scores else 1

    if max_score > min_score:
        normalized_scores = {
            song_id: (score - min_score) / (max_score - min_score)
            for song_id, score in final_scores.items()
        }
    else:
        normalized_scores = {song_id: 0.5 for song_id in final_scores}

    sorted_songs = sorted(normalized_scores.items(), key=lambda x: x[1], reverse=True)
    recommended_song_ids = [song_id for song_id, _ in sorted_songs[:10]]

    recommended_songs = (
        pd.DataFrame(recommended_song_ids, columns=['song'])
        .merge(df_songsDB[['song', 'title', 'release', 'artist_name', 'year']], on='song', how='left')
    )
    return recommended_songs
```

Output: The recommended songs are shown in the next step.

8. User Interface and interaction.

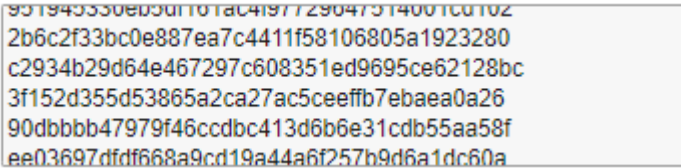
The system includes an interactive user selection interface for the user to select their ID and click a button to get recommendations. The code includes interactive elements using ipywidgets for user selection and displaying the recommendations.

```
user_select = widgets.Select(
    options=df_songsDB.user.unique(),
    description='Select User:',
    disabled=False,
    layout=widgets.Layout(width='auto', height='auto'),
)
user_select.style.description_width = 'initial'
display(user_select)
```

Result of the User Selection

When the widget is executed, it displays a dropdown menu with all the unique user IDs from the dataset. A screenshot of the interface is shown below:

Result

Select User: 

This allows the system to dynamically adjust recommendations based on the user selected from the dropdown.

9. Get Recommendations Interface Step

The Get Recommendations interface is the next step after selecting a user ID. It processes the selected user and generates personalized song recommendations. The feature combines the dropdown menu (for user selection) with a button to trigger the recommendation engine.

This step focuses on how recommendations are generated and displayed after the user has been selected.

```
recommend_button = widgets.Button(  
    description='Get Recommendations',  
    disabled=False,  
    button_style='success',  
    tooltip='Click to get recommendations',  
    icon='check'  
)  
display(recommend_button)  
  
def on_recommend_button_click(b):  
    user_id = user_select.value  
    recommendations = hybridRecommendationEngine(user_id)  
    print("Recommended Songs:")  
    display(recommendations)  
  
recommend_button.on_click(on_recommend_button_click)
```

Result

When the green button is clicked after the user selects an ID, they click the green button labeled **Get Recommendations**. The button triggers a function that takes the selected user ID, processes it using the hybrid recommendation engine, and displays the recommended songs in a clear table format

Get Recommendation...					
Recommended Songs:					
	song	title	release	artist_name	year
0	SOSJLSI12A8C13FB69	Your Rocky Spine	Penny Candy_ Vol. 2	Great Lake Swimmers	2007
1	SOMAKIT12A58A7E292	Bodies	Sinner	Drowning Pool	2001
2	SOQBUFQ12A6D4F7F4C	MIC (Speak Life Album Version)	Speak Life	Sev Statik	0
3	SOAUWYT12A81C206F1	Undo	Vespertine Live	Björk	2001
4	SOEKNHF12A3F1E9B8E	Times	Over And Underneath	Tenth Avenue North	2008
5	SOEGIYH12A6D4FC0E3	Horn Concerto No. 4 in E flat K495: II. Romanc...	Mozart - Eine kleine Nachtmusik	Barry Tuckwell/Academy of St Martin-in-the-Fie...	0
6	SOILVXC12AB017C10D	Batang-bata Ka Pa	Tala-Arawan	Sugar Free	0
7	SOSX LTC12AF72A7F54	Revelry	Only By The Night	Kings Of Leon	2008
8	SOZWBVE12AAF3B515C	Golden Mummy Golden Bird	Desperate Living	Horse The Band	2009
9	SOBONKR12A58A7AE0	You're The One	If There Was A Way	Dwight Yoakam	1990
Recommended Songs:					
	song	title	release	artist_name	year
0	SOSJLSI12A8C13FB69	Your Rocky Spine	Penny Candy_ Vol. 2	Great Lake Swimmers	2007
1	SOMAKIT12A58A7E292	Bodies	Sinner	Drowning Pool	2001
2	SOQBUFQ12A6D4F7F4C	MIC (Speak Life Album Version)	Speak Life	Sev Statik	0
3	SOAUWYT12A81C206F1	Undo	Vespertine Live	Björk	2001
4	SOEKNHF12A3F1E9B8E	Times	Over And Underneath	Tenth Avenue North	2008
5	SOEGIYH12A6D4FC0E3	Horn Concerto No. 4 in E flat K495: II. Romanc...	Mozart - Eine kleine Nachtmusik	Barry Tuckwell/Academy of St Martin-in-the-Fie...	0

Output. The **Get Recommendations Interface** turns the selected user ID into a meaningful output by generating and displaying personalized song suggestions. It is the final step in the interaction process, completing the recommendation cycle

Why This Design Was Chosen

This design was chosen because it's simple and works well with the data. For Collaborative Filtering (CF) the system use SVD because it's easy to implement and effective for predicting user preferences. While ALS could be faster but it's better for larger datasets, SVD is sufficient for the current data.

Content Based Filtering (CBF) uses TF IDF and a similarity score to recommend songs based on details like the artist and title. Advanced techniques such as analyzing audio features like tempo or pitch could enhance recommendations but the required audio data was not available.

The hybrid approach combines CF and CBF by averaging their scores this keeps system simple to ensure balanced and diverse recommendations. Advanced methods like Random Forest could improve this further but they would require more resources and data.

Possible Alternatives

For future improvements ALS could replace SVD for faster performance with bigger datasets and deep learning models could better capture complex user preferences. CBF could be improved by analyzing audio features or using tools like Word2Vec for better text comparisons. The hybrid system could use smarter models like Random Forest or rules to decide when to use CF or CBF.

Conclusion: The current design strikes a balance between simplicity, performance, and scalability, making it ideal for the given dataset and system goals.