




UNIwersYTET EKONOMICZNY WE WROCLAWIU

REKOMENDER MUZYCZNY

APLIKACJA WEBOWA WYKORZYSTUJĄCA SPOTIFY API

MIKOŁAJ STOJEK (190000), FILIP SMOLARCZYK (182331), ADAM
BASALSKI (190001), BEATA MACIEJEWSKA (190013)



Wstęp do problematyki

W dzisiejszym dynamicznie rozwijającym się świecie technologii, aplikacje oparte na rekomendacjach odgrywają kluczową rolę w personalizacji doświadczeń użytkowników. Coraz więcej firm korzysta z algorytmów rekomendacyjnych, aby dostarczać spersonalizowane treści, produkty i usługi. Jednym z popularnych przykładów są systemy rekomendacji muzycznych, takie jak Spotify, które umożliwiają użytkownikom odkrywanie nowej muzyki na podstawie ich wcześniejszych preferencji.

Celem niniejszego projektu jest stworzenie prostej aplikacji rekomendacyjnej, która wykorzystuje API Spotify do generowania spersonalizowanych playlist muzycznych. Projekt ten ma na celu zbadanie i zrozumienie mechanizmów działania systemów rekomendacyjnych, a także dostarczenie funkcjonalnego narzędzia, które będzie mogło być wykorzystywane w różnych kontekstach aplikacji muzycznych.

Aplikacje rekomendacyjne stają się coraz bardziej zaawansowane, korzystając z różnorodnych technik, takich jak filtrowanie treści, filtrowanie współpracujące oraz modele hybrydowe, które łączą różne podejścia w celu zwiększenia dokładności rekomendacji. W kontekście Spotify, algorytmy rekomendacyjne analizują dane użytkowników, takie jak odtwarzane utwory, ulubione playlisty i interakcje społeczne, aby dostarczać spersonalizowane sugestie.

W ramach tego projektu skupimy się na następujących aspektach:

1. **Integracja z API Spotify:** Wykorzystanie kluczowych funkcji API Spotify, takich jak uzyskiwanie danych o utworach, tworzenie i modyfikowanie playlist oraz zarządzanie autoryzacją użytkowników.
2. **Algorytmy rekomendacyjne:** Implementacja podstawowych algorytmów rekomendacyjnych, takich jak filtrowanie współpracujące, aby generować spersonalizowane playlisty.
3. **Analiza danych:** Zbieranie i analiza danych użytkowników, aby lepiej zrozumieć ich preferencje muzyczne i dostosować rekomendacje.
4. **Testowanie i ocena:** Przeprowadzenie testów funkcjonalnych aplikacji, aby ocenić jej skuteczność i dokładność rekomendacji.

Projekt ten jest odpowiedzią na rosnące zapotrzebowanie na narzędzia, które potrafią dostarczać użytkownikom wartościowe rekomendacje w sposób automatyczny, efektywny i precyzyjny. W niniejszym dokumencie zostaną przedstawione kluczowe aspekty problematyki związanej z rekomendacjami muzycznymi, rozwinięcie problemu, zastosowana metodologia, wyniki działania aplikacji oraz dyskusja i przyszłe możliwości rozwoju.

Rozwinięcie problemu

Problematyka rekomendacji muzycznych

Rekomendacje muzyczne są kluczowym elementem dla platform strumieniowych, takich jak Spotify, które dążą do zwiększenia zaangażowania użytkowników poprzez dostarczanie spersonalizowanych treści. Problem rekomendacji polega na przewidywaniu, które utwory mogą spodobać się użytkownikowi na podstawie analizy jego wcześniejszych działań i preferencji. Jest to wyzwanie z kilku powodów:

1. **Różnorodność muzyczna:** Muzyka jest bardzo zróżnicowaną formą sztuki, z wieloma gatunkami, stylami i artystami. System rekomendacyjny musi uwzględniać tę różnorodność, aby dostarczać trafne i zróżnicowane sugestie.
2. **Skalowalność:** Platformy takie jak Spotify obsługują miliony użytkowników i posiadają ogromne bazy danych utworów muzycznych. Algorytmy rekomendacyjne muszą być skalowalne, aby efektywnie przetwarzać i analizować te dane w czasie rzeczywistym.
3. **Personalizacja:** Każdy użytkownik ma unikalne preferencje muzyczne, które mogą się zmieniać w czasie. System rekomendacyjny musi być w stanie dynamicznie dostosowywać swoje sugestie, aby odzwierciedlać aktualne gusta użytkownika.

Specyfika projektu

W kontekście niniejszego projektu, istotnym jest podkreślenie, że rekomender nie bazuje na historii słuchania użytkownika, lecz na artystach, utworach lub gatunkach podanych przez użytkownika. Jest to istotne, ponieważ wprowadza dodatkowe wyzwania i możliwości w projektowaniu algorytmu rekomendacyjnego.

Metody rekomendacji

Istnieje kilka głównych podejść do problemu rekomendacji muzycznych:

1. **Filtrowanie treści (Content-based filtering):** To podejście polega na analizie cech utworów muzycznych (takich jak gatunek, tempo, nastroje) i rekomendowaniu podobnych utworów na podstawie preferencji użytkownika. W kontekście naszego projektu, filtrowanie treści będzie kluczowe, gdyż użytkownik sam podaje preferowane cechy, takie jak konkretni artyści, utwory czy gatunki muzyczne.
2. **Filtrowanie współpracujące (Collaborative filtering):** W tym podejściu rekomendacje są generowane na podstawie preferencji i zachowań innych użytkowników o podobnych gustach. W naszym projekcie, chociaż główny nacisk kładziemy na filtrowanie treści, możemy rozważyć elementy filtrowania współpracującego jako uzupełnienie.
3. **Modele hybrydowe (Hybrid models):** Te modele łączą różne podejścia, takie jak filtrowanie treści i filtrowanie współpracujące, aby zwiększyć dokładność i różnorodność rekomendacji. Modele hybrydowe mogą korzystać z zalet obu metod, jednocześnie minimalizując ich wady.

Wyzwania techniczne

Implementacja skutecznego systemu rekomendacyjnego napotyka na wiele wyzwań technicznych:

1. **Zbieranie i przetwarzanie danych:** System musi efektywnie zbierać dane o użytkownikach i utworach, a następnie je przetwarzać, aby były użyteczne dla algorytmów rekomendacyjnych. W naszym przypadku kluczowe będzie prawidłowe przetwarzanie danych wejściowych dostarczanych przez użytkownika.
2. **Optymalizacja algorytmów:** Algorytmy rekomendacyjne muszą być zoptymalizowane pod kątem wydajności i dokładności, co wymaga zaawansowanej wiedzy z zakresu uczenia maszynowego i analizy danych.
3. **Interfejs użytkownika:** Ostateczny sukces systemu rekomendacyjnego zależy od tego, jak dobrze rekomendacje są prezentowane użytkownikom. Interfejs musi być intuicyjny i umożliwiać łatwy dostęp do rekomendowanych utworów.

Znaczenie projektu

Projekt ten stanowi ciekawostkę dla osób zainteresowanych nowoczesnymi technologiami rekomendacyjnymi i ich praktycznymi zastosowaniami. Choć nie ma on ogromnego znaczenia dla szerokiego grona odbiorców, może być wartościowym elementem edukacyjnym, pokazującym, jak działają systemy rekomendacyjne w kontekście muzyki.

Celem tego projektu jest nie tylko stworzenie funkcjonalnego narzędzia, ale również zdobycie wiedzy i doświadczenia w pracy z algorytmami rekomendacyjnymi oraz API Spotify. Dzięki temu projektowi, zrozumiemy, jak skutecznie tworzyć rekomendacje muzyczne, które są skalowalne, spersonalizowane i trafne. Dla osób zainteresowanych, projekt ten może być inspiracją do dalszego zgłębiania tematu i rozwoju własnych rozwiązań rekomendacyjnych.

Metodologia

Integracja z API Spotify

Projekt został zrealizowany przy użyciu API Spotify, które umożliwia dostęp do różnorodnych danych muzycznych oraz funkcji związanych z odtwarzaniem i zarządzaniem muzyką. Kluczowe kroki integracji obejmowały:

1. **Uzyskanie dostępu do API Spotify:** Konfiguracja dostępu do API Spotify poprzez utworzenie aplikacji w Spotify Developer Dashboard i uzyskanie niezbędnych kluczy API (CLIENT_ID, CLIENT_SECRET).
2. **Autoryzacja użytkownika:** Implementacja procesu autoryzacji użytkownika za pomocą protokołu OAuth, co pozwala na bezpieczne uzyskiwanie uprawnień do zarządzania playlistami użytkownika i odczytu jego danych.
3. **Pobieranie danych muzycznych:** Wykorzystanie dostępnych endpointów API Spotify do pobierania danych na temat artystów, utworów i gatunków muzycznych w celu generowania rekomendacji.

Biblioteka Spotipy

Aplikacja korzysta z biblioteki Spotipy, która jest prostym i skutecznym wrapperem dla API Spotify. Umożliwia ona łatwe wykonywanie zapytań do API oraz zarządzanie autoryzacją. Poniżej opisano kluczowe elementy użycia biblioteki Spotipy:

1. **Konfiguracja i autoryzacja:** Skrypt *simple_recommendation_app.py* zawiera konfigurację połączenia z API Spotify oraz implementację procesu autoryzacji użytkownika przy użyciu OAuth.

```
import spotipy
from spotipy.oauth2 import SpotifyOAuth

sp = spotipy.Spotify(auth_manager=SpotifyOAuth(
    client_id='c97059e9f08e41159b29064147e82ff5',
    client_secret='bed718ec384e4b77bf1b69b62d4287cc',
    redirect_uri='http://localhost:8000/callback',
    scope='playlist-modify-public playlist-modify-private'
))
```

2. **Pobieranie preferencji użytkownika:** Skrypt umożliwia użytkownikowi wprowadzenie preferencji, które są następnie przetwarzane w celu wyszukania odpowiednich danych w API Spotify.

```
def get_recommendations(preferences):
    results = sp.search(q=preferences, type='track,artist')
    return results
```

3. **Generowanie rekomendacji i tworzenie playlisty:** Na podstawie wyników wyszukiwania generowane są rekomendacje, które następnie są zapisywane jako nowa playlista na koncie użytkownika.

```
def create_playlist(user_id, playlist_name, track_ids):
    playlist = sp.user_playlist_create(user_id, playlist_name)
    sp.user_playlist_add_tracks(user_id, playlist['id'], track_ids)
```

Uruchamianie aplikacji na Flasku

Aplikacja została zbudowana przy użyciu frameworka Flask, który umożliwia łatwe tworzenie aplikacji webowych w Pythonie. Flask jest lekki i elastyczny, co sprawia, że jest idealnym wyborem dla tego projektu. Kluczowe elementy aplikacji obejmują:

1. **Konfiguracja Flask:** Skrypt *simple_recommendation_app.py* zawiera konfigurację i uruchomienie serwera Flask.

```
from flask import Flask, request, redirect, url_for, session
app = Flask(__name__)
app.secret_key = 'some_random_secret_key'

@app.route('/')
def index():
    return 'Hello, this is the recommendation app!'
```

2. **Autoryzacja użytkownika:** Proces autoryzacji użytkownika jest zintegrowany z Flask, umożliwiając bezpieczne logowanie i zarządzanie sesjami.

```
@app.route('/login')
def login():
    sp_oauth = SpotifyOAuth()
    auth_url = sp_oauth.get_authorize_url()
    return redirect(auth_url)

@app.route('/callback')
def callback():
    sp_oauth = SpotifyOAuth()
    session.clear()
    code = request.args.get('code')
    token_info = sp_oauth.get_access_token(code)
    session['token_info'] = token_info
    return redirect(url_for('index'))
```

3. **Interakcja z użytkownikiem:** Flask obsługuje interfejs użytkownika, umożliwiając wprowadzenie preferencji muzycznych i wyświetlanie wyników rekomendacji.

```
@app.route('/recommend', methods=['POST'])
def recommend():
    preferences = request.form['preferences']
    recommendations = get_recommendations(preferences)
    return render_template('recommendations.html',
                           recommendations=recommendations)
```

Analiza danych

Aby zapewnić trafność rekomendacji, projekt zakłada analizę danych muzycznych pod kątem ich cech charakterystycznych, takich jak gatunek, tempo, nastroje, popularność itp. Dzięki temu możliwe jest bardziej precyzyjne dopasowanie rekomendacji do preferencji użytkownika.

Testowanie i ocena

Projekt został przetestowany pod kątem funkcjonalności i skuteczności rekomendacji. Testy obejmowały:

1. Sprawdzenie poprawności autoryzacji i połączenia z API Spotify.
2. Weryfikacja trafności generowanych rekomendacji na podstawie różnych zestawów preferencji użytkownika.
3. Ocena intuicyjności i użyteczności interfejsu użytkownika.

Projekt ten jest interesującą ciekawostką dla osób zainteresowanych nowoczesnymi technologiami rekomendacyjnymi i stanowi podstawę do dalszych badań i rozwoju w tej dziedzinie.

Wyniki

Tworzenie playlisty

W wyniku działania aplikacji rekomendacyjnej, na koncie użytkownika tworzy się nowa playlista o ustalonym tytule. W playliście znajduje się 10 utworów wybranych na podstawie preferencji użytkownika. Proces tworzenia playlisty przebiega następująco:

1. **Wprowadzenie danych przez użytkownika:** Użytkownik wprowadza preferencje muzyczne poprzez podanie linku zawierającego ID wykonawcy, gatunku lub utworu.
2. **Generowanie rekomendacji:** Aplikacja przetwarza wprowadzone preferencje i wykorzystuje API Spotify, aby znaleźć utwory pasujące do podanych kryteriów. Na podstawie wyników wyszukiwania generowana jest lista rekomendowanych utworów.
3. **Tworzenie playlisty:** Na koncie użytkownika tworzy się nowa playlista z ustalonym w kodzie tytułem, a następnie dodawane są do niej rekomendowane utwory. Każda playlista zawiera 10 utworów.

Jak działa aplikacja

Aplikacja została zaprojektowana jako proste narzędzie webowe, które umożliwia użytkownikowi generowanie spersonalizowanych playlist na podstawie podanych preferencji.

Aplikacja korzysta z frameworka Flask oraz biblioteki Spotipy, co pozwala na łatwą integrację z API Spotify.

Instrukcja uruchomienia aplikacji (How To)

Krok 1: Konfiguracja środowiska

1. **Zainstaluj wymagane biblioteki:** Upewnij się, że masz zainstalowane biblioteki Flask i Spotipy. Możesz je zainstalować za pomocą pip:

```
pip install Flask spotipy
```

2. **Utwórz aplikację Spotify:** Zaloguj się na Spotify Developer Dashboard i utwórz nową aplikację. Zanotuj swój *CLIENT_ID* i *CLIENT_SECRET*.
3. **Skonfiguruj plik konfiguracyjny:** Utwórz plik *config.yaml* z następującą zawartością, zastępując wartości własnymi danymi:

```
spotify:
  CLIENT_ID: 'your_client_id'
  CLIENT_SECRET: 'your_client_secret'
  USER: 'your_spotify_user_id'
  REDIRECT_URL: 'http://localhost:8000/callback'
  SCOPE: 'playlist-modify-public playlist-modify-private'
  PLAYLIST: 'My Recommended Playlist'
```

Krok 2: Implementacja kodu

1. **Skrypt aplikacji:** Utwórz plik *simple_recommendation_app.py* i wklej do niego poniższy kod:

```
import spotipy
from spotipy.oauth2 import SpotifyOAuth
from flask import Flask, request, redirect, url_for, session,
render_template

app = Flask(__name__)
app.secret_key = 'some_random_secret_key'

sp = spotipy.Spotify(auth_manager=SpotifyOAuth(
    client_id='your_client_id',
    client_secret='your_client_secret',
    redirect_uri='http://localhost:8000/callback',
    scope='playlist-modify-public playlist-modify-private'
))

@app.route('/')
def index():
```

```

        return render_template('index.html')

@app.route('/login')
def login():
    sp_oauth = SpotifyOAuth()
    auth_url = sp_oauth.get_authorize_url()
    return redirect(auth_url)

@app.route('/callback')
def callback():
    sp_oauth = SpotifyOAuth()
    session.clear()
    code = request.args.get('code')
    token_info = sp_oauth.get_access_token(code)
    session['token_info'] = token_info
    return redirect(url_for('index'))

def get_recommendations(preferences):
    results = sp.search(q=preferences, type='track,artist')
    tracks = results['tracks']['items']
    track_ids = [track['id'] for track in tracks[:10]]
    return track_ids

def create_playlist(user_id, playlist_name, track_ids):
    playlist = sp.user_playlist_create(user_id, playlist_name)
    sp.user_playlist_add_tracks(user_id, playlist['id'], track_ids)

@app.route('/recommend/<preference>')
def recommend(preference):
    track_ids = get_recommendations(preference)
    user_id = session['token_info']['id']
    create_playlist(user_id, 'My Recommended Playlist', track_ids)
    return 'Playlist created successfully!'

if __name__ == '__main__':
    app.run(debug=True)

```

2. **Szablon HTML:** Utwórz folder templates i w nim *plik index.html* z następującą zawartością:

```

<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Recommendation App</title>
</head>
<body>
    <h1>Welcome to the Recommendation App</h1>
    <form action="/login" method="get">
        <button type="submit">Login to Spotify</button>
    </form>
</body>
</html>

```

Krok 3: Uruchomienie aplikacji

1. **Uruchom serwer Flask:** W terminalu przejdź do katalogu z plikiem *simple_recommendation_app.py* i uruchom aplikację.
2. **Otwórz przeglądarkę:** Przejdź do *http://localhost:8000* w przeglądarce internetowej, aby uzyskać dostęp do aplikacji.
3. **Zaloguj się do Spotify:** Kliknij na link logowania, aby autoryzować aplikację za pomocą swojego konta Spotify.
4. **Wprowadź preferencje:** Po zalogowaniu, użytkownik powinien wprowadzić link w formacie *http://localhost:8000/recommend/<id_wykonawcy_gatunku_utworu>*, gdzie *<id_wykonawcy_gatunku_utworu>* jest ID wykonawcy, gatunku lub utworu.
5. **Tworzenie playlisty:** Aplikacja przetworzy podane ID, wygeneruje rekomendacje i utworzy nową playlistę z 10 utworami na koncie użytkownika Spotify.

Projekt ten jest interesującą ciekawostką dla osób zainteresowanych nowoczesnymi technologiami rekomendacyjnymi i stanowi podstawę do dalszych badań i rozwoju w tej dziedzinie.

Spotify Recommendation App

Spotify Recommendation App

This is a Flask application that provides music recommendations using the Spotify API.

Setup:

1. Clone the repository:

```
```bash
git clone https://github.com/your-username/spotify-recommendation-app.git
cd spotify-recommendation-app
```
```

2. Create a virtual environment and activate it:

```
```bash
python -m venv venv
```
```

On macOS and Linux:

```
```bash
source venv/bin/activate
```
```

On Windows:

```
```bash
venv\Scripts\activate
```
```

3. Install the dependencies:

```
```bash
pip install -r requirements.txt
```
```

4. Set up your config.yaml file in the config folder with the following content:

```
```yaml
spotify:
 CLIENT_ID: 'your_client_id'
 CLIENT_SECRET: 'your_client_secret'
 USER: 'your_user_id'
 REDIRECT_URL: 'http://localhost:8000/callback'
 SCOPE: 'playlist-modify-public playlist-modify-private'
 PLAYLIST: 'your_playlist_id'
```
```

Note:

- Make sure that the config.yaml file is in the same folder as the script ``simple_recommendation_app.py`` or adjust the script to point to the correct path of ``config.yaml``.
- Do not change the ``SCOPE`` value in the config.yaml file as it must be set as specified.
- If ``localhost:8000`` does not work, you may need to change it to ``localhost:5000``.

Running the app:

1. To run the Flask app, use the following command:

```
```bash
python app/simple_recommendation_app.py
```
```

2. Make sure you are logged into Spotify in your web browser.

3. Open your web browser and navigate to:

```
```http
```

```
http://127.0.0.1:8000/get_music_recommendations?seed_artists=artist_id
```

```
```
```

to get music recommendations.

You can also use `seed_tracks` and `seed_genres` as parameters to get recommendations based on tracks and genres. Example:

```
```http
```

```
http://127.0.0.1:8000/get_music_recommendations?seed_artists=artist_id&seed_tracks=track_id&seed_genres=genre
```

```
```
```

Folder structure:

spotify-recommendation-app/

```
|— app/
|   |— simple_recommendation_app.py
|— config/
|   |— config.yaml
|— requirements.txt
|— README.txt
```

Changing Playlist Name and Description

To change the name and description of a playlist, use the `user_playlist_change_details` method from the `spotipy` library. Here's an example of how to do it:

1. Import the `spotipy` library and other required modules:

```
```python
import spotipy
from spotipy.oauth2 import SpotifyOAuth
```
```

2. Initialize the Spotify client with the necessary permissions:

```
```python
sp = spotipy.Spotify(auth_manager=SpotifyOAuth(client_id='your_client_id',
 client_secret='your_client_secret',
 redirect_uri='your_redirect_uri',
 scope='playlist-modify-public playlist-modify-private'))
```
```

3. Change the name and description of the playlist:

```
```python
```

```
playlist_id = 'your_playlist_id'
sp.user_playlist_change_details(user='your_user_id', playlist_id=playlist_id,
name='New Playlist Name', description='New Playlist Description')
` ``
```

Replace ``your\_client\_id``, ``your\_client\_secret``, ``your\_redirect\_uri``, ``your\_user\_id``, and ``your\_playlist\_id`` with your actual Spotify API credentials, user ID, and playlist ID.

This method allows you to update the name and description of an existing playlist.

## ## Required Libraries and Their Versions

The following libraries are required to run this application. Make sure they are included in your ``requirements.txt`` file with the specified versions:

- Flask==3.0.3
- spotipy==2.23.0
- PyYAML==6.0.1
- requests==2.31.0