Fundamentals and Applications of AI
Final Project

Lorenzo D'Amico

June 2024

0.1 Title

Personal assistant for song prediction.

0.2 Abstract

The project focuses on the development of software that, given as input the title of a song or a set of song titles, produces as output a finite set of song titles similar to the input song(s). The software is implemented in Python using specific libraries for data management, analysis, visualization, and user interface handling (libraries mentioned in the Materials and Methods section 0.4). The project partially includes code developed by Vatsal Mavani in the project *Music Recommendation System using Spotify Dataset* (https://www.kaggle.com/code/vatsalmavani/music-recommendation-system-using-spotify-dataset/notebook).

0.3 Introduction

The software consists of several components that sequentially perform different tasks in order to achieve results that are as consistent as possible with the project objectives. The two main components are a Constraint Satisfaction Problem (CSP) and a Machine Learning component. In summary, by using a dataset of songs, the goal is to obtain a set of recommendations starting from one or more input songs. From this initial set, the final solution set is derived by passing through a CSP, which filters out songs that do not satisfy the defined constraints.

0.4 Materials and Methods

The project is implemented in Python and is structured as follows:

- **classes/CSP folder**: contains files for creating a CSP and solving the problem using different algorithms.
- **dataset folder**: contains datasets used for data analysis.
- **templates folder**: contains HTML files used by Flask to build the graphical user interface.
- **csp class (classes/csp.py)**: responsible for creating the CSP and implementing project-specific functions.

- **datacluster class (classes/datacluster.py)**: creates objects used to train models for data analysis and visualization.
- **ML class (classes/ml.py)**: collects input objects and returns song recommendations.
- **Recommender class (classes/recommender.py)**: manages the interaction between input data and the dataset.
- **Configuration class (classes/configuration.py)**: manages the connection to the Spotify API and parses the `config.txt` file containing API keys.
- **index.py file**: main entry point of the application. The application can be launched with:

```
flask --app index run --debugger
```

0.5 Experiments and Results

To run the experiment, some prerequisites are required. The code must be placed in a single directory with an arbitrary name. A configuration file must be created containing a JSON object with `client_id` and `client_secret`:

```
{"client_id" : "xxx", "client_secret" : "yyy"}
```

0.5.1 Configuration File Creation

The configuration file can be created by following these steps:

1. Access Spotify for Developers and log in.
2. Create a new application.
3. Fill in the application details.
4. Obtain the Client ID and Client Secret.
5. Configure the application permissions.
6. Use the credentials to create the configuration file.

After launching the application, a browser page opens with a search bar that allows the user to search for a song by title. Once the correct song is selected, the user may add additional songs or request similar song recommendations. The machine learning and CSP components then select similar songs and filter out those that do not satisfy the constraints.

0.5.2 Data Analysis Components

The machine learning model is based on a pipeline composed of **StandardScaler** and **KMeans**, which normalize numerical features and group songs with similar characteristics. At the end of the process, a numerical matrix representing songs is obtained. Similar songs are identified by searching for nearby vectors in this matrix.

The CSP constraint checks whether a recommended song already appears in the user's playlists, avoiding recommendations of already-known songs. Discarded songs are visually marked with a red border.

0.5.3 Data Analysis

Data analysis is performed only if the configuration file contains the key `"view": "False"`.

0.5.4 Recommendation Function Explanation

The recommendation system standardizes numerical features before applying cosine distance to measure similarity between songs. Only a fixed number of recommendations (10) is returned.

0.6 Conclusion

The developed system provides a set of songs similar to a given input. The output remains consistent across identical experiments since predictions rely on deterministic mathematical computations over a fixed dataset. The recommendation quality is therefore limited by the size and diversity of the dataset.