https://github.com/keodai/cs310-project

keodai / cs310-project Private

Applying Machine Learning and Audio Analysis Techniques to Music Recommendation

Edit

Add topics

57 commits | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

keodai Updated README — Latest commit fbdb9fe a minute ago

.idea	Modified for lower resolution (720p) displays	2 months ago
data	Timing	3 months ago
static	Modified for lower resolution (720p) displays	2 months ago
templates	Updates to fv, ml and link to survey	2 months ago
README.md	Updated README	a minute ago
dirdiff	Refactored for submission	3 hours ago
fma_preprocessing.py	Refactored for submission	3 hours ago
gtzan_preprocessing.py	Refactored for submission	3 hours ago
initialise.py	Refactored for submission	3 hours ago
mfcc_plot.py	Refactored for submission	3 hours ago
multi_logging.py	Refactored for submission	3 hours ago
paths.py	Refactored for submission	3 hours ago
recommender.py	Refactored for submission	3 hours ago
results-screen.png	Refactored for submission	3 hours ago
server.py	Refactored for submission	3 hours ago
setup.py	Refactored for submission	3 hours ago
song.py	Refactored for submission	3 hours ago
sound_recording.py	Refactored for submission	3 hours ago
tinytag.py	Added metadata extraction for genre, title and artist information usi...	6 months ago

README.md

Applying Machine Learning and Audio Analysis Techniques to Music Recommendation

Module: CS310 Computer Science Project
Author: Matthew Penny [1509819]
Supervisor: Victor Sanchez
Department: Department of Computer Science, University of Warwick, United Kingdom
Year: 2018

This document provides installation and usage instructions for the music recommendation system created during this project. More information on the background research, development and system evaluation undertaken during the creation of this system can be found in the accompanying report, along with proposed areas of future work.

Prerequisites:

Software Requirements

The application is created in Python 2.7.14. In addition to the standard Python installation, the following libraries are required to be installed prior to using the system, many of which are included in the Anaconda distribution:

- scikit-learn
- NumPy
- LibROSA*
- Matplotlib
- SciPy
- Flask*
- PyAudio

* not included in Anaconda distribution.

These libraries may be installed in the Python environment using the `conda` (Anaconda) or `pip` (Python) package managers

FFmpeg is used to implement file format conversion, and may require additional codecs (libavcodec-extra-53) to be installed or enabled to handle MP3 files (for the GTZAN dataset). This can be performed by compiling FFmpeg manually, through Homebrew (macOS) or by the following command on Linux systems:

```
sudo apt-get install ffmpeg libavcodec-extra-53
```

System Requirements

For the server, the system hardware requirements, including memory and storage, will vary depending on the size of the training dataset used. A minimum of 8GB of memory is recommended, whilst approximately 70GB of available storage space is required. This storage capacity includes the dataset (MP3 and WAV formats), trained models and song data for a 10,000 song subset of the FMA dataset (30 second samples).

Users can access the system via a web browser, such as Google Chrome.

Installation & Basic Setup:

Firstly, save the project files to a new directory.

Next, the `project_audio_dir` variable needs to be updated to point to the new location of the audio within the application.

Once the required software libraries have been installed in the Python environment, the feature extraction and model training can be performed by running the **setup.py** script, e.g. by:

```
cd <path_to_project>
python setup.py
```

`<path_to_project>` should be replaced by the actual directory of the project code. This script performs feature extraction from the audio directory defined in **paths.py**, creating a `Song` object containing the data for each track, and trains the machine learning models using the data extracted from the training dataset. The song data and models are then saved to disk for later use.

Extracted song data and trained machine learning models are included on the provided USB, using the FMA dataset with restricted genres. However, the song data will still refer to the old file path unless re-trained, which will cause song previews to be unavailable. This subset of the FMA dataset is also included on the USB, although only in MP3 format due to the file size when converted to WAV. This allows for retraining using the same dataset, although conversion and feature extraction will also have to be performed to create the WAV files and update the file paths for song previews. Depending on the size of the dataset, feature extraction and model training may take several hours to complete.

The server can be started, including the loading of the required data files, by running **server.py**, e.g.:

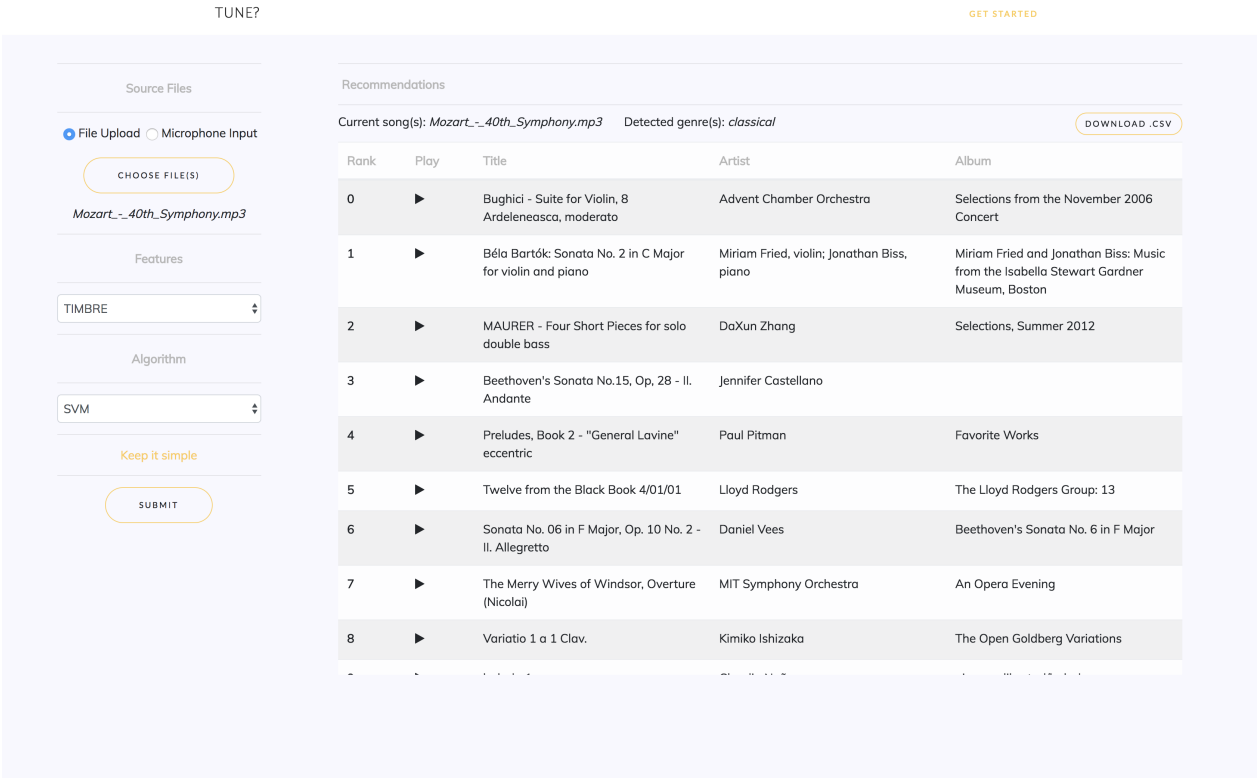
```
python server.py
```

Usage:

The application can be accessed once the server is running and the song data and model files have loaded. For localhost, the application can be accessed at IP: `127.0.0.1:5000` . If the application were to be deployed, the system administrator would perform the server setup steps detailed in *Installation and Basic Setup*, with users then able to access the hosted system in their web browser via a URL.

The user interface was designed to be simple to use. Recommendations can be retrieved by selecting the input method (file upload or microphone input). Advanced features, offering the option to select the feature vector format and machine learning model to use, can be displayed by clicking 'Under the hood'. A screen capture of this form and the table of returned results, providing information and previews for each song, can be seen below.

A video demo, showing the operation of the system is also included (**demo.mov**).



Tips:

- Preprocessing scripts are included to select valid training songs in a standardised format from both the FMA and GTZAN datasets.
- If additional feature vectors or models are added, these approaches need to be included in the dictionary created in **initialise.py**, as well as implementing their setup and recommendation behaviours.
- MFCC spectrograms can be created using the **mfcc_plot.py** script.
- Recommendation can be performed without the UI, by running **recommender.py** directly to obtain a list of textual information about the recommended songs, such as title, artist and path: `python recommender <path_to_music> <recommendation_mode> <vector_type> <data>`
- The directories used for audio data can be updated by modifying the paths in **paths.py**. If new directories are assigned, they must already exist before running the system.