matsubara
Case-based Recommendation for Music Playlists

Alexander Stradnic

Prof. Derek G. Bridge

University College Cork
Final Year Project, BSc. Computer Science
April 2023

Abstract

In this Final Year Project I present the use of Case-based Recommendation in creating music playlists, along with providing an implementation of this CBR system and examples of output. Playlists are sequences of songs arranged in a particular order. Using this information, context can be gained from previous playlists in order to build a new playlist from a given seed song or starting list. Patterns in previous playlists can be analysed and used to create an enjoyable and coherent listening experience. This paper mainly makes reference to the paper below [BP06]. Other research and ideas were drawn from [PA02] and [Smy07].

Declaration of Originality

In signing this declaration, you are conforming, in writing, that the submitted work is entirely your own original work, except where clearly attributed otherwise, and that it has not been submitted partly or wholly for any other educational award. I hereby declare that:

- this is all my own work, unless clearly indicated otherwise, with full and proper accreditation;
- with respect to my own work: none of it has been submitted at any educational institution contributing in any way to an educational award;
- with respect to another's work: all text, diagrams, code, or ideas, whether verbatim, paraphrased or otherwise modified or adapted, have been duly attributed to the source in a scholarly manner, whether from books, papers, lecture notes or any other student's work, whether published or unpublished, electronically or in print.

Signed: [illegible]
Date: 12th April 2023

Contents

Chapter 1

Introduction

1.1 Music Listening in the Pre-Industrial Era

Throughout most of history, the only way for a person to listen to a piece of music was to either attend a live performance, or perform themselves. While the written recording of music as notation has existed for hundreds of years, the majority of the population would not have the education and time to be able to read and reproduce this music, aside from the nobility who would have court musicians writing and performing music for them. W.A. Mozart, one of the most well-known composers today, had to travel the courts of Europe after leaving Salzburg due to low pay and the desire to compose more freely; he battled with financial insecurity throughout his life. Turlough O'Carolan was a famous blind harper who traveled across Ireland composing planxties for patrons.

This also generated a rift between the organised, written music of the elites with professional performers, compared to more accessible folk/traditional music of the populace which could be performed solo or by a group, and was passed down by ear.

1.2 Early Sound Reproduction

Attempts to reproduce the sound of music have been attempted and improved as technology has advanced, beginning with spring-wound machines in which a metal pin strikes a series of raised pitches on a rotating cylinder or disc. Music boxes are the most well-known, and reproduce a single melody line, though more complex contraptions automatically playing flutes, violins and keyboard instruments also existed, and player-pianos continue to exist, though less common and usually digitally operated.

The issue with these machines is that they would play exactly as inscribed, and with simplistic musicality. They tended to be large, expensive and limited to few tracks.

1.3 19-20th Century Innovations

With the development of early cylinder-based phonographs, followed by disc-based gramophones, allowed the widespread recording of music to occur. Gramophone recording would inscribe the sound vibrations onto a master record, from which copies would be printed onto shellac-based, and later polyvinyl-based discs. These discs were analog recordings of actual artist performances and improved in clarity with advances in materials and recording techniques. Acoustic recording gave way to electronic with microphones allowing more control and flexibility. Later on, magnetic tape recording was invented, and while initially only used in studio recording, eventually allowed for a small, portable and rugged music format to be developed in the form of the cassette, which started being produced from the 1960s and became more popular into the '70s.

Towards the end of the 20th Century, digital solutions to storing and playing music were explored, with the Compact Disc exploding onto the scene in 1982, effectively replacing any market share records had remaining, and competing with compact cassettes, replacing them for audio playback in the 2000s.

1.4 Music Ownership, Albums and The Computer Age

Until the rise of computers and the Internet, the way in which people listened to recorded music was starkly different. One would listen typically hear a song on the radio or live, and, if they wished to listen to it repeatedly, would visit the record store and purchase a copy of the song on vinyl, cassette, or CD. Cassettes also allowed for radio streams to be recorded.

They could also purchase a collection, typically an album on the medium of their choice, which would tend to be better value. In many styles of music such as pop and rock, collections of pieces or songs would be released together by an artist in albums. Music albums, initially inspired by photo albums, typically comprise a collection of tracks which tend to cover a theme or idea, and are intended by the artist to be listened sequentially. Even in other genres, music would be grouped and sold, length depending on the format and album. This allowed for longer uninterrupted listening before the listener would have to change the disc or cassette manually or with a suitable player.

The practice of “owning” your own music would continue to remain popular with many people until the advancement of Internet infrastructure and services. Apple pioneered the digital sale of music online, allowing songs to be purchased individually, to be played on MP3 players such as the iPod. This broke with the previous paradigm of selling and playing by the album. People now had the ability to create their own custom collections of songs called playlists. At this point, most people still consciously which songs to purchase, and gathered their own personal collections on their own. However, with the rise of Napster and later Spotify, this model of music ownership becamse less popular, as people wished to listen to more music without having to purchase every single song they wanted to listen to. Apple iTunes also started facilitating streaming, leading us to today, where most stream music from a platform such as YouTube, iTunes, or Spotify.

1.5 Modern Problems Require Modern Solutions

This rise in music streaming, along with the ability to purchase individual songs for a cheap price, has meant that listening to albums from start to finish drastically reduced in popularity. Many people listen to playlists or just the individual songs themselves as shown in these surveys carried out by Deezer[D320] and MusicBiz[LM16].

A major convenience offered by music streaming providers nowadays is the automatic generation of playlists, making it easier for users to listen to a variety of music without having to create their own playlists manually. This is now done using many methods ranging from popularity counts (seen in many services as the "Hot" or "Trending" pages) to personalised user-user and user-item algorithms, which take into account previous user listening activity and the preferences of "similar" users. Thus the problem: *Which methods generate the most interesting playlists for users?* In this paper, I explore various ways of generating music playlists, focusing on case-based recommendation.

Chapter 2

General Implementation Details

2.1 Data

The data was acquired from Spotify's Million Playlist Dataset. Processing was done on the dataset running in SQLite. The data was downloaded from Spotify's servers, structured as a set of JSON files. This was then converted into an SQLite database to improve lookup performance. Additional data was queried from Spotify's API.

2.2 Tables

- `tracks` – specific track info, song name, album name, etc.
- `track_features` – generic track features as measured by Spotify such as acousticness, danceability, speechiness etc.
- `artist_genres` – artists and their associated genres
- `playlists` – playlist information
- `playlist_tracks` – link between tracks and playlists

- `seq2_simple` – all sequences of $\langle song1, song2 \rangle$, usually aggregated in processing
- `similarities` – artist similarity rating based on their genres

2.3 Processing

The implementations are written in *Python*. This is due to the large amount of useful libraries available for the language such as *Numpy*, as well as the quick writing and readability.

2.4 Output

All algorithms described in this paper take in an input seed song or starting list of URIs, and output a generated list of track URIs.

`(spotify:track:1, spotify:track:2,..., spotify:track:`λ`)`

2.5 Limitations in Song Output

Due to the nature of Case-based Recommendation, it can only generate playlists containing songs that are already present in the seed dataset, thus limited to songs up to 2017. This is because it ranks songs rather than a generic set of features, and even an output comprising a list of feature sets would be limited to the dataset due to the inability of querying Spotify by features.

The similarity-based approach likewise only works for pre-computed similarities in the dataset.

2.6 Limitations in Personalisation

Personalising songs and genres based on a Spotify User Profile is possible by preferring user playlists and song features over generic playlists, it was beyond the scope of this project, requiring user authentication, processing and further development of the demonstration web-app. User-user recommendation and weighting is also theoretically possible, but would require manual input of other users as Spotify does not have a public API for getting "similar users".

Chapter 3

Similarity-based Recommendation

3.1 Background

words

Jaccard Similarity This was used to precompute the similarity between different songs based on their attributes. This can then be used to create a playlist based purely on track similarity.

$$Sim(\boldsymbol{s_1}, \boldsymbol{s_2}) = \frac{|\boldsymbol{s_1} \cap \boldsymbol{s_2}|}{|\boldsymbol{s_1} \cup \boldsymbol{s_2}|}$$

Chapter 4

Case-based Learning

Case-based Recommendation (CBR): The approach in which a playlist is generated from a seed song/shorter playlist using sequential patterns learned from a dataset of existing playlists. CBR is a method of recommendation which can be used in contexts in which a meaningful order or sequence to objects is desired or useful. There can also be a large variety of possible values (such as songs in this case).

4.1 Case Base

A set of playlists may then be filtered to remove meaningless and/or noisy playlists, forming the Case Base. The playlists are then analysed, using their song order as well as the songs themselves, in order to come up with new playlists to generate from an initial seed song.

4.2 Retrieval of Playlists

From this Case Base of playlists $\mathcal{C}$, two main values are computed for each playlist:

- The Attribute Variety: How varied and diverse the playlist is, based on song attributes

- The Coherence of the playlist: How relevant the sequences are to the seed song, as well as how many

After computing the Variety and Coherence for each playlist $p \in \mathcal{C}$, a rating function combines both and allows the playlists to be ordered.

$$\rho(p, s) = Var(p) \cdot Coh(p, s), \forall p \in \mathcal{C}$$

4.3 Reuse

After the k playlists have been ranked, they have to be then converted in some way to form a single playlist of a particular length λ. This is to be done using a process called Constructive Adaptation(references [PA02] in [BP06]), which comprises of two parts:

1. Hypothesis Generation(HG): How partial solutions are extended from the currently generated state of the playlist (either the seed song at the beginning or the highest ranked generated sequence chosen by HO afterwards)
2. Hypothesis Ordering(HO): Uses the ranking of sequences from $\mathcal{C}$ to rank the list of newly generated playlists after each iteration of HG

An ideal solution is generated by appending or prepending a song at a time to the seed song, decided but creating potential successor solutions using Hypothesis Generation and ranking using Hypothesis Ordering. This is continued recursively down the most ideal node. However, if no successor solution is generated at a particular level, then it is discarded from the list of ranked generated sequences, and the process continues from that point, until a playlist of length λ is generated.

4.4 Formulae Used

[BP06, pp. 5–10]

Variety The variety of each playlist in $\mathcal{C}$ is calculated based on the repetition of song features within a set distance. This is first calculated per attribute and then each attribute's variety is combined together:

$$V_a(p) = \prod_{i=1}^{n} \begin{cases} \frac{j-i}{\gamma_a} & \text{if } \exists j, i < j <= \min(n, i+\gamma_a) : a(s_i) = a(s_j) \\ & \wedge\ \forall k, i < k < j, a(s_i) \neq a(s_k) \\ 1 & \text{otherwise} \end{cases} \tag{4.1}$$

If no value of a is repeated within the safe distance γ_a, then $V_a(p)$ is 1. Otherwise it is some value between 0 and 1. Combined variety for all attributes $a \in \mathcal{A}$ for playlist p:

$$Var(p) = \prod_{a \in \mathcal{A}} V_a(p)$$

Coherence The coherence of each playlist in $\mathcal{C}$ is calculated based on how related it is to the input song. In order to calculate the Coherence for each playlist, the Relevance is first computed. The Relevance is determined by the amount of times the sequence q appears in the playlists that make up $\mathcal{C}$, adjusted for the biases of length and song popularity, and returns the degree in which q is a relevant pattern for song t.

$$Rel(q,t) = \phi(q) \cdot \frac{\alpha^{\theta - \Lambda(q)}}{\psi^{\beta}(q,t)}$$

The Coherence is then calculated from the sum of all these relevant patterns in a particular playlist.

$$Coh(p,s) = \sum_{q \in \Omega(s,p)} Rel(q,s)$$

Hypothesis Generation The nodes generated at each stage are sequences of songs from the k playlists, and consist of $T = (t_1, t_2, ..., t_n)$ songs. T' is a successor playlist, thus it contains all $t \in T$, along with an additional song u. This song is either prepended or appended to the list of songs. Thus $T' = \langle u + T\rangle$ or $\langle T + u\rangle$.

Considering the first case, $u+t_1$ should appear at least once in the set of k retrieved playlists, in order to be coherent. Similar is done in the second case, where t_n+u should appear at least once in k. Thus the set of all successors would be all combinations of these two rules done recursively.

Hypothesis Ordering The sequences are ordered by computing their Relevance and Variance, similar to how the k playlists were chosen.

$$H(T') = Rel(T', u) \cdot Var(T')$$

Where T' is the successor list and u is the added song. However, in order to determine whether future evolutions of T' will also be varied and coherent, a look-ahead parameter is added.

- L = 0: The above formula is applied as is
- L = 1: H is performed on all the successors of T', and the maximum value found is returned
- L = 2: H is performed on *all the successors* of all the successors of T' (depth 2), and the maximum value is returned

As with many such parameters, a trade-off between precision and performance has to be chosen, as computing cost increases exponentially with L.

Chapter 5

Testing

A variety of CBR systems will be tested with varying weightings of Coherence and Variance, along with the simpler pseudorandom and similarity-based algorithms. These will then be ranked out of 10 by survey, with top ranked lists passed back into the dataset.

Chapter 6

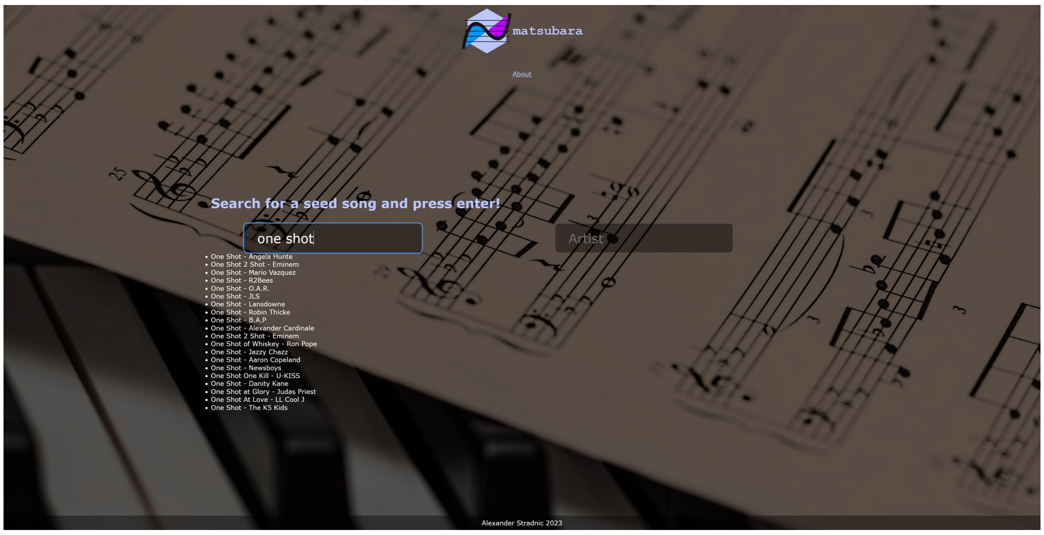
Beyond Testing

6.1 Additional Algorithms

Other styles of algorithms such as machine learning approaches could be explored and compared to the case-based configurations, taking the input playlists as the training set.

6.2 Web Application

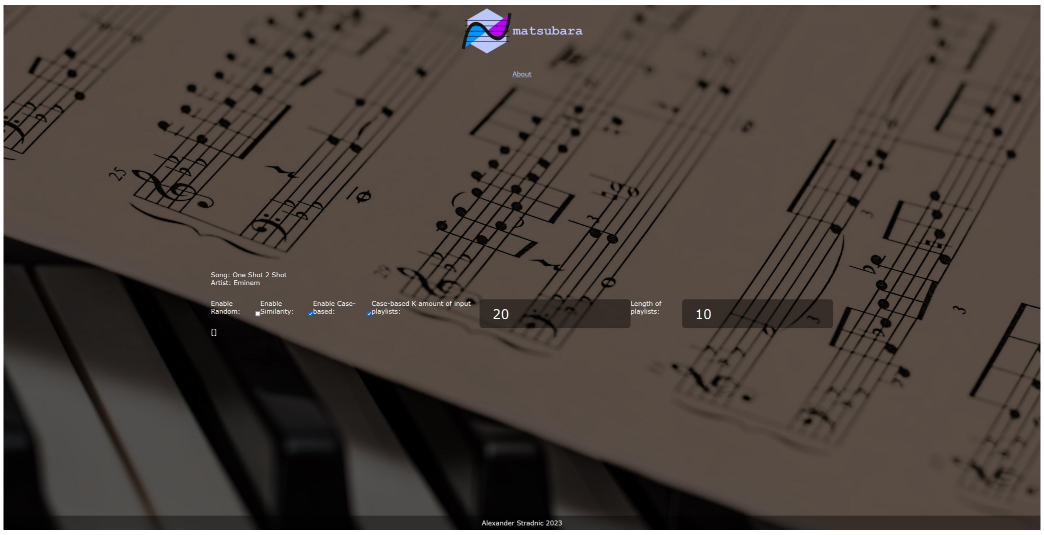
Current Implementation



- One Shot - Angela Hunte
- One Shot 2 Shot - Eminem
- One Shot - Mario Vazquez
- One Shot - R2Bees
- One Shot - O.A.R.
- One Shot - JLS
- One Shot - Lansdowne
- One Shot - Robin Thicke
- One Shot - B.A.P
- One Shot - Alexander Cardinale
- One Shot 2 Shot - Eminem
- One Shot of Whiskey - Ron Pope
- One Shot - Jazzy Chazz
- One Shot - Aaron Copeland
- One Shot - Newsboys
- One Shot One Kill - U-KISS
- One Shot - Danity Kane
- One Shot at Glory - Judas Priest
- One Shot At Love - LL Cool J
- One Shot - The K5 Kids

Alexander Stradnic 2023

A webapp was developed for the open day and showcases a potential application of the project. A user chooses their desired song from the list returned after querying a title and/or artist name and is then taken to a screen where they can configure different settings and options, such as the length of the desired playlist and K amount of input playlists to be factored in. More settings could also be added here. Multiple algorithms could be configured with different settings.



Song: One Shot 2 Shot
Artist: Eminem

Enable Random: ☐ Enable Similarity: ✓ Enable Case-based: ✓ Case-based K amount of input playlists: 20 Length of playlists: 10

[]

Alexander Stradnic 2023

Further Development

Unfortunately the main case-based algorithm was not able to output an answer in time due to connection timeout which I did not have the time to solve, but the premise still works and similarity-based playlists are able to be generated and returned to the client in the form of a list of playlists, each consisting of song URIs.

Song previews could be queried from Spotify on the website, and if a user likes any of the playlists, they could log into their Spotify account and add them to their profile.

6.3 Browser Extension

Instead of developing a separate website that users have to travel to separately from Spotify, an extension could be developed which a user would simply install, and would modify the Spotify web client, adding a tab, or simply as a popup like standard browser addons, allowing the user to do the same as above but be more convenient and even reading context from Spotify. The dataset and program could be downloaded with the addon, allowing local generation instead of pinging a server.

Bibliography

[PA02] Enric Plaza and Josep-Lluís Arcos. “Constructive Adaptation”. In: *Advances in Case-Based Reasoning*. Ed. by Susan Craw and Alun Preece. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 306–320. ISBN: 978-3-540-46119-7. DOI: 10.1007/3-540-46119-1.

[BP06] Claudio Baccigalupo and Enric Plaza. “Case-Based Sequential Ordering of Songs for Playlist Recommendation”. In: *Advances in Case-Based Reasoning*. Ed. by Thomas R. Roth-Berghofer, Mehmet H. Göker, and H. Altay Güvenir. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 286–300. ISBN: 978-3-540-36846-5. DOI: 10.1007/11805816.

[Smy07] Barry Smyth. “Case-Based Recommendation”. In: vol. 4321. Jan. 2007, pp. 342–376. ISBN: 978-3-540-72078-2. DOI: 10.1007/978-3-540-72079-9_11.

[LM16] LOOP and MusicBiz. *Playlists Overtake Albums in Listenership, Says LOOP Study*. 2016. URL: https://musicbiz.org/news/playlists-overtake-albums-listenership-says-loop-study/.

[D320] Deezer and 3Gem. *Over half admit to listening to less albums in last five years*. 2020. URL: https://www.deezer-blog.com/press/over-half-admit-to-listening-to-less-albums-in-last-five-years/.

Reference Links

- Spotify Million Playlist Dataset – https://engineering.atspotify.com/2018/05/introducing-the-million-playlist-dataset-and-recsys-challenge-2018/
- Spotify Web API Docs – https://developer.spotify.com/documentation/web-api
- MusicBrainz – https://musicbrainz.org/
- Million Song Dataset – http://millionsongdataset.com/
- Python Documentation – https://docs.python.org/3.11/index.html
- SQLite Documentation – https://www.sqlite.org/doclist.html
- Numpy Documentation – https://numpy.org/doc/stable/index.html#numpy-docs-mainpage/
- Vue.js Documentation – https://vuejs.org/guide/introduction.html
- Golang Documentation – https://go.dev/doc/
- Sheet Music on Keys Image – https://www.pexels.com/photo/chords-sheet-on-piano-tiles-210764/