



Improving music mixability by using rule-based stem modification and contextual information

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Robert Sowula, BSc.

Matrikelnummer 11708475

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Associate Prof. Dipl.-Ing. Dr.techn. Peter Knees

Wien, 4. April 2024

Robert Sowula

Peter Knees

Improving music mixability by using rule-based stem modification and contextual information

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Robert Sowula, BSc.

Registration Number 11708475

to the Faculty of Informatics

at the TU Wien

Advisor: Associate Prof. Dipl.-Ing. Dr.techn. Peter Knees

Vienna, April 4, 2024

Robert Sowula

Peter Knees

Erklärung zur Verfassung der Arbeit

Robert Sowula, BSc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 4. April 2024

Robert Sowula

Danksagung

In erster Linie möchte ich meinem Betreuer, Peter Knees, für seine Ratschläge und Unterstützung während des Schreibens dieser Arbeit, danken. Ich schätze sehr, dass du mir die Gelegenheit gegeben hast, an einem für mich leidenschaftlichen Thema zu arbeiten und mir darüber hinaus auch so viele Freiheiten bezüglich der Erkundung und Entwicklung meiner Ideen gegeben hast. Dein Feedback und Ratschläge waren für mich von unschätzbarem Wert.

Ich möchte meinen wunderbaren Freunden für ihre Unterstützung und Ermutigung während des Schreibens dieser Arbeit danken. Ihr habt mir geholfen, fokussiert und motiviert zu bleiben und habt mir so viele Möglichkeiten gegeben, eine Pause zu machen und das Leben zu genießen. In diesem Sinne würde ich gerne Luki und Vale nennen, welche immer für eine Brainstorming-Session zu haben waren und mich kontinuierlich mit Motivation versorgt haben. Mein Dank gilt vor allem auch all denen, die an meiner Umfrage teilgenommen haben. Euer wertvolles Feedback hat diese Arbeit erst möglich gemacht.

Ein besonderer Dank geht an meine Freunde, die mich in die Welt der Musik und des DJings eingeführt haben. Danke, dass ihr euer Wissen und eure Leidenschaft mit mir geteilt und mir die Möglichkeit gegeben habt, die Welt des Musik-Mixens zu erkunden. Ich möchte auch Flo danken, der mich gerade in der Anfangsphase mit seinem Musikwissen unterstützt und mir Feedback zur Qualität der generierten Mixe gegeben hat.

Vor allem aber möchte ich meiner Familie und meiner Freundin Ema für ihre ständige Unterstützung und Ermutigung danken. Danke, dass ihr immer für mich da seid und mir die Möglichkeit gebt, meine Träume zu verfolgen.

Acknowledgements

First and foremost, I want to thank my supervisor, Peter Knees, for his guidance and support throughout the process of writing this thesis. I appreciate the opportunity you gave me to work on a topic that I am passionate about and give me the freedom to explore and develop my ideas. Your feedback and advice were invaluable to me.

I want to thank my wonderful friends for their support and encouragement while writing this thesis. You helped me stay focused and motivated and gave me so many opportunities to take a break and enjoy life. In that sense, I want to mention my friends Luki and Vale, who were always there for a quick brainstorming session and supported me with continuous motivation. I also extend my gratitude to everyone who participated in the listening experiment and provided valuable feedback, making this thesis possible.

A special thanks goes to my friends who brought me to the world of music and DJing. Thank you for sharing your knowledge and passion and giving me the opportunity to explore the world of music mixing. I would also like to express my gratitude to Flo, for his valuable input on music theory and the quality of the generated mixes in the early stages of this thesis.

Most of all, I want to thank my family and my girlfriend, Ema, for their continuous support and encouragement. Thank you for always being there for me and for giving me the opportunity to pursue my dreams.

Kurzfassung

Diese Arbeit evaluiert, wie Music Source Separation (MSS) und kontextuelle Informationen genutzt werden können, um musikalische Ähnlichkeitsmaße für die automatische Mix-Generation zu verbessern. Wir erkunden, wie MSS dem Bereich der musikalischen Ähnlichkeitsberechnung beitragen kann, indem inkompatible Stems mittels eines regelbasierten Ansatzes modifiziert werden. Weiters untersuchen wir, wie audiobasierte Ähnlichkeitsmaße durch kontextuelle Informationen ergänzt werden können, um ein breiteres Spektrum an Aspekten von Musik abzudecken.

Im Zuge dieser Arbeit implementieren wir ein System zur automatischen Erstellung von DJ Mixes, welches eine Vielzahl von Musikähnlichkeitsmetriken und Music Information Retrieval (MIR) Techniken integriert. Weiters stellen wir einen neuen Ansatz für die Tempobestimmung von Liedern vor, welcher bei niedriger Fehlertoleranz Ansätze des derzeitigen Standes der Technik übertrifft. Auf dieses System aufbauend, implementieren wir zwei weitere Modelle, welche regelbasierte Stem Modifikation und kontextuelle Informationen integrieren. Um die Leistung unserer Modelle zu evaluieren, implementieren wir eine Webbasierte Audio-Umfrageplattform und führen eine Hörstudie mit unseren drei Modellen und einem weiteren Modell des aktuellen Stands der Technik, welches als Baseline dient, durch.

Die Ergebnisse der Hörstudie zeigen, dass unser Ansatz zur Liederauswahl und automatischen Mix Generation den derzeitigen Stand der Technik signifikant übertrifft. Weiters zeigen wir, dass unser regelbasierter Stem Entfernung Ansatz die Qualität des generierten Mixes signifikant erhöht. Durch unsere Ergebnisse kann jedoch keine signifikante Steigerung der Qualität des Mixes durch Ergänzung musikalischer Ähnlichkeitsberechnung durch kontextuelle Informationen nachgewiesen werden. Bis auf das Baseline-Modell, bei dem Studienteilnehmer mit mehr Musikwissen und DJ-Erfahrung den Mix signifikant schlechter bewertet haben, gab es bei unseren Modellen keinen signifikanten Unterschied in den Bewertungen basierend auf dem Musikwissen oder der DJ-Erfahrung der Teilnehmer.

Abstract

This thesis assesses how music source separation (MSS) and contextual information can be used to improve musical similarity measures in the context of automatic music mixing. In particular, we explore how MSS can contribute to the field of music similarity calculation by modifying incompatible stems using a rule-based approach. Additionally, we investigate how audio-based similarity measures can be supplemented by contextual information to capture more aspects of music.

In this work, we propose and implement an automatic music mixing system, incorporating a variety of music similarity measures and music information retrieval (MIR) techniques. We also propose a novel approach for tempo detection, outperforming state-of-the-art techniques in low error-tolerance windows. Building upon this system, we implement two additional models, incorporating rule-based stem modification and contextual similarity. To evaluate the performance of our models, we implement a web-based listening survey and performed a listening experiment across our three models and a state-of-the-art model as a baseline.

The result of the listening experiment shows that our approach to song selection and automatic music mixing significantly outperforms comparable state-of-the-art. Additionally, we show that our rule-based stem removal approach significantly improves the quality of a mix. Our results do, however, not indicate any improvement in the quality of the mix by including contextual similarity to the music similarity measure. Except for the baseline model, where participants with higher musical knowledge and DJ experience rated the mixes significantly worse, no significant differences in ratings are found for different musical knowledge or DJ experience across our models.

Contents

Kurzfassung	xi
Abstract	xiii
Contents	xv
1 Introduction	1
1.1 Problem Statement	1
1.2 Aim of the Work	2
1.3 Methodological Approach	3
1.4 Outline	4
2 State of the Art	5
2.1 Beat Detection	5
2.2 Automatic Drum Transcription	7
2.3 Key Detection	8
2.4 Song Segmentation	10
2.5 Music Similarity	12
2.6 Music Source Separation	14
2.7 Automatic Mix Generation	17
3 Proposed Method	19
3.1 Beat and Tempo Detection	19
3.2 Structural Segmentation	23
3.3 Music Similarity	24
3.4 Rule-based Stem Modification	31
3.5 Mixing	31
3.6 Models	37
4 Implementation	39
4.1 Packages and Frameworks	39
4.2 Software Architecture	40
4.3 Performance	41
4.4 Frontend	42
	xv

5 Listening Experiment	47
5.1 Setup	47
5.2 Dataset	51
5.3 Mix Generation	51
6 Results and Discussion	55
6.1 Exploratory Analysis	55
6.2 Statistical Significance	63
6.3 Discussion	67
7 Conclusion	69
List of Figures	71
List of Tables	73
List of Algorithms	75
Glossary	77
Acronyms	79
Bibliography	81

Introduction

Hardly anyone these days can picture a social event without the presence of a DJ, be it a cozy company party or a big music festival. DJing has simply become an integral part of modern entertainment and even though attempts were made to replace this role with some more automated system, such as Automix by Spotify¹, their use case is heavily limited. At the time of writing, a DJ is still considered indispensable.

Contrary to what some people might think, DJing is more than just pressing a play button and is, as with any other form of musical performance, a creative process. Their skill is to seamlessly combine splices of two or more songs [Shi07], creating a unique listening experience, which manifests itself as a mix. The mix itself heavily depends on the particular DJ's experience, knowledge of music, and understanding of what resonates with the audience [BB07].

1.1 Problem Statement

Before a DJ can mix two songs, the songs need to be segmented into structural parts, such as intro, verse, chorus, bridge, and outro. A DJ also considers if these segments contain vocals or are purely instrumental. The structural segments can then be used to define transition points and to prevent the clashing of incompatible sections, such as two vocal segments.

Song selection is essential to a DJ set, setting the mood for the whole listening experience. Timbre is considered one of the fundamental attributes DJs use for song selection and ordering. The average timbral difference between songs in a DJ set is smaller than compared with random [electronic dance music (EDM)] songs. Hence, it is usually desired to order songs to reduce the timbral difference. This is also supported by the findings of

¹<https://spotify.com>

Kell et al. [KT13], who also found that the timbral difference of songs in an album is the smallest due to the songs being sonically coherent and featuring similar sound design and instruments.

Research in this area [Pan+17] hints that models based solely on timbre perform considerably worse than only rhythm-based models. Combinations thereof, however, can drastically improve the similarity performance [SWP10]. Those models are usually evaluated on the complete song and do not consider stems. Stem refers to a format of audio files that separate the different elements of a track, such as the drums, bass, vocals, and melody, into individual tracks or "stems". This separation is done using a music source separation (MSS) technique.

On-the-fly stem generation is becoming an indispensable feature of modern DJ software. Thus, more and more DJs are incorporating it into their live mixes. However, using stems for automatic mix generation is unexplored and would open up new opportunities. Publications such as [Hua+21] only evaluate the generation of a mashup by replacing stems with each other, which is not a desired goal for an automatic mix system or live performances.

A typical transition technique is fading out the first song while fading in the second. During this transition period, the two songs are audible for some time. Even if the BPM and key match perfectly and the timbral compatibility is high, a dissimilar drum pattern, e.g., with off-beats at different times than the original track or clashing vocals, can result in a combination that does not sound right. Removing incompatible stems of one of the two tracks during a transition or mix would solve many problems in traditional mixing.

Less often considered by DJs is the use of contextual information for track selection. Contextual information, such as song lyrics, contains information that audio-based approaches cannot capture and vice versa. Since approaches such as [HDE09] have shown that lyrics can be used to predict the mood of a song, combining audio-based and contextual information might further improve the quality of track selection.

1.2 Aim of the Work

This thesis addresses two problems in the field of automatic music mixing. First, we aim to solve the problem of making seemingly not-mixable songs compatible by utilizing music source separation to extract stems and modifying them by a rule-based approach. Since percussive elements are the defining component of rhythm, we hypothesize that removing the drum stem of a song with low rhythmic similarity will increase the mixability of the song. Further, we hypothesize that vocal stem removal will make song segments with clashing vocals suitable candidates for mixing in.

As a second objective of this thesis, we will analyze if our musical similarity metric can be improved by additionally considering contextual information. We are especially interested in the effect of mixing song excerpts with high lyrical similarity. We hypothesize that

mixing song excerpts with high lyrical similarity will result in a more natural-sounding mix and might also allow the mixing of segments with clashing vocals.

To this end, we formulate the following research questions (RQs):

- RQ1: Can the "mixability" of tracks be increased by selectively removing stems? Which measures can be used to formulate rules on which stems to remove?
- RQ2: Can mixes be improved by considering contextual data such as lyrics during the similarity computation?

Answering the first research question (RQ1) will contribute to the area of similarity calculation and will remove certain constraints that prohibit songs from being mixed. As for the second research question (RQ2), the results will be beneficial to improving the accuracy of music similarity calculation.

1.3 Methodological Approach

To answer our research questions, we will closely follow the Design Science Research Methodology (DSRM) [BHM20]. We will start by conducting a literature review to identify the state-of-the-art in automatic music mixing, their music similarity metrics, and music source separation techniques. The literature review results will act as the theoretical basis for this thesis.

We will then design and implement an automatic music-mixing system incorporating stem modification and contextual information. In total, we will design three systems: a base model incorporating our mixability approach, a model incorporating stem modification, and a model incorporating both stem modification and contextual information. Additionally, we will develop a baseline model based on existing research to compare the performance of our system.

The quality of a mix is a very subjective measure and depends on factors like music background, taste, and understanding of rhythmic and harmonic compatibility [Hua+21]. Thus, similarity ratings between songs also lack ground truth corpora [RBH13]. We will follow the evaluation methods from prior works [Dav+14], [Hua+21] and evaluate the performance of our proposed solutions with subjective methods. In particular, we will conduct a listening experiment to evaluate the performance of our system.

To evaluate the song scheduling aspect of our models in more detail, we will split the listening experiment into two parts. In the first part, the participants will evaluate the pair-wise mixability of the complete songs corresponding to the excerpts scheduled by our models. In the second part, the participants will listen to the mix and evaluate each transition's quality.

Afterward, we will conduct an exploratory data analysis to identify general trends and patterns between the different models and the listening experiment results. To support our findings, we will conclude a statistical analysis to test the significance of our results.

1.4 Outline

[Chapter 2](#) provides an overview of the state-of-the-art in automatic music mixing and builds the theoretical foundation for this thesis. [Chapter 3](#) presents our proposed method and the design of our automatic mixing system. In this chapter, we also address shortcomings of the current state-of-the-art and how our methods aim to overcome those. [Chapter 4](#) gives insights into the implementation of our system and provides performance metrics. Details regarding the listening experiment and the used dataset are described in [chapter 5](#). [Chapter 6](#) explores the results of the listening experiment and analyzes the data with regard to previously stated hypotheses using statistical significance tests. Finally, [chapter 7](#) concludes the thesis and provides an outlook on future work.

State of the Art

Automatic music mixing is an ensemble of tasks and techniques from [music information retrieval \(MIR\)](#). This chapter introduces the state-of-the-art research in the field of automatic music mixing and its subfields. In doing so, we will describe naive approaches followed by more sophisticated techniques and current state-of-the-art methods.

We will first introduce beat and key detection, which are fundamental to the scheduling and mixing process. Afterward, we discuss automatic drum transcription to provide the foundation for our rhythmic similarity approach. We then introduce naive and sophisticated song segmentation approaches, followed by a detailed analysis of musical similarity measures. Finally, we explain [music source separation \(MSS\)](#) techniques to establish the foundation for our stem modification approach and give an overview of the current state-of-the-art mix generation systems.

2.1 Beat Detection

The estimation of beat- and downbeat positions is critical for the process of beatmatching two songs. This section gives an overview of early beat detection approaches and explores the current state-of-the-art techniques.

2.1.1 Early Signal Processing Techniques

Early methods in beat detection relied on detecting the *onsets*, the beginnings of musical notes or sounds, which are usually characterized by significant changes in the audio signal. Ellis [\[Ell07\]](#) estimates a global tempo to construct a cost function and then uses dynamic programming to predict the beats that correlate most to the tempo and moments of high *onset strength*. Other approaches, such as [\[GM95\]](#), extract a noise component besides the onset components and use that to detect the drum components' onset times to increase the reliability of the beat predictions.

Besides detecting onsets, other approaches are based on the harmonic and timbral characteristics of the audio. Harmonic approaches, such as [PP11], are based on the observation that the harmonic progressions change more often on strong beats than on other beats. On the other hand, timbral approaches, such as [Jeh05b], are based on the observation that alternation to the timbral content occurs more likely on downbeats and section boundaries.

These approaches can be broken down into three steps, as depicted in Figure 2.1. First, feature vectors are computed over the audio. Then, a beat/downbeat detection function is derived over those feature vectors. Finally, the beat/downbeat positions are derived with the help of a temporal model. [Dur+17]

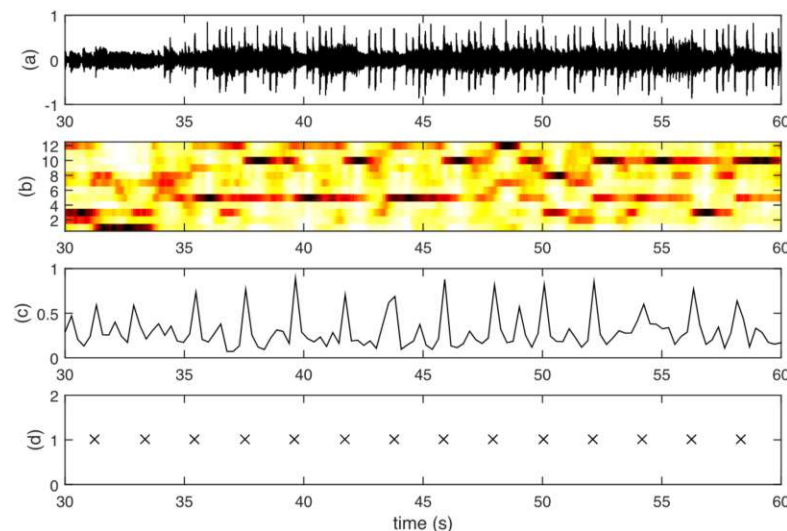


Figure 2.1: This figure depicts the three-step beat detection process. (a) shows the original audio signal x . (b) shows the feature vector F_i of a specific feature, in this case, the chroma vectors. (c) illustrates the downbeat detection function d . (d) shows the discrete downbeat position sequence s . [Dur+17]

2.1.2 Supervised Deep Learning Techniques

With the continuous success of deep learning, supervised beat-tracking approaches have gained popularity. Böck and Schedl [BS11] proposed a new approach by employing recurrent neural networks (RNNs), specifically a Bidirectional Long Short-Term Memory (BLSTM) model, to detect beat locations. Durand et al. [Dur+16; Dur+17] employed convolutional neural network (CNN) structures to obtain beats and downbeats.

Both [RNN] and [CNNs] have advantages and disadvantages in beat tracking. [RNNs] perform better in modeling short- and long-term time series, whereas CNNs perform better at extracting features in a broader range [JLL19; Fue+18]. Recent works [HCD21; CFG21] used convolutional recurrent neural networks (CRNNs) to leverage the advantages of [RNNs] and [CNNs] and outperform previous beat-tracking approaches. A common [CRNN]

architecture is to attach the outputs of the convolutional models to the input of the recurrent layers [Vog+17; Fue+18; DB19].

Davies and Böck [DB19] proposed another well-performing architecture for offline beat tracking using [CNNs] and Temporal Convolutional Networks (TCN).

We differentiate between offline and online beat-tracking approaches. Online approaches are casual, meaning they solely rely on past- and present information and are usually used for real-time beat tracking [HCD21].

Online beat tracking is usually computationally constrained, only has partial access to data, and cannot correct previous mistakes, thus facing many unique challenges [HD21]. On the other hand, offline approaches such as [BKW16] require a time signature as input to classify the downbeats correctly. In contrast, online approaches such as [HCD21] automatically monitor the time signature and tempo on their own.

2.2 Automatic Drum Transcription

Automatic drum transcription (ADT) is the task of creating symbolic transcripts from audio data. Many approaches have been proposed, yet a general solution still needs to be provided for this problem, as reviewed by Wu et al. [Wu+18].

While high accuracies have been achieved for detecting isolated drum hits, classifying multiple drum hits that occur simultaneously poses another challenge. This is further complicated when multiple instruments are combined, creating a polyphonic mix. [SSH16]

ADT systems can be mostly categorized into the following categories: *segment and classify*, *separate and detect*, *match and adapt*, and deep neural network (DNN)-based approaches. Segment and classification methods divide audio into segments using information derived from beat tracking. Afterward, features are extracted from those segments, and the drum instruments are classified over these segments. Match and adapt approaches associate instruments with pre-determined templates and iteratively update these templates to match the recording's spectral character. Separate and detect methods focus on separating the music signal into individual drum sources before identifying the onsets of each source. [Wu+18; GR08]

DNN-based approaches mostly consist of three steps, as depicted in Figure 2.2. First, features are extracted from the audio, then fed into a trained DNN model. The activations of the output layer of the DNN are then peak-picked to obtain the individual drum onsets.

Southall et al. [SSH16] proposed a DNN-based approach that uses a RNN and achieves state-of-the-art performance on drum-solo audio but lacks accuracy in polyphonic audio. Later on, CNN and CRNN architectures were proposed by Vogl et al. [Vog+17; Vog+17] and Southall et al. [SSH17] with high accuracy also in polyphonic audio.

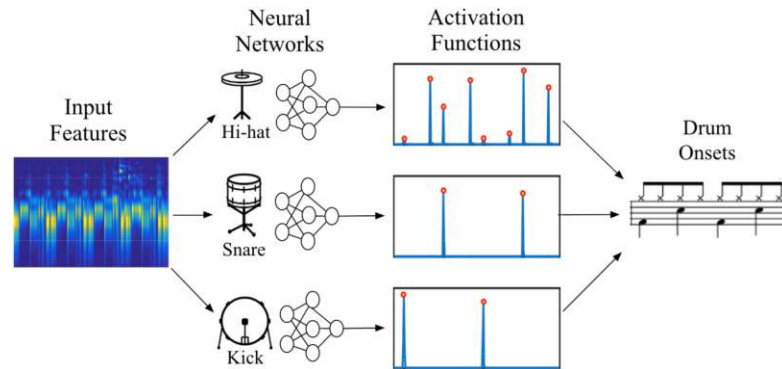


Figure 2.2: Illustrations of the three-step [DNN](#)-based [ADT](#) approach. After extracting features from the audio, the features are fed into three separate [DNNs](#), one for each drum instrument. The activations of the output layer of the [DNNs](#) are then peak-picked to obtain the individual drum onsets. [SSH16](#)

Wei et al. [WWS21](#) proposed a beat-informed [CNN](#) architecture that leverages a state-of-the-art beat tracker to make a beat-informed prediction. This model currently achieves state-of-the-art performance on polyphonic audio.

2.3 Key Detection

Classifying the musical key in a musical piece is fundamental to mixing two songs, as it ensures harmonic compatibility between those songs. Mixing songs with a harmonic-compatible key avoids dissonance and allows for smooth transitions and a more cohesive sound. This section will first give some musical theory background regarding musical keys and their compatibility based on [Lai16](#). Afterward, we will discuss current state-of-the-art approaches for key detection.

2.3.1 Music Theory

Musical notes are the fundamental units of sound in music and are characterized by their pitch, which is determined by their frequency. An octave is the interval between a note and another with double or half its frequency. For example, if a note has a frequency of 440 Hz, such as A_4 , the note an octave above it would be A_5 with 880 Hz. Despite the pitch difference, notes separated by an octave are perceived as having the same tonal quality.

In Western music, the twelve-tone equal temperament system is mainly used, where an octave is divided into twelve equally tempered parts. This division forms the basic framework for musical scales, the sequences of notes ordered by pitch. The interval between adjacent notes is also called semitones or half-steps. The major and minor scales in Western music are the most commonly used scales. Each of those scales consists of seven distinct notes with specified intervals. For example, the major scale follows the

semitone interval pattern 2-2-1-2-2-1. Therefore, the C major scale would consist of the tones C-D-E-F-G-A-B-C.

A musical piece's key is the scale used as a basis for the played notes. It establishes the basis for how harmonies interact within the composition and affects how individual notes and chords work together. We have already introduced the concept of an octave with a 12-semitone distance. Due to their harmonically stable and pleasing sound, octaves and the so-called perfect fifth are called perfect consonance. A perfect fifth is an interval of seven semitones. Another harmonically compatible key relation is keys that are *relative*. Relative keys such as F major and D minor share the same key signature, thereby being considered harmonically compatible. Figure 2.3 illustrates those relations. Adjacent keys in a scale are separated by a perfect fifth. Adjacent keys (up or down) are relative.

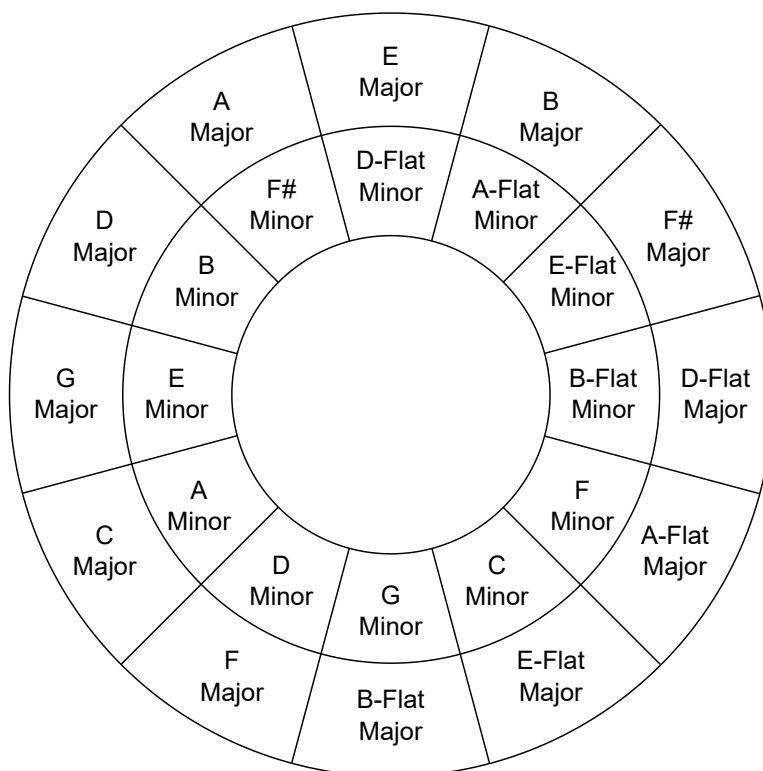


Figure 2.3: The circle of fifths - also known as *Camelot Wheel* using a different key-coding **Mix** - is widely used by DJs to find a key-compatible song to mix.

2.3.2 State of the Art

One of the most common approaches to key detection was based on a template-matching principle. First, a time-frequency representation of the audio is extracted. Afterward, this representation undergoes a process to filter out irrelevant information and is converted into a pitch-class profile (PCP) or chromagram. The chromagram is a vector that

encapsulates the intensity of each semitone of the chromatic scale at a given time frame. These chroma vectors are then accumulated over time, creating a comprehensive feature vector. The key is then derived by matching each key's template vector with the feature vector. The challenge with key templates is that they differ for musical genres. This leads to key detection systems performing well on genres they were designed for and poorly on others. [Far+16; FJH17]

A notable template-matching-based approach is *KeyFinder* [Sha11], largely due to its available open-source implementation. Faraldo et al. [FJH17] improved upon *KeyFinder* by utilizing a multi-profile template-matching system for key classification.

Mahieu [Mah16] proposed key detection architectures based on supervised learning. The system takes a chroma feature vector as input and returns the key. The best-performing architecture proposed by the author consists of two layers. The first layer contained a softmax classifier to determine dominant notes. In the second layer, logistic regression classifiers were used to classify the key using the determined dominant notes as input. However, with an accuracy of around 58%, this approach could not outperform *KeyFinder*.

A notable advancement was made with the CNN-based key detection method proposed by Korzeniowski and Widmer [KW18]. The model was trained using short audio snippets rather than complete songs to better generalize across genres. The result is a genre-agnostic key classifier with state-of-the-art performance across various genres.

With the gaining popularity of transformer models and their encoder/decoder architectures in large language models (LLMs), the focus is increasingly shifting towards music understanding. As introduced by Li et al. [LI+24], the MERT model exemplifies this shift. It integrates advanced transformer architectures, combining deep acoustic understanding with self-supervised learning, reaching state-of-the-art performance in many MIR tasks. While this model did not outperform the current state-of-the-art by Korzeniowski and Widmer [KW18], it offers a promising direction for future developments in key detection techniques.

2.4 Song Segmentation

Song segmentation is the task of dividing a song into musically meaningful segments. It is essential in DJ mixing as it helps to find suitable transition points, also referred to as *cue points*, to mix two or more songs. This task usually involves two steps: (i) detecting the segment boundaries and (ii) classifying the segments into musically meaningful categories (labeling), such as verse, chorus, bridge, etc.

Foote [Foo00] proposed the first approach that worked reliably on any audio source, regardless of complexity, by extracting Mel-frequency cepstral coefficients (MFCCs) and then computing a self-similarity matrix over them. This approach is based on the observation that the repetition of musical patterns is a common characteristic of music. Local self-similarity and cross-similarity are computed over past and future time frames to detect repeating patterns. Segment boundaries will have a high self-similarity and

low cross-similarity, whereas boundaries within a segment will have low self-similarity and high cross-similarity. Although its performance was surpassed by more recent works, such as [NB14; Ser+14], it is, due to its simplicity, still widely used by more recent works in the automatic mashup generation field, such as [Dav+14; VD18].

Nieto and Bello [NB14] extracted chroma vectors (cf. subsection 2.3.2) to capture the harmony of the song. Then, 2D-Fourier Magnitude Coefficients (2D-FMC) segments are computed using a magnitude 2D-Fourier transform over the chroma vectors. These resulting feature segments are invariant to key, phase shift, and local tempo changes. Finally, the segments are clustered using k-means, with the Bayesian Information Criterion used to determine the optimal number of clusters.

Serrà et al. [Ser+14] proposed a boundary detection method that computes a novelty curve over the audio's computed *structure features*. Structural features capture both local and global characteristics of an audio signal. By examining the relationships of each time frame in the audio with all others in the same series, these features provide a detailed, frame-wise representation that captures the overall structural characteristic of the time series [Ser+12; Ser+14]. The resulting segments are then labeled using a three-step approach. First, similar segments are grouped together. Then, dynamic thresholding is employed to determine which segments are similar to share the same label. Finally, a transitivity constraint is applied to ensure that if two segments are similar to a third one, they should also be similar to each other. This step is crucial to maintaining consistent labeling, as it prevents contradictory or inconsistent labeling of segments with similar characteristics.

Recent works rely on deep learning approaches to detect segment boundaries and label segments. Many well-performing [CNN]-based approaches have been proposed in the past [GSU14; McC19; Won+20].

The current state-of-the-art approach employs a transformer-based architecture [WHS22]. Transformer-based models require large amounts of data to train on. Since the available datasets for song segmentation are small, transformer-based models have not previously been used for song segmentation. The authors have overcome this limitation by employing an adapted version of the spectral transformer-in-transformer (SpecTNT) architecture [Lu+21]. SpecTNT differs from conventional transformer models in its unique spectral and temporal data handling. Unlike standard transformers, which typically process sequential data, SpecTNT incorporates two encoders: a spectral encoder and a temporal encoder. The spectral encoder extracts spectral features, such as timbre or harmony, in the form of embeddings for each time step. The temporal encoder then exchanges those spectral embeddings along the time axis, capturing structural information of the audio. In contrast to other segmentation approaches that assign an abstract label to each segment, such as 'A', 'B', 'C', etc., SpecTNT assigns a semantic meaningful label, such as 'verse', 'chorus', 'bridge', etc. to each segment, offering more meaningful labeling.

2.5 Music Similarity

Playlist generation with an inherent sequential ordering and song selection and its ordering for a mix mainly rely on the audio similarity between songs [KT13; Fle+08]. The most prominent musical similarity types used in the context of automatic mashup systems [Dav+14; VD18] are rhythmic and timbral similarity. This section will give a more detailed overview of those similarity types and introduce how contextual similarity can be used to improve the performance of automatic mashup generation.

2.5.1 Rhythmic Similarity

A common rhythmic descriptor are the Fluctuation-Patterns (FPs). They measure the periodicity of the loudness per frequency band and model characteristics of the audio signal, which can not be captured by using a spectral representation [Pam06]. Pohle et al. [Poh+09] proposed two modifications to the FP approach, the *Onset Patterns (OPs)* and the *Onset Coefficients (OCs)* and proposed a unified algorithm that included timbral information to improve the performance of rhythmic extraction.

Rocha et al. [RBH13] used *event density*, *fluctuation patterns* and *rhythm patterns* to model rhythmic similarity. The authors defined event density as the average frequency of events, such as the amount of onsets per second, and computed them over four frequency bands. Following the modifications of Pohle et al. [Poh+09], the authors utilized the likely onsets from the *Onset Pattern* approach to extract patterns resistant to tempo changes. The onsets are detected over four frequency bands, and four rhythmic patterns are extracted from each band over different measure lengths, resulting in 16 rhythmic patterns.





Rhythm in Musical Notation	Attack Positions of Rhythm	Most Common Instrumental Associations
	1/5/9/13	Bass drum
	5/13	Snare drum; handclaps
	3/7/11/15	Hi-hat (open or closed); also snare drum or synth "stabs"
	All	Hi-hat (closed)

Figure 2.4: Example of the most typical (even) drum patterns found in [EDM] [But06], p. 82] [PBH14].

Panteli et al. [PBH14; Pan+17] proposed an approach that works similarly to the *rhythm patterns* approach by Rocha et al. [RBH13]. It builds upon the same concept that there is a repeating rhythmic pattern in [EDM] songs (e.g., [Figure 2.4]) from which rhythmic characteristics can be derived. Since the rhythmic patterns usually consist of multiple instruments, the authors proposed to split the audio into so-called *rhythm streams*. *Rhythm streams* is a more informed approach than the one by Rocha et al. [RBH13], which uses a fixed number of frequency bands to extract the rhythmic patterns. Then, the underlying rhythm of each rhythmic stream is characterized using the repetition

of the rhythmic sequences, the metrical distribution, and the attack shape to classify percussive and non-percussive instruments. The rhythmic characteristics of the song are then represented by constructing a feature vector out of the rhythmic characteristics of each rhythmic stream. Using their constructed feature vectors, the authors used the cosine distance to compute the similarity between the two songs.

Davies et al. [Dav+14] used a different approach in their automatic mashup system. Instead of extracting rhythmic patterns of different instruments, the authors extracted kick and snare drum onsets using the approach of Robertson et al. [RSD13]. The detected kick and snare drum onsets are then quantized at 12 equally spaced positions over each beat and stacked into a 24-dimensional feature vector. The similarity between the two songs is computed using the cosine similarity between the two feature vectors.

2.5.2 Timbral Similarity

Timbre has been identified as the key factor in determining the order and selection of songs in a playlist. Similarly, timbre plays a crucial role in DJ mixes. It was observed that songs in DJ mixes tend to be more similar regarding timbre compared to a random assortment of EDM songs, though they are not as similar as those found within an individual album. [KT13]

Timbre is perceived as a multidimensional phenomenon and represented in terms of timbral spaces [Pee+11]. Most approaches proposed so far are based upon multidimensional scaling (MDS) to map perceived psychoacoustic timbral features to a geometric space of as few dimensions as possible [BM08]. Spectral centroid, attack time, and irregularities in the spectral envelope have been found to be the primary components of timbre [Cac+05]. Following this definition, timbre was modeled using various approaches.

Pohle et al. [Poh+09] described timbre using MFCCs, spectral contrast coefficients, and the descriptors *Harmonicness* and *Attackness*, where the latter two capture the harmonic and percussive characteristics of the audio signal. Rocha et al. [RBH13] and Panteli et al. [Pan+17] considered timbre as the perception of the polyphonic texture and modeled the similarity through the following three types of features. MFCCs were used to represent the spectral envelope and spectral flatness was computed to describe the tonal character. The authors additionally incorporated “dirtiness” (also referred to as auditory roughness) as it is an essential part of EDM music and is believed to be explained by the de-tuning that producers apply to their synthesizer sounds.

2.5.3 Contextual Similarity

Besides audio-based information, contextual information proved beneficial in music recommendation tasks, even outperforming audio-based approaches in certain settings [VP20]. Contextual information also enables capturing moods and topics, opening up the possibility for more meaningful song recommendations.

Knees et al. [KS13] categorized contextual similarity approaches into three categories: *text-retrieval based*, *co-occurrence based* and *user ratings or listening habits based*. Approaches based on text retrieval use web texts, collaborative tags, or song lyrics as primary data sources. Co-occurrence methods, on the other hand, employ resources such as page counts, playlists, microblogs, and peer-to-peer networks. Finally, strategies focusing on user ratings or listening habits typically utilize collaborative filtering techniques.

We will focus on song lyrics-based text-retrieval approaches, as song lyrics are easy to obtain and have proven to improve performance when combined with audio features [WSW21].

Early approaches, such as Mahedero et al. [Mah+05], relied on a standard TF-IDF approach in combination with the cosine similarity measure to compute the similarity between lyrics of two songs. Other early approaches tried to reveal topic clusters [KKP08] or classify songs into genres [MNR08] or moods [HDE09].

With the advancement of natural language processing (NLP) techniques, more sophisticated approaches incorporating the lyrics' semantic meaning have been proposed. Much of the recent work in semantic text similarity is based on learning word vector representations using neural language models. Those word vectors are fed into a neural network to capture the semantic relationships between words. [WSW21]

Word vectors, or word embeddings, project words from a sparse 1-of- V encoding (where V is the vocabulary size) to a dense, lower-dimensional vector space through hidden layers. This process effectively extracts and encodes semantic features of words. Semantically similar words are positioned closer to each other in this reduced vector space, as measured by Euclidean or cosine distance. [Kim14]

One of the most prominent approaches for word embedding learning was the *Word2Vec* technique [Mik+13b; Mik+13a]. It uses a continuous bag-of-words (CBOW) or a skip-gram model to obtain word embeddings. The CBOW model predicts a word by first masking it and then predicting it based on its surrounding context. The Skip-gram model inverts this task by predicting context words given a specific target word. [Mik+13b; SWB20]

Recently, transformer-based models have become the state-of-the-art approach for NLP tasks [VTH23]. Reimers and Gurevych [RG19] proposed a transformer-based model called *Sentence-BERT* (SBERT). SBERT builds upon the pre-trained BERT model by fine-tuning it on a siamese and triplet network architecture to learn semantically meaningful sentence embeddings. Semantic similarity of sentences is then computed using cosine similarity between the sentence embeddings.

2.6 Music Source Separation

Music source separation (MSS) separates a music signal into its components, so-called stems. Recently, work in [MSS] focused on separating a song into four stems: *vocals*, *drums*,

bass, and *other* [Mit+22]. The recent adaptation of music source separation (MSS) in DJ software, such as by *Serato*¹, marks a shift in mixing techniques. Traditional mixing techniques are complemented by new techniques that isolate certain stems, opening up new creative possibilities for DJs.

MSS approaches either work on the spectrogram or the waveform domain, with recent trends showing a blend of both [Déf21; RMD23]. Traditional approaches were primarily based on unsupervised methods, whereas recent approaches rely entirely on supervised deep learning techniques [Déf+21].

Spectrogram-based approaches initially utilized simple networks and progressed to more advanced multi-scale CNN and RNN architectures [TGM18]. Notable developments include MMDenseLSTM [TGM18] and D3Net [TM21], which employ dilated convolutions with dense connections. Spectrogram-based models have had state-of-the-art performance in MSS for a long time but have recently been surpassed by waveform-based models [Déf+21]. Waveform-based approaches deal directly with raw audio waveforms. Initial models in this domain, such as *Wave-U-Net* [Jan+17], were adapted from spectrogram models [SED18] but have generally shown less success compared to their spectrogram counterparts [Déf+21].

Recently, waveform-based approaches, such as the initial *Demucs* model [Déf+21], have shown state-of-the-art performance in MSS. The initial *Demucs* model features a U-Net architecture with a convolutional encoder and decoder linked by bidirectional LSTM layers. This structure facilitates learning from the raw input waveform to generate waveforms for each source, focusing on music synthesis rather than masking approaches. *Demucs* has shown superior performance in terms of Signal-to-Distortion Ratio (SDR) and naturalness of audio in human evaluations, especially excelling in separating bass and drums.

Défossez [Déf21] improved upon the *Demucs* architecture by introducing a hybrid model that combines spectrogram and waveform domains. It features dual branches for processing both time-domain and frequency-domain aspects. The model integrates compressed residual branches, dilated convolutions, LSTM layers, and local attention mechanisms.

Rouard et al. [RMD23] further improved the *Demucs* architecture by introducing the Hybrid Transformer *Demucs* (HT *Demucs*), a model that replaces the inner layers with a cross-domain Transformer Encoder. This architecture incorporates cross-domain Transformer Encoders, enhancing spectral and temporal information processing. This approach enhances spectral and temporal information processing, improving performance and higher Signal-to-Distortion Ratio (SDR) values. Figure 2.5 depicts the architecture of the HT *Demucs* model.

¹<https://serato.com/dj/pro/stems>

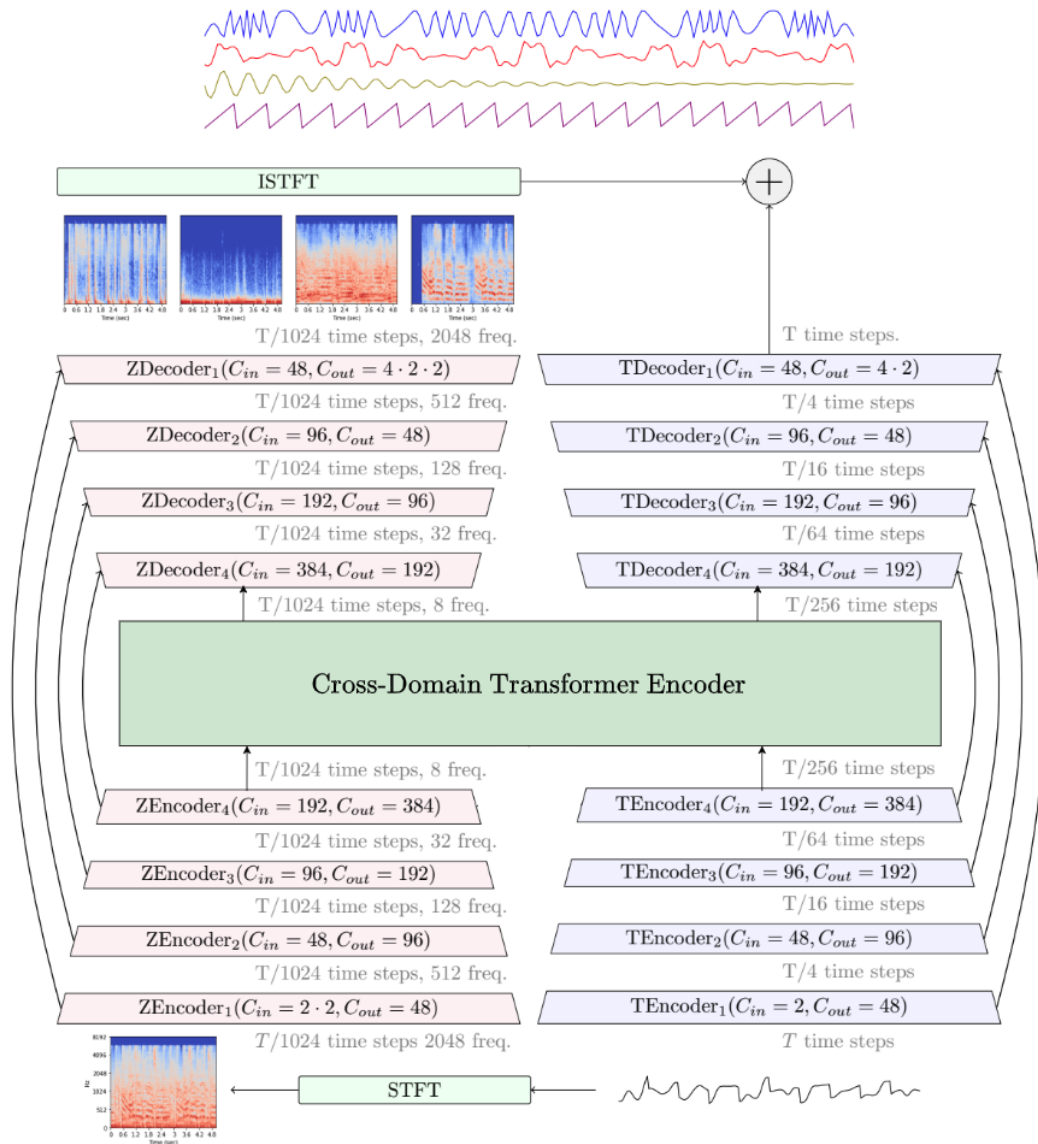


Figure 2.5: Illustration of the hybrid transformer Demucs architecture. The architecture consists of two parallel stages. The input waveform is processed by a temporal encoder and simultaneously transformed into its spectral representation using the **short-time Fourier transform (STFT)** and processed by a spectral encoder. Spectral encoders are prefixed with a Z, whereas temporal encoders are prefixed with a T. These two representations are fed into a Cross-domain Transformer Encoder using interleaved Transformer Encoder layers and cross-attention Encoder layers. Temporal and spectral decoders then process the output of the Cross-domain Transformer Encoder. The output spectrogram is then transformed back into the waveform dimension using the inverse-STFT (ISTFT) and summed with the output of the waveform outputs of the other stage, giving the final model output. **RMD23**

2.7 Automatic Mix Generation

Jehan [Jeh05a] proposed an automated DJ system capable of beat matching on downbeats and transitioning to the next song on rhythmically similar segments. The system assesses rhythmic similarity by computing rhythmic patterns and uses a simple time-stretching algorithm for tempo matching. However, the proposed system does not incorporate harmonic or timbral information or implement automatic track selection or ordering.

Lin et al. [LLT09] improved upon Jehan's [Jeh05a] work by incorporating pitch information and introducing a method for automatic track selection and ordering. First, the songs that are too dissimilar from the rest are filtered out to avoid mixing songs that differ too much from each other and to reduce the computational complexity. The dissimilarity measure is based on the loudness, tempo, and pitch information, using simple chroma features to capture the pitch. The transition point is then determined by considering rhythmic and chroma features. The system additionally allows mixing songs with heavily differing tempi by calculating the transition length based on the tempi difference and gradually adjusting the tempo during the transition to the tempo of the following song.

Ishizaki et al. [IHT09] proposed a method for reducing discomfort when mixing songs with heavily differing tempi in his automatic DJ system. A naive DJ method would be to adjust the following song's tempo to the previous song's tempo. However, with significantly differing tempi, such a tempo adjustment would lead to a degradation in audio quality and, thus, discomfort for the listener. To avoid this, the authors proposed using an intermediate tempo between the two songs, spreading the discomfort of the tempo adjustment over the two songs.

Davies et al. proposed AutoMashUpper (AMU) [Dav+14], an automatic mashup system that mixes songs using a mashability estimate over phrase-level segments of songs. The authors employ snare and kick drum onset detection functions to estimate the beat and tempo of songs. Downbeats are estimated using spectral information and additional information on the drum onsets. The authors estimate the mashability of two songs independently per section by employing a structural segmentation approach based on Foote [Foo00], as described in section 2.4. The mashability measure consists of three components: rhythmic similarity, harmonic similarity, and spectral balance. The rhythmic similarity is computed using an implicit representation of rhythm by using the rhythmic information obtained from the snare and kick drum onset detection function. This differs from previous approaches described in subsection 2.5.1 that extract rhythmic patterns. The snare and kick drum onset detection function are then quantized over 12 equally spaced beat positions and stacked into a 24-dimensional feature vector. The rhythmic similarity is then computed over all the beat shifts of the next song by computing the cosine similarity between the feature vector of the current song segment and the feature vector of the next song across all beat shifts. The harmonic similarity is computed similarly. A beat synchronous chromagram is computed for each song. Then, in addition to computing the harmonic similarity over all beat shifts, the harmonic similarity is computed over all possible key shifts. This allows determining the key shift to achieve

the highest harmonic similarity. The spectral balance measure ensures a balanced mix by computing the perceived loudness across three frequency bands and all beat shifts. Combining these three similarity measures and determining the beat with the maximal similarity across all beat shifts determines the starting point of the next song. Finally, songs are mixed by first beat matching the song segments, pitch shifting using the key shift value obtained during the harmonic matching stage, and adjusting the loudness to create a balanced mix.

Hiari et al. [HDM15; HDM16] proposed an automatic DJ system based on latent topic modeling and beat similarity for song selection and cue point estimation. The authors proposed a method to map chroma features to a textual description, called *Chroma Words*. Then a generative statistical model, latent Dirichlet allocation (LDA) [BNJ03], is used to create a topic model of songs of various genres by using the *Chroma Words* as input. The similarity between two songs is then computed by segmenting a song and comparing each segment's latent topics to the other song's latent topics. The second similarity feature, beat similarity, is computed by detecting peaks under the 500 Hz frequency band, assuming that the rhythm is expressed using bass and drum sounds, and comparing the peak distances between two song segments.

Huang et al. [Hua+21] proposed a mashup system that uses isolated stems of different songs to create a mashup. Unlike the previously described automated mixing systems, this approach focuses on mixing a combination of stems, ensuring that each stem type is used only once. Stems are separated using an in-house music source separation model. The stems are then used to train a machine learning model to predict the compatibility of stems based on self-supervised and semi-supervised methods. The model was trained using stems from the same songs as positive examples, and combinations of stems from different songs with unadjusted tempo and key as negative examples. The model was additionally trained using "average" examples using combinations of stems with matching tempo and key. The system was evaluated using a listening test with [AMU] as a baseline and proved to outperform [AMU] in most aspects when calculating compatibility between stems.

Proposed Method

This chapter outlines the proposed method for our automatic music-mixing system. In [section 3.1](#), we detail our beat and tempo detection algorithm and explain why our proposed algorithm is more suitable for our task than other state-of-the-art algorithms. We support our claims by presenting a benchmark and comparing our tempo estimation algorithm with other state-of-the-art algorithms.

In [section 3.2](#), we explain how we divide a song into musically meaningful segments, which we will later use to transition between songs. [Section 3.3](#) is the core of this work, where we detail our music similarity and mixability calculations and song scheduling algorithm. In addition to audio-based similarity measures, we incorporate contextual similarity and capture the semantic meaning of music.

In [section 3.4](#), we detail the rules for applying our stem modification algorithm and propose our vocal segment detection approach. We then explain our mixing process in [section 3.5](#), including the design of our equalization filters, their application during transitions, and our process of modifying individual stems based on previously established rules.

3.1 Beat and Tempo Detection

Most of the [EDM](#) music is characterized by a constant tempo. Popular DJ software, such as *Serato*¹ and *Rekordbox*² build upon this very assumption and estimate an evenly spaced beat grid that allows for simplified alignment of beats of two or more songs.

We will build our beat and tempo detection pipeline upon the same fixed beat grid approach. To build a beat grid, we need two types of information: the song's tempo

¹<https://serato.com>

²<https://rekordbox.com>

and the location of the first downbeat, as song segments usually start at the beginning of a downbeat. We derive the beat positions, including the beat types (first-, second-, third- and fourth beat) using the state-of-the-art beat tracking system *BeatNet* [HCD21]. *BeatNet* returns only the beat positions that are characterized by audio features and returns no or fewer beat positions during silent parts or low-energy parts of the song. Calculating the tempo by averaging the inter-beat intervals, represented as

$$bpm = \frac{60}{\frac{\sum_{i=1}^{n-1} t_{i+1} - t_i}{n-1}}, \quad (3.1)$$

where t_i is the timestamp of the i -th beat and n is the number of beats, can lead to octave errors ($\frac{1}{2}$, $\frac{1}{3}$, 2, 3 multiple of the tempo). Specifically, the problematic tempos are the $\frac{1}{3}$ and 3 multiples of the true tempo for non-duple meter music.

Using a rounded median of the inter-beat intervals

$$bpm = \frac{60}{\text{Median}(\Delta t)} \quad (3.2)$$

where $\Delta t_i = t_{i+1} - t_i$ for $i = 1, 2, \dots, n - 1$ and t_i are the timestamps of the beat with the median inter-beat intervals rounded to two decimal places, proved to be slightly more reliable, although still showing similar problems as the averaging approach.

To solve the octave error problem, we model the beat grid estimation as a 2-dimensional constrained minimization problem. Given the detected beat timings t_i , where $i = 1, 2, \dots, n$. Note that some beats might be missing due to detection errors. We want to find the optimal first downbeat position g_1 and the tempo bpm such that the beat positions of the constructed beat grid g_j are evenly spaced and that the average difference between the detected beat positions t_i and the estimated beat grid positions g_j is minimized.

[Algorithm 3.1] describes the beat grid estimation algorithm in detail. The algorithm first estimates the tempo using the median of the inter-beat intervals and then performs a global search to find a good solution. The global search utilizes the dual annealing algorithm, a variant of the simulated annealing algorithm, that is paired with a local search algorithm for accepted solutions [Xia+13]. As a local search algorithm, we use the L-BFGS-B algorithm, a version of the limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm that supports bound-constrained optimization [BPN96; Zhu+97]. As an objective, we calculate the mean of the minimum absolute differences between each estimated beat grid position g_i and all detected beat positions t_j . The median tempo is usually a good initial guess, but since we are doing a global search with loose boundaries, we might discard valid solutions close to the initial guess. Therefore, we additionally perform a refined optimization with restricted boundaries around the initial guess. Since the offset of the solution can be more than a beat duration away from the true first downbeat position, we shift the estimated beat grid to the left and perform local optimization over the offset position using the L-BFGS-B algorithm again.

Algorithm 3.1: Beat Grid Estimation Using 2D Constrained Minimization

Input : Detected beats t_i , where $i = 1, 2, \dots, n$
Output : Beat grid g_j , where $j = 1, 2, \dots, m$

- 1 tempo_estimate \leftarrow estimate_tempo_median(t_i);
- 2 boundaries \leftarrow $(0, \frac{60}{\text{tempo_estimate}} \times 1.4), (60, \text{tempo_estimate} \times 1.15)$;
- 3 init_guess \leftarrow $(t_1, \text{tempo_estimate})$;
// Global search to find a good solution
- 4 objective, offset, bpm \leftarrow dual_annealing_l_bfgs_b(t_i , boundaries, init_guess);
// Refined optimization with restricted boundaries
- 5 boundaries_restricted \leftarrow $(0, \frac{60}{\text{tempo_estimate}} \times 1.05), (\text{tempo_estimate} / 1.05,$
tempo_estimate $\times 1.05)$;
- 6 objective_r, offset_r, bpm_r \leftarrow dual_annealing_l_bfgs_b(t_i ,
boundaries_restricted, init_guess);
- 7 **if** *objective_r* < *objective* **then**
- 8 | objective, offset, bpm \leftarrow objective_r, offset_r, bpm_r;
- 9 **endif**
- // Local optimization for downbeat position
- 10 beat_duration \leftarrow $\frac{60}{\text{bpm}}$;
- 11 offset_bounds \leftarrow $(0, \frac{60}{\text{tempo_estimate}} \times 1.4)$;
- 12 offset_cand \leftarrow offset - beat_duration;
- 13 **while** *offset_cand* ≥ 0 **do**
- 14 | objective_offset, offset_new \leftarrow local_minimize_l_bfgs_b(t_i , offset_bounds,
offset_cand, bpm);
- 15 | **if** *objective_offset* < *objective* **then**
- 16 | | objective, offset \leftarrow objective_offset, offset_new;
- 17 | **endif**
- 18 | offset_cand \leftarrow offset_cand - beat_duration;
- 19 **endw**
- 20 $g_j \leftarrow \text{offset} + j \times \frac{60}{\text{bpm}}$ for $j = 1, 2, \dots, m$;
- 21 **return** g_j ;

3.1.1 Benchmark

We selected the GiantSteps dataset [Kne+18; SM18] to evaluate the performance of our beat grid and tempo estimation algorithm, as neither BeatNet [HCD21] nor current state-of-the-art tempo estimations such as by Böck and Davis [BD20] have been trained on this dataset. The metrics *Accuracy 1* and *Accuracy 2*, as defined by Gouyon et al. [Gou+06], are mainly used to evaluate the performance of tempo estimation algorithm [BD20; FP19; Kne+18]. *Accuracy 1* measures the percentage of correctly estimated tempos within a tolerance of 4% of the ground truth tempo. *Accuracy 2* is similar to *Accuracy 1*, but allows for octave errors ($\frac{1}{2}$, $\frac{1}{3}$, 2, 3 multiple of the tempo).

Slight deviations in the estimated tempo lead to significant errors in the beat grid estimation. To compare the performance of our tempo estimation algorithm for small tolerance windows, we define *Accuracy 1* and *Accuracy 2* for the tolerance windows of 1% and 0%.

Table 3.1 shows the comparison of our tempo estimation algorithm with the state-of-the-art tempo estimation algorithm by Böck and Davis [BD20] on the GiantStep. While our algorithm does not outperform the state-of-the-art algorithm for the 4% tolerance window, it demonstrates better performance for the 1% and 0% tolerance windows.

	Böck and Davis [BD20]	Ours
Accuracy 1 (4%)	87.29	82.30
Accuracy 1 (1%)	67.02	69.59
Accuracy 1 (0%)	0.15	19.97
Accuracy 2 (4%)	96.97	90.77
Accuracy 2 (1%)	74.38	76.70
Accuracy 2 (0%)	0.45	24.51

Table 3.1: Comparison of our tempo estimation algorithm with a state-of-the-art tempo estimation algorithm by Böck and Davis [BD20] on unseen data from the GiantSteps dataset.

Figure 3.1 further visualizes the comparison of the two algorithms for errors below 4%. Up until around 162 samples out of 661, our algorithm predicted the tempo with 0% error, whereas Böck and Davis’ algorithm had an average error of 0.184% and a median error of 0.20%. After 207 samples, the current accuracy of our algorithm per sample decreases below Böck and Davis’ algorithm but stays nearly constant until ~ 500 samples, surpassing Böck and Davis below the 1% error window.

Since the accuracies for the 0% and 1% tolerance windows are more important for the beat grid estimation, we conclude that our tempo estimation algorithm is more suitable for the beat grid estimation than Böck and Davis’.

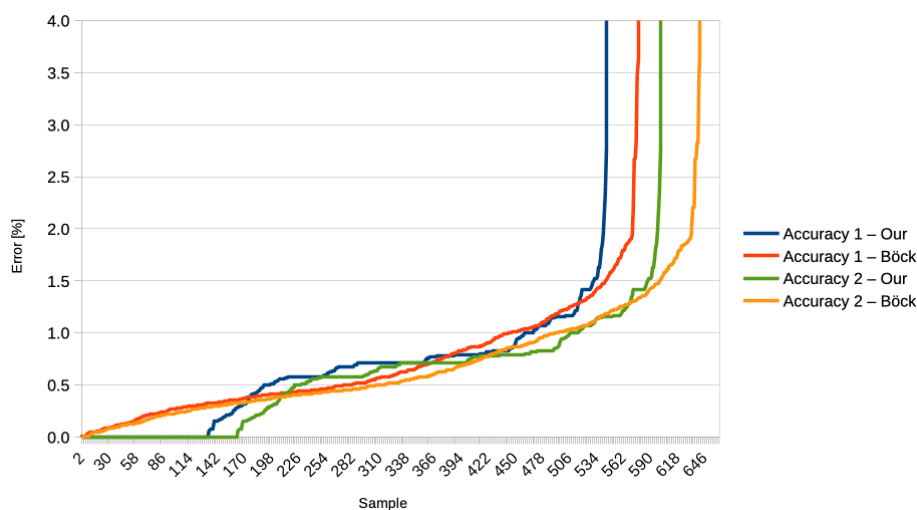


Figure 3.1: Comparison of our tempo estimation algorithm with a state-of-the-art tempo estimation algorithm by Böck and Davis [BD20] on the GiantSteps dataset for errors below 4%.

3.2 Structural Segmentation

Music transitions sound most pleasing when performed at musically meaningful positions of a song, such as the beginning of a bridge, chorus, verse, etc. We will use boundary detection and labeling algorithms to group similar segments together and detect these musically meaningful positions.

We found that the boundary detection algorithm by Serrà et al. [Ser+14], as described in section 2.4, performed most accurately on EDM music. In addition, the labeling approach by Nieto and Bello [NB14] (cf. section 2.4) performed best on the segments detected by Serrà et al.’s approach. The implementation of *MSAF* [NB16] was used for both algorithms. Figure 3.2 shows the result of the segmentation algorithm on a popular EDM song. The boundary detection algorithm separated the song into nine segments. The labeling algorithm grouped the segments into four groups, as depicted by the different colors in the figure. The intro segment (first, visualized in purple) has the same label as two other segments, for which we can conclude that they are similar in terms of their musical content. The outro segment (last, visualized in blue) is also evident. Classifying the remaining segments into structural segments (chorus, verse, bridge, etc.) would require a rule-based approach, which would differ for each sub-genre. Thus, we will not attempt to classify the structural segments and leave this task to the similarity algorithm, which will choose the best-fitting segment for the transition.

In electronic music, segments typically start and end at downbeat positions. We, therefore, first quantize the detected segment boundaries to the nearest beat position. To counteract segment boundaries that have been detected one beat too early or too late, we shift them

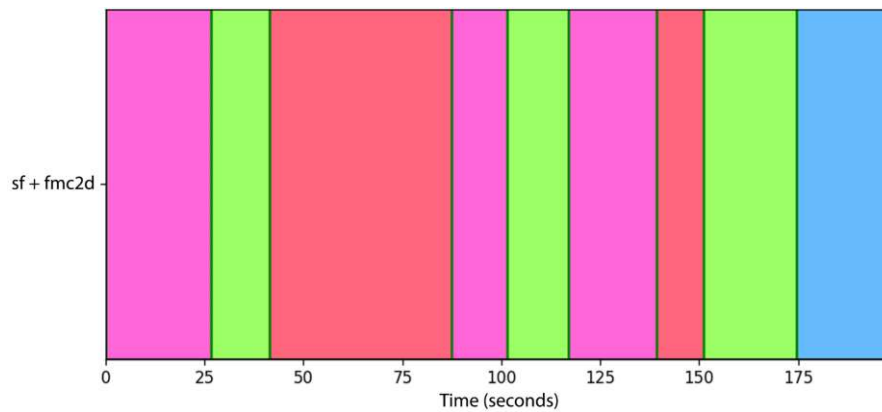


Figure 3.2: Song segments where the boundaries were detected by Serrà et al.’s algorithm [Ser+14] and the segments were labelled by Nieto and Bello’s algorithm [NB14]. Segments are separated by a vertical line. The color of the segments represents the label of the segment. The song is *Falling* by *Camo & Krooked*.

by one beat position to the nearest downbeat. Boundaries that start or end at the third beat positions will not be shifted, as the cause of it might have multiple reasons, such as errors in the downbeat detection, time signature estimation, or different song structures.

Although mixing intros of songs with outros is a straightforward way of transitioning whole songs, we will abstain from this practice as we aim for a more energetic mix. Thus, we penalize intro and outro segments by the factor 0.5, which is then multiplied by the similarity measure. The progression of the energy level is a task for the similarity measure. We assume that low-energy and high-energy segments will not be mixed and thus will not differentiate between other segment types.

3.3 Music Similarity

In this section, we will propose and detail our music similarity and mixability calculations and song scheduling algorithm.

3.3.1 Rhythmic Similarity

The rhythmic similarity was modeled in various ways in the literature, as described in subsection 2.5.1. A common approach is to use onset detection functions for specific instruments. Besides being an essential component in audio similarity, we hypothesize that the rhythm is a measure that can also be used to detect the accuracy of the beat grid estimation. A low rhythmic similarity score could indicate a tempo error and, thus, a beat grid error.

We believe that the drums are the primary rhythmic component in EDM music. Instead of relying on onset detection functions, which have poor performance in polyphonic audio,

we decided to use the drum transcription system by Southall et al. [SSH16; SSH17] to extract drum patterns from the audio. To be able to detect different kinds of rhythm patterns besides the classical "straight" pattern, such as "swing", "shuffle" or "offbeats" which are a primary component in EDM subgenres such as drum and bass, we follow the AutoMashUpper (AMU) approach of Davies et al. [Dav+14] and sub-divide the beat grid into 12 equally spaced intervals. We then detect the *kick*, *snare*, and *hi-hat* drum positions and quantize them over the sub-beat grid. By stacking the *kick*, *snare*, and *hi-hat* information on top of each other, we obtain a 3-dimensional binary vector R_n for all songs n of length $K * 12$, where K is the number of beat positions of a song. The rhythmic similarity is then calculated between phrase sections p of the seed song s and a candidate song c for all the beat shifts k beat shifts of c . While AMU used cosine-similarity as a rhythmic similarity measure, we decided to employ a stricter similarity measure to capture dissimilarities in the drum patterns. Thus, we define the similarity measure as the average of the sub-beat positions where the drum patterns of the seed song section and the candidate song section match.

For each drum vector $R_{s,p,d}$ within phrase section p of the seed song s , where $d \in 1, 2, 3$ denotes the drum vector dimensions corresponding to the *kick*, *snare*, and *hi-hat*, we compute the average number of matching sub-beat positions l over all beat shifts k against all candidate songs c . The overall rhythmic similarity $M_{R,s}(k)$ is then derived by averaging the similarities obtained across the three drum dimensions d ,

$$M_{R,c}(k) = \frac{1}{3} \sum_{d=1}^3 \left(\frac{1}{m} \sum_{l=1}^m R_{s,p,d,l} = R_{c,k,d,l} \right), \quad (3.3)$$

where m is the length of the drum vector $R_{s,p}$ of the phrase section in the seed song.

3.3.2 Timbral Similarity

Songs played in a DJ set have been identified to be similar in terms of their timbral content [KT13]. To model the timbral component, we will follow the approach of Rocha et al. [RBH13] and Panteli et al. [Pan+17]. Our timbral component will consist of Mel-frequency cepstral coefficients and the auditory descriptors: spectral flatness and dirtiness. We will use the MFCCs to represent the spectral envelope of the audio. Analogous to Panteli et al., we will use the first 20 MFCCs, following Aucouturier et al. [APS05], who found this number to optimally balance capturing the broad shape of the spectral envelope without over-focusing on finer spectrum details. The MFCCs are computed in half-overlapping frames of 1 beat duration and averaged over all frames.

Spectral flatness is a feature that reflects the tonal character and compression potential in the audio coding of an audio signal [HAH01]. It measures how smooth or spiky a signal's spectrum distribution is by calculating the ratio between its geometric mean and arithmetic mean [LT07]. We compute the spectral flatness in half-overlapping frames of 1 beat duration and then average it over all frames. The computation is done over four equally spaced frequency bands within the 300 to 6000 Hz range [HAH01].

Dirtiness (or auditory roughness) is a feature that reflects the de-tuning that producers apply to their synthesizer sounds. In contrast to Panteli et al., who used the algorithm by Vassilakis [Vas01], we decided to use the more recent approach [Vas07] by the same author, which outperforms the previous model. We then computed the dirtiness over the same frequency bands as the spectral flatness.

By stacking the MFCCs, spectral flatness, and dirtiness information on top of each other, we obtain a 28-dimensional vector $T_{n,p}$ for a song n and phrase section p . Due to the computational demand of the timbre calculation, it was infeasible to calculate the timbral component for all beat shifts k of the candidate song c , similar to the rhythmic similarity (c.f. subsection 3.3.1). We believe that the timbral component is more or less constant over phrase sections and thus calculate the timbral component over each phrase section p of the candidate song c . The timbral similarity is then calculated by computing the cosine-similarity between the timbral component $T_{s,p}$ of phrase section p of seed song s and the timbral component $T_{c,q}$ of all phrase sections q of candidate song c ,

$$M_{T,c}(q) = \frac{T_{s,p} \cdot T_{c,q}}{\|T_{s,p}\| \|T_{c,q}\|}. \quad (3.4)$$

3.3.3 Key Similarity

Harmonic compatibility is essential when mixing songs, as it avoids dissonance and leads to a more pleasing blend.

We decided to use the key detection algorithm *KeyFinder* [Sha11] due to its open-source availability and good performance compared to recent state-of-the-art key detection algorithms. Additionally, we will incorporate pitch shifting in the song selection process to be more flexible and less constrained by the harmonic aspect of the songs. Pitch-shifting algorithms, however, can hurt the quality of audio as they introduce a variety of artifacts. These artifacts include *detuning*, where specific frequencies are shifted a bit more than others, and *clipping*, where a signal that exceeds the maximum value is cut off at its maximum, among other problems mentioned by Royer [Roy19]. Figure 3.3 illustrates the clipping issue on two pitch-shifted audio excerpts by 1 and 2 semitones using the popular pitch shifting algorithm *RubberBand*³. Instead of allowing clipping and thus introducing even more issues, such as the generation of high-frequency components [Roy19], *RubberBand* normalized the audio to match the maximum amplitude of the signal where the clipping occurred. This, however, leads to a loss of the original audio's dynamics and, thus, a loss of audio quality.

We will base our key similarity measure on the wheel of fifths (cf. subsection 2.3.1). We define a harmonic key distance measure $D_{K_1}(K_2)$ as the minimum semitone distance between two keys K_1 and K_2 on the wheel of fifths. The key similarity measure $M_{K,c}$ is then defined as follows,

³<https://breakfastquay.com/rubberband>

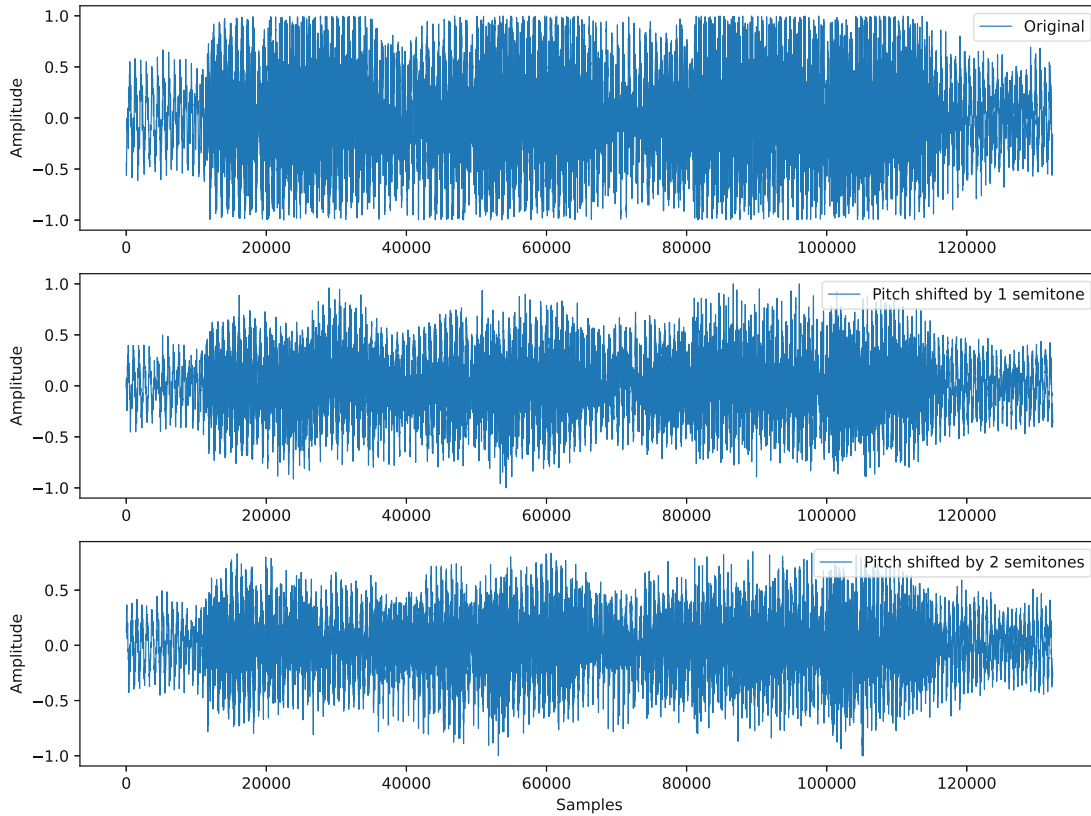


Figure 3.3: A three seconds excerpt from 03:66-06:66 of the song *Falling* by *Camo & Krooked* pitch shifted by 1 and 2 semitones using the pitch shifting algorithm *RubberBand*. The top plot shows the original audio, transformed into mono. The middle plot depicts the waveform of the original audio after pitch shifting by 1 semitone. The bottom plot shows the waveform of the original audio after pitch shifting by 2 semitones.

$$M_{K,c} = \begin{cases} 1, & \text{if } D_{K_s}(K_c) = 0 \\ D_{K_s}(K_c)^{-1}, & \text{otherwise} \end{cases}, \quad (3.5)$$

where $D_{K_s}(K_c)$ is the key distance between the key K_s of the seed song s and the key K_c of the candidate song c .

3.3.4 Harmonic Similarity

Harmonic content changes throughout a song and thus needs to be reflected in the similarity measure. We will complement our key similarity measure with a beat-wise harmonic similarity measure by following the approach of Davies et al. [Dav+14] and calculate the harmonic similarity between beat-synchronous chromagrams C_γ . We calculate beat-synchronous chromagrams $C_{c,k}$ across all beat shifts k over all candidate

songs c . The harmonic similarity is then computed by calculating the cosine-similarity between the beat-synchronous chromagram $C_{s,p,q}$ of the phrase section p of seed song s over all rotational key shifts q and the beat-synchronous chromagram $C_{c,k}$ of all beat shifts k for all candidate songs c ,

$$H_c(q, k) = \frac{C_{s,p,q} \cdot C_{c,k}}{\|C_{s,p,q}\| \|C_{c,k}\|}. \quad (3.6)$$

To improve the computational efficiency, we also followed the 2D-correlation approach of Davies et al. [Dav+14] to compute the harmonic similarity over all rotational key shifts q and beat shifts k more efficiently. To obtain the final harmonic similarity measure $M_{H,c}(k)$, we take the maximum harmonic similarity over all rotational key shifts q for all beat shifts k of the candidate song c ,

$$M_{H,c}(k) = \max_q (H_c(q, k)). \quad (3.7)$$

3.3.5 Spectral Balance

To ensure a balanced mix, we follow the approach by Davies et al. [Dav+14] and compute the spectral balance of the audio. We calculate the spectral representation L by computing the perceived loudness in a beat-synchronous fashion across the three frequency bands: the low band (< 220 Hz), the mid band ($220 - 1760$ Hz), and the high band (> 1760 Hz). We then compute spectral flatness β_k for every beat shift k of the candidate song c by averaging the spectral representations $L_{s,p}$ of the phrase section p of the seed song s and the spectral representation $L_{c,k}$ for the beat shift k of the candidate song c , over the beat dimension,

$$\beta_k = \frac{1}{K_p} \sum_{k=1}^{K_p} L_{s,p} - L_{c,k}, \quad (3.8)$$

where K_p is the number of beats in the phrase section p .

The spectral balance measure $M_{L,c}(k)$ is then defined as follows,

$$M_{L,c}(k) = 1 - std(\beta_k). \quad (3.9)$$

3.3.6 Contextual Similarity

Mixing songs at positions with similar lyrics is a transition technique that makes the transition, independently of audio-based similarity, more seamless. This technique is commonly executed by playing a repeated phrase of the first song and then mixing in the second song with a similar vocal phrase.

We will use the *Genius* API⁴ to obtain the lyrics of a given song. We found *Genius* to be the most complete, free-to-use source for lyrics. At the time of writing, no freely available API provided lyrics synced with the audio.

Since the lyrics are not synced with the audio, we will consider the lyrics as a whole and not in a beat-synchronous fashion. Because lyrics sometimes heavily differ for each paragraph, we find classical textual similarity measures such as **TF-IDF** or **BM25** unsuitable for our task. Instead, we will capture the lyrics' similarity by extracting the whole lyrics' semantic meaning.

We will use Reimers and Gurevych **RG19** approach (cf. **subsection 2.5.3**) to compute sentence embeddings over all sentences of the lyrics. To accelerate the computation, we pre-compute all sentence embeddings C_n over the lyrics of all songs n . The similarity measure $M_{C,c}$ is then calculated by computing the cosine-similarity between the sentence embedding C_s of the seed song s and the sentence embedding C_c of the candidate song c ,

$$M_{C,c} = \frac{C_s \cdot C_c}{\|C_s\| \|C_c\|}. \quad (3.10)$$

3.3.7 Mixability

We compute the beat-wise mixability for a candidate song c against the phrase section p of the seed song s by combining the weighted similarity measures of rhythm, timbre, key, harmony, and spectral balance, as follows:

$$M_c(k) = \omega_R M_{R,c}(k) + \omega_T M_{T,c}(q) + \omega_K M_{K,c} + \omega_H M_{H,c}(k) + \omega_L M_{L,c}(k), \quad (3.11)$$

where q is the phrase section of c corresponding to the beat shift k . The mixability measure considers the 64 beats after the phrase section p of the seed song s instead of the entire phrase section p . This forward-moving approach enables us to maintain the song's dynamics by focusing on the upcoming segments instead of past segments. Through extensive, informal testing, we found the following weights to give the most pleasing results: $\omega_R = 0.3$, $\omega_T = 0.75$, $\omega_K = 0.2$, $\omega_H = 0.2$, and $\omega_L = 0.1$.

To incorporate the contextual similarity measure, we extend the audio-based mixability measure $M_c(k)$ by the contextual similarity measure $M_{C,c}$ with the weight $\omega_C = 0.25$, as follows:

$$M'_c(k) = M_c(k) + \omega_C M_{C,c}. \quad (3.12)$$

Our initial experiments showed that choosing the transition point by selecting the beat shift k with the highest mixability score, such as done by Davies et al. **Dav+14**, did not lead to the most pleasing results. Songs were transitioned at non-downbeat positions or unnatural downbeat intervals (e.g., 7, 9, 15, 17 downbeats), leading to a misaligned mix. To counteract this, we consider only beat shifts k that correspond to the segment boundary q of the candidate songs c and calculate the transition (cue) point as follows,

⁴<https://genius.com>

$$k_{\text{cue}}(c) = \arg \max_{k \in q} M_c(k). \quad (3.13)$$

We also record the harmonic and rhythmic similarity at the transition point, and will use this information to improve the equalization in the mixing process,

$$h_{\text{cue}}(c) = M_{H,c}(k_{\text{max}}(c)), \quad (3.14)$$

$$r_{\text{cue}}(c) = M_{R,c}(k_{\text{max}}(c)). \quad (3.15)$$

We compute the song schedule by selecting the candidate song c with the highest mixability and extract the phrase section p for c . The phrase section p is played until the corresponding segment boundary q of c is reached, but at least for λ_{minPlay} . We found that a λ_{minPlay} value of 55 seconds leads to a good balance between how long a song is played and how often songs are changed. We then select the phrase section p of c as the seed phrase section and repeat the process until the desired length of the mix is reached.

A visualization of the mixability and its features per beat shift for a given candidate song is depicted in [Figure 3.4](#). The timbral feature is computed over segments and thus constant over them. The key similarity is constant over all beat shifts, as it is computed over the whole song. Notable are the peaks in the rhythmic similarity every four beats, which are explained by the alignment of the drum pattern of both songs on the downbeat.

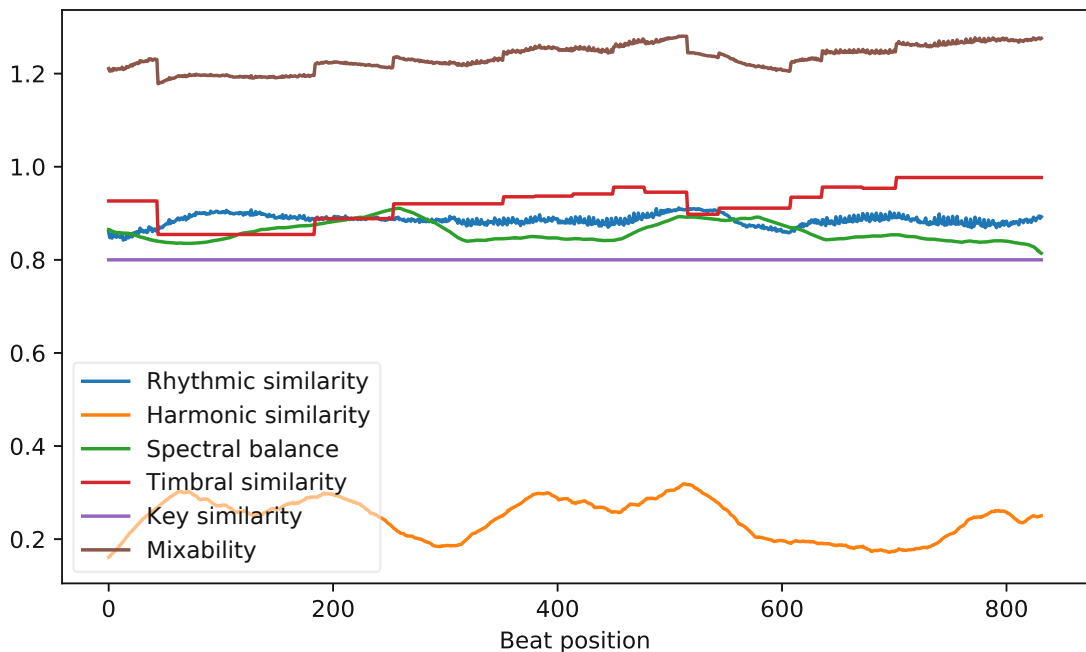


Figure 3.4: Mixability and its features per beat shift for a candidate song.

3.4 Rule-based Stem Modification

Mixing two songs is a challenging task involving multiple steps. The beat-matching process is one of the most important steps, which, if done incorrectly, can lead to a displeasing mix. Beat-matching aligns the beats of two song excerpts to be played in sync. As we already found out in [subsection 3.1.1](#), our tempo estimation algorithm predicts the tempo of only 25% of songs with perfect accuracy. Slight errors in tempo estimation can lead to misalignments in the beat grid, leading to a slightly off-beat mix. Another important rule in mixing is to mix song excerpts that both contain vocals with care or to avoid mixing them altogether, as vocal clashing can similarly lead to a reduced mix quality.

To counteract these two cases, we will separate the individual audio stems of the songs and remove the troubling stems based on a rule-based approach. We will use the pre-trained *HT Demucs* model [\[RMD23\]](#); [\[Déf+21\]](#) (cf. [section 2.6](#)) to separate the audio into the four stems: vocals, drums, bass, and other. Based on the rhythmic similarity value $r_{\text{cue}}(c)$ of the candidate song c at the transition point, we will employ drum stem modifications to counteract slight tempo errors and drum pattern incompatibilities, as later in-detail described in [section 3.5](#). Based on several experiments, we found that applying the drum stem modification procedure at a rhythmic threshold below $r_{\text{thresh}} = 0.95$ leads to the most pleasing results.

We take a similar approach to the vocal segments of the songs. First, we have to detect the vocal segments in the song. Instead of relying on [\[RNN\]](#) [\[LSW18\]](#) or [\[CNN\]](#) [\[YLL21\]](#) architectures for vocal detection over polyphonic audio, we will use the vocal stem obtained from the [\[MSS\]](#) step to detect the presence of vocals. We detect vocal segments by splitting the vocal stem into boundaries on "silent" sections that persist for one second or longer and where the loudness is below -40 dBFS. We then filter out every vocal segment shorter than 400ms, as we assume that vocals present in such short segments are not the main vocals of the song and, thus, do not interfere with the mix. If the vocals during the transition intersect for more than two seconds, we apply a vocal stem modification procedure to counteract vocal clashing (cf. [section 3.5](#)).

3.5 Mixing

The mixing pipeline consists of multiple stages to ensure a high-quality mix and good generalization on various songs. To achieve a good generalization of transition quality, we define the transition length as 16 downbeats. The transition starts with eight downbeats before the song's end and ends with eight downbeats after the transition point of the current song. We apply various equalization techniques and stem modification procedures described in the following sections to ensure a smooth transition.

Before transitioning, we first need to bring the loudness of each song to a consistent level. We achieve this by measuring the perceived loudness of an audio and adjusting its gain to -14 [\[LUFS\]](#). To calculate the perceived loudness, we use the *ITU-R BS.1770-*

4 algorithm [Int15], which analyzes the energy of the audio over time, using specific frequency weighting and temporal integration to account for human hearing sensitivity. We then beat-match the song by time-stretching the audio to the same tempo as the previous song. To avoid a loss in the song’s dynamic due to too big tempo changes, we only consider songs with a time-stretching factor of $\pm 8\%$ as candidates. Afterward, we pitch-shift the audio to the harmonically compatible key, as computed during the key similarity step in subsection 3.3.3. Time stretching and pitch shifting is done utilizing the *RubberBand*⁵ algorithm.

3.5.1 Filter Design

Selecting the right filter type is crucial for the quality of the equalization process. An ideal filter would have a perfectly flat passband and an infinitely sharp cutoff, completely attenuating all frequencies outside the desired range without introducing phase distortion. However, such filters do not exist due to physical limitations and the principles of causality. Furthermore, achieving complete frequency blocking and instantaneous signal response is impossible, as it would require an infinite impulse response and, if implemented digitally, an infinite number of coefficients. [Dro07a] [Orf96, Chapter 10]

Since the ideal filter can not be realized, we must take a trade-off between a small transition band of the filter, referred to as steep roll-off, and a favorable phase response [Dro07b]. We consider a linear (flat) phase response as the most favorable one, as their phase delay is independent of frequency, meaning that all frequency components of the signal are delayed by the same amount of time, preserving the shape of the waveform [Orf96, Chapter 6].

Table 3.2 depicts the different trade-offs of common filter types based on [Dro07b]. The Bessel and Elliptic filters are on the extreme end of the trade-off spectrum. The Bessel filter achieves a maximally flat phase response at the cost of a steep roll-off. The Elliptic filter, on the other hand, achieves a very steep roll-off at the cost of ripples in both the passband and stopband. The Chebyshev filter only has a ripple on either the passband or the stopband, but not on both, at the cost of a less steep roll-off than the Elliptic filter. The Butterworth filter is in the middle of the trade-off spectrum, with a moderate roll-off and a moderate phase response.

Filter	Roll-off	Phase response
Bessel	–	++
Butterworth	\pm	\pm
Chebyshev	+	–
Elliptic	++	--

Table 3.2: Overview of different filter types’ roll-off and phase response characteristics.

⁵<https://breakfastquay.com/rubberband>

We will utilize the Butterworth filter, as it has a moderate roll-off and a moderate phase response, which is a good compromise for the equalization process. A low-pass Butterworth filter is characterized by the magnitude-squared frequency response

$$|H(\Omega)|^2 = \frac{1}{1 + \left(\frac{\Omega}{\Omega_c}\right)^{2N}}, \quad (3.16)$$

where N is the order of the filter and Ω_c is the cutoff frequency, characterized by a -3dB attenuation [PM95]. The frequency response of a low-pass Butterworth filter for the first seven orders and a cutoff frequency of 2800 Hz is depicted in Figure 3.5. We decided to use a 5th-order Butterworth filter, as it has a steeper roll-off compared to lower orders and does not attenuate lower frequencies as much as lower orders while still having a moderate phase response.

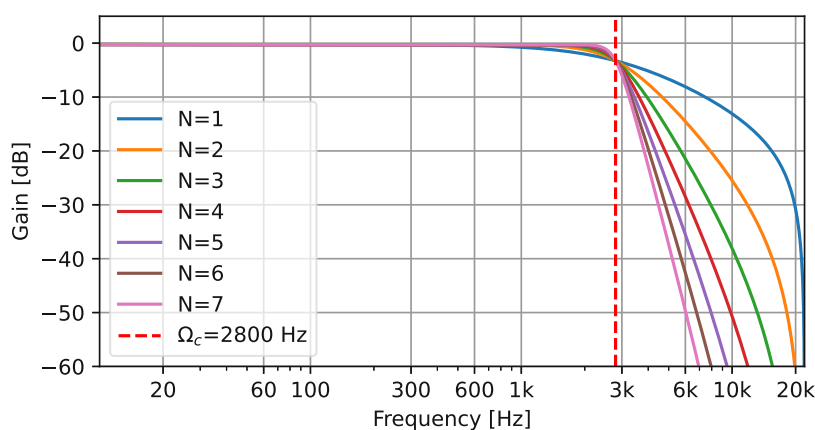


Figure 3.5: Frequency response of a low-pass Butterworth filter for the first 7 orders and a cutoff frequency of 2800 Hz .

Most DJ mixers have a three-band equalizer for the low, mid, and high frequencies. We model our low-frequency filter as a low-pass filter with a cutoff frequency of 140 Hz and model our high-frequency filter as a high-pass filter with a cutoff frequency of 2800 Hz . To attenuate the mid frequencies, we use a band-stop filter with a stopband of 140 Hz to 2800 Hz . Figure 3.6 depicts the frequency response of our low-, mid- and high-shelving filters. Even though the cutoff frequencies have a maximal attenuation of -6dB in case of the simultaneous application of either low- and high-frequency filter in combination with the mid-frequency filter, we decided not to enlarge the stopband of the mid-frequency filter. If we had enlarged the stopband, it would have over-attenuated the low and high frequencies, leading to a more complex equalization process. Furthermore, we will apply the mid-filter only to partly attenuate mid frequencies in specific scenarios, as described in the following sections. Thus, there is no need to enlarge the stopband of the mid-frequency filter.

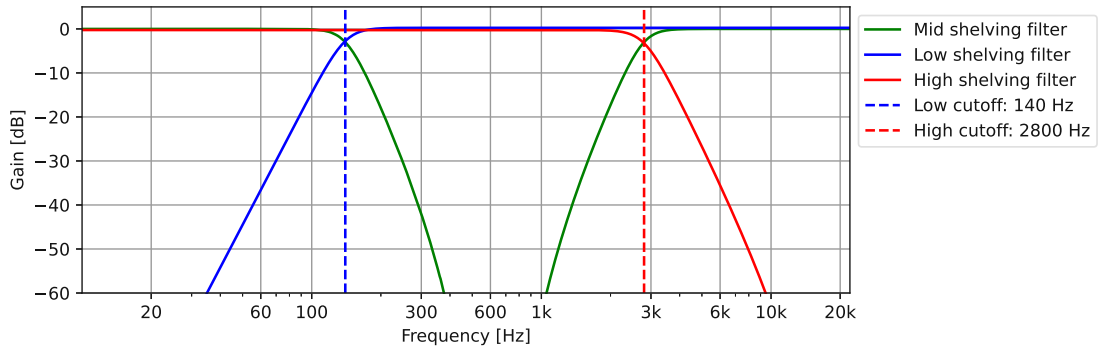


Figure 3.6: Frequency response of the low-, mid- and high-shelving filters.

When mixing songs, a DJ applies equalization in a time-varying way by turning the low, mid, and high equalization knobs over time, in the case of a three-band equalizer. Instead of directly applying our filters to the audio signal, which could introduce sudden and unpleasant changes to the audio, we will transform our filters into linear time-varying shelving filters, emulating the real-world equalization process. To achieve this computationally efficiently, we apply a shelving filter with maximal gain attenuation on the audio signal by employing a passband and stopband filter and mixing the original and filtered audio signal linearly. Our algorithm to apply a time-linear Butterworth shelving filter to the audio signal for a given start gain factor s and end gain factor e , where 0 is complete attenuation, and 1 is no attenuation, is depicted in [algorithm 3.2](#). After the time-linear filter is applied, we normalize the audio to prevent clipping caused by the filter's phase response.

Algorithm 3.2: Time linear Butterworth shelving filter algorithm

Input : Audio y , Filter f , Start factor s , End factor e

Output : Equalized audio y_{eq}

```

1  $y_{eq} \leftarrow f(y)$ ;
2 for  $i \leftarrow 1$  to  $|y|$  do
3    $b \leftarrow 1 - s + (s - e) \times \frac{i}{|y|}$ ;
4    $y_{eq}[i] \leftarrow (1 - b) \times y[i] + b \times y_{eq}[i]$ ;
5 endfor
6  $y_{eq} \leftarrow \text{normalize}(y_{eq})$ ;
7 return  $y_{eq}$ ;

```

3.5.2 Equalization

Equalization techniques differ across music genres, depending on the song dynamics and the DJ's personal taste. We propose a generic equalization approach to resolve the most common issues in mixing [EDM](#) music.

A common technique in mixing **EDM** music is the "bass-swap". When mixing two songs with strong basslines, the basslines can clash and lead to a "muddy" mix. The newly introduced song is mixed in with heavily attenuated bass frequencies to counteract this. On the point where the newly introduced song is planned to become dominant, the bass frequency attenuation is swapped with the one that was already playing. The same technique is commonly applied to the high frequencies but with lower attenuation, as the high frequencies are commonly less dominant. Equalizing the mid frequencies heavily depends on the song itself and if the mid frequencies "clash". [Ste10, Chapter 16]

We designed the equalization process using that information, as depicted in **Figure 3.7**. The song that is to be mixed is introduced over four downbeats. During this time, the bass frequencies are heavily attenuated. The highs, on the other hand, are continuously increased over the four downbeats. Four downbeats before the transition point, the high frequencies are further increased while the bass frequencies are only slightly increased. Half a beat before the transition point, we bass swap the two songs until the transition point and switch the high frequencies. The high and low frequencies of the song to be mixed out are then slightly decreased until four downbeats after the transition and kept constant until the end. During the last four downbeats, the song fades out by decreasing its gain.

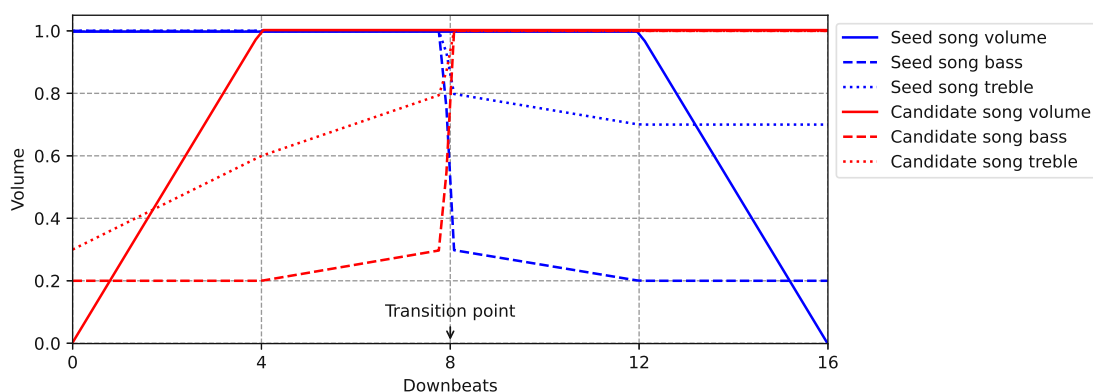


Figure 3.7: Equalization applied to both audio excerpts, in case of low rhythmic similarity, low timbral similarity and no drum stem modification.

Since we extracted a variety of information about the audio during our mixability calculation stage, we can improve the equalization process by detecting problematic frequency ranges and treating the separate song stems differently. We consider mid frequencies of songs with a dissimilar timbre, i.e., timbral similarity below $t_{\text{thresh}} = 0.95$, as clashing and reduce the mid frequencies of the song that is currently playing, shifting the focus on the mid frequencies to the new song. During the half beat until the transition point, we attenuate the mid frequencies to half of the original gain to prevent "muddiness" in the mix. We additionally assume that rhythmically similar songs are less likely to clash in the lower frequencies and thus need less attenuation in the bass frequencies. We realize this by using slightly less attenuation in the bass frequencies of the song that is

to be mixed in and even less attenuation in the bass frequencies for the song that is to be mixed out. This preserves more of the original song’s dynamics and leads to a more pleasing mix. Finally, we also assume that songs with an attenuated drum stem need even less equalization in the high frequencies, as drums, especially hi-hats, are a primary contributor to the high frequencies. We, therefore, introduce the high frequencies of the song that are to be mixed in earlier and with less attenuation. Figure 3.8 depicts this equalization process for the case of high rhythmic similarity, low timbral similarity, and the application of drum stem modification.

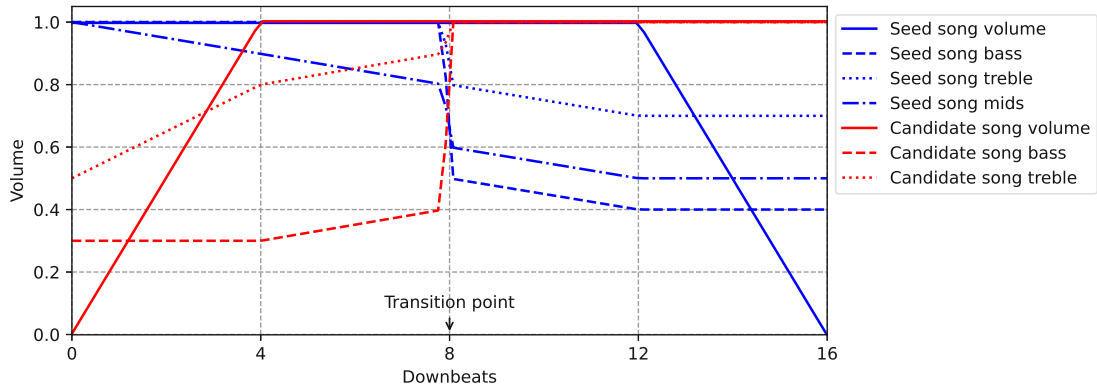


Figure 3.8: Equalization applied to both audio excerpts, in case of high rhythmic similarity, high timbral similarity and drum stem modification.

3.5.3 Stem Equalization

Mixing rhythmically incompatible songs is usually implicitly avoided by the rhythmic component of our mixability measure. However, since the rhythmic component is only one of many components, a song with an incompatible drum pattern may be selected for mixing. Excluding these songs from the scheduling process would significantly reduce the variety of songs that can be mixed together. We, therefore, introduce a drum stem modification procedure to counteract rhythmic incompatibility and consider songs with a rhythmic similarity below $r_{\text{thresh}} = 0.95$ as rhythmically incompatible. We counteract the audibility of clashing incompatible drum patterns by introducing the new song without the drum stem and introducing the drum stem gain to 80% until half a beat before the transition point. While doing so, we also attenuate the drum stem of the song that is currently playing to limit the combined drum components gain to around 110% during the transition. During the same half-beat duration in which the bass swap takes place, we reintroduce the drum stem to its original gain while slowly fading out the drum stem of the song to be mixed out without fully attenuating it to prevent phase distortion.

We apply the same principle to the vocal stems to prevent vocal clashing. We decided to attenuate the vocals of the currently playing song, as we believe that the vocals of the song that is to be mixed in are more important in creating a more coherent

transition. Therefore, we attenuate the vocals of the current song playing entirely until four downbeats before the transition point. The vocals during the four downbeat attenuation period should clash only minimally, as the song to be mixed in is already introduced in a highly attenuated way. The drum stem and vocal removal filters are depicted in [Figure 3.9](#).

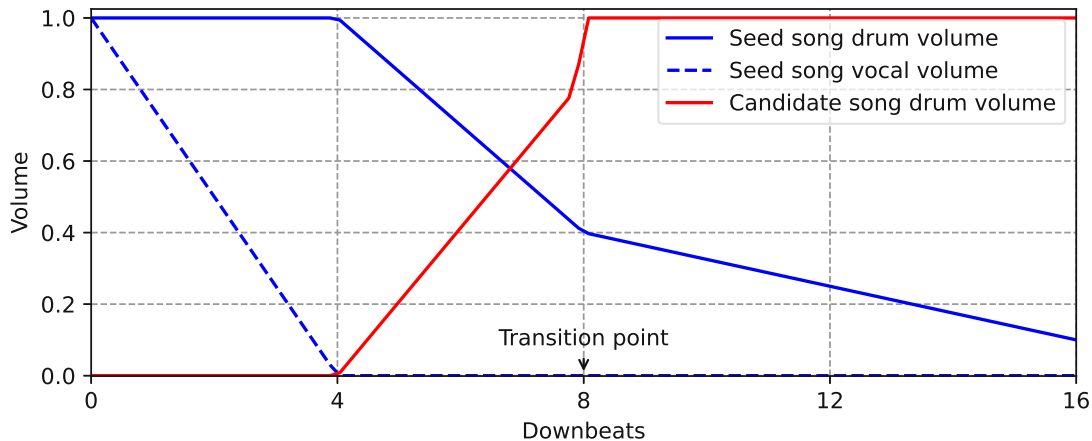


Figure 3.9: Equalization applied to the vocal and drum stems to prevent vocal clashing and enable mixing of rhythmically incompatible songs.

3.6 Models

In [section 3.4](#), we proposed our rule-based stem modification approach, and in [subsection 2.5.3](#), we introduced contextual information into our mixability measure. Given the assumption that stem modification mainly improves upon the mixing process without necessarily changing the song schedule, we define the following three models to assess the contribution of each approach:

- *Mosaikbox* (MB_{base}) - Our base model, without the stem modification and contextual information.
- *Mosaikbox + Stem Modification* (MB_{stem}) - Our model with the stem modification approach.
- *Mosaikbox + Stem Modification + Contextual Information* (MB_{full}) - Our model with both the stem modification and contextual information approach.

3.6.1 Baseline

To ensure that our base model MB_{base} is a good starting point for the evaluation, it needs to be compared to a baseline model. We chose [AutoMashUpper \(AMU\)](#) by Davies

3. PROPOSED METHOD

et al. [Dav+14](#) as our baseline model, as it is the most similar state-of-the-art approach that works on audio-based similarity measures.

To ensure a fair comparison of the models, we replace outdated components of [AMU](#) with more recent state-of-the-art approaches, as used in our approach. In particular, we replace their beat tracking and percussion detection approach with our approaches and utilize our mixing procedure to create the mix. This improved version of [AMU](#) shifts the focus of the evaluation to the song scheduling and removes the negative influence of mismatched beats.

Implementation

This chapter provides insights into the implementation of our automatic music-mixing models. We will first give an overview of the used technologies and frameworks in [section 4.1](#). Then, we will discuss design decisions regarding our software architecture in [section 4.2](#). To give insights into the runtime of our system and possible bottlenecks, we will provide performance metrics in [section 4.3](#). Finally, we will present and explain the web-based user interface for the automatic mix generation in [section 4.4](#). The complete source code of the implementation is available on [GitHub](#)¹.

4.1 Packages and Frameworks

Due to the wide availability of packages and [DNN](#) models, we decide to use *Python 3.11* as the main programming language for the project. We use a variety of packages for the song analysis and scheduling procedure. For the loading of audio files and the computation of the [STFT](#), [MFCCs](#), and chromagrams we use *librosa 0.10.1* [\[MMF+23\]](#). For data manipulation, we utilize *numpy 1.23.5*. The segmentation, specifically the boundary detection and segment labeling, is realized using *MSAF 0.1.80* [\[NB16\]](#). To improve the accuracy of the segmentation, we supply *MSAF* with our beat grid using *JAMS 0.3.4* and disable its internal beat tracking. Beat positions are derived using a slightly modified version² of *BeatNet 1.1.0* to ensure compatibility with other packages. For [ADT](#) we utilize *ADTLib 2.1.2*³ with its *madmom* dependency bumped to the commit 0551aa8 to fix a variety of incompatibilities. Lyrics are retrieved using the *Genius API* client *lyricsgenius* with its version bumped to the commit aac9dad to be able to strip metadata from the response. The semantic similarity between the lyrics is then computed using *sentence-transformers 2.3.1* [\[RG19\]](#). Stems are separated using *Demucs*

¹<https://github.com/robaerd/mosaikbox>

²<https://github.com/robaerd/BeatNet>

³<https://github.com/CarlSouthall/ADTLib>

4.0.1 [Déf21](#). The key of the song is estimated using the *C++* implementation of the *KeyFinder* [Sha11](#) algorithm from *libkeyfinder* ⁴.

For mixing song excerpts we utilize *pydub 0.25.1*. As a digital signal processing (DSP) library for the filter creation, we use *scipy 1.11.4*. For loudness normalization, we use *pyloudnorm 0.1.1* which implements the *ITU-R BS.1770-4* standard. Pitch shifting and time stretching are realized using *pyrubberband*. Due to dependency issues with the currently published *pip* package, we bumped the version to the commit `10634aa` to establish compatibility.

4.2 Software Architecture

Our architecture is designed with scalability and maintainability in mind. Since an important component of our system is available only as a *C++* library, we employ a combination of a layered and service-oriented architecture. This enables us to have a "core" system that contains the most important functionality and bridges the gap to the *C++* library. Additionally, we can outsource resource-demanding tasks to separate services for horizontal scalability.

4.2.1 Core System

The Python "core" system is designed using a classical 3-layer architecture and is exposed as a [REST API](#) using *FastAPI*. The system consists of three main procedures: the song analysis, the scheduling, and the mixing.

During the song analysis, as much pre-computation as possible is performed to avoid redundant computation in the scheduling process. First, for each of the songs, we download its lyrics and perform the four stems separation. Then, we compute the key, the beat grid, the segments, and their timbre vectors, the semantic lyrics embeddings, transcribe the percussions and detect vocal segments of the audio for all songs in parallel. The results are then stored on the file system and will be used by the scheduling procedure.

The scheduling procedure compares each currently scheduled song with all other remaining songs and selects the song excerpt with the highest mixability as the succeeding song. We use a combination of in-memory and on-disk caching to speed up the loading and time-stretching of audio files. Additionally, we cache intermediate results of the mixability computation, such as the beat synchronous chroma vectors.

The mixing procedure takes the song schedule as input, which contains all the necessary mixability scores to make rule-based mixing decisions, and sequentially creates the final mix. The mix is then exported and made available to the user as an MP3 file with a bitrate of 320 kbps and a sample rate of 44.1 kHz.

⁴<https://github.com/mixxxdj/libkeyfinder>

4.2.2 KeyFinder Service

To make the *C++* implementation of the *KeyFinder* algorithm available to our Python system, we model it as a separate service. The service is written in *C++* and exposes a [remote procedure call \(RPC\)](#) interface to the core system. Since our core system accepts various audio formats, we send the [pulse-code modulation \(PCM\)](#) encoded audio data to the service to reduce additional audio decoding dependencies.

The service then computes the key using the *libkeyfinder* library, converts it into the *Camelot* notation, and sends it back to the core system.

We use *gRPC* as the communication protocol due to its efficient communication and programming language agnostic nature. The *proto* file containing the service definition is shown in [Listing 4.1](#).

```

1  syntax = "proto3";
2  package keyfinder;
3
4  message KeyRequest {
5      bytes pcm_data = 1;
6      int64 channels = 2;
7      int64 frame_rate = 3;
8  }
9
10 message KeyResponse {
11     string key = 1;
12 }
13
14 service KeyFinder {
15     rpc GetKey (KeyRequest) returns (KeyResponse);
16 }

```

Listing 4.1: Protobuf definition for KeyFinder service

4.3 Performance

Although the song analysis procedure is computationally expensive, it has a linear runtime and offers a parallelism degree of n , where n is the number of songs to be analyzed. Since we compare each song with all other available songs during the scheduling procedure, we would need to compute the similarity between $\frac{n(n+1)}{2}$ songs, leading to a time complexity of $O(n^2)$. Even though there are various ways to reduce the time complexity, they all come at the cost of the optimality of the mix. Thus, we decided to consider the quadratic runtime and compute the optimal mix. Approaches to reduce the time complexity at the cost of the optimality of the mix include:

- Randomly sample k songs out of the available n songs and compute the similarity between all k songs. Repeat until n songs are scheduled. This leads to a linear

runtime of $O(k) \times \frac{n}{k} = O(n)$.

- Select the top m songs with the highest mixability score instead of comparing all songs with each other, resulting in a linear time complexity of $O(\frac{n+nm}{2}) = O(n)$.

To give an overview of the runtime in a real-world setting, we randomly sample sets of 5, 10, 15, 20, 25, and 30 EDM songs and compute a mix, using all these songs with our MB_{full} model. We repeat the experiment five times to ensure stable results and take the average runtime. The computation is carried out on a MacBook Pro with an M1 Max and 32 GB of memory. The results are depicted in Figure 4.1.

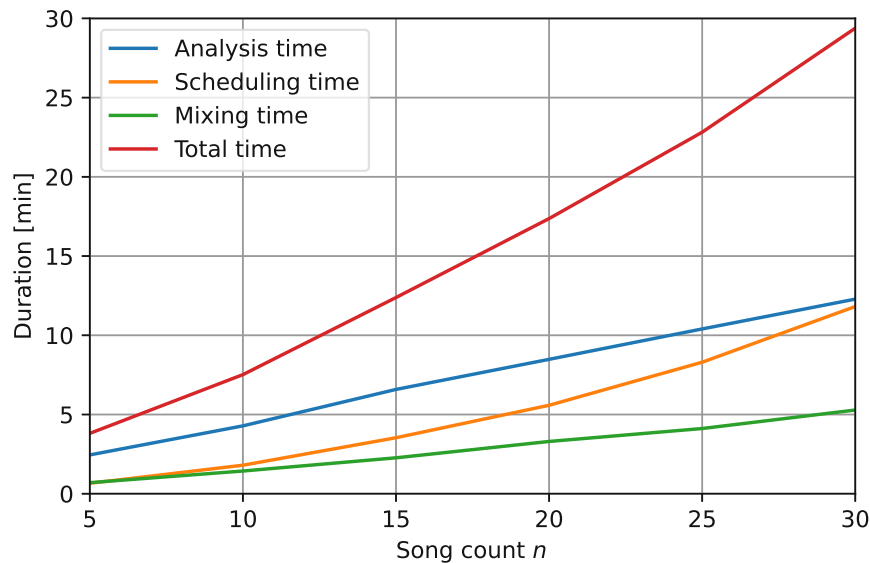


Figure 4.1: Runtimes of the song analysis, scheduling, and mixing procedures for different amounts of songs.

We can see the linear growth of the runtime for the song analysis and the mixing procedure and the quadratic runtime growth of the scheduling procedure. Contrary to our expectations, the song analysis is the most demanding task until a mix of 30 songs. This suggests that, given a reasonable amount of n songs, further performance improvements could be achieved by vertical scaling, specifically by using more CPU cores and GPUs to speed up the song analysis procedure.

4.4 Frontend

We develop a web-based frontend, using *Vue.js* to allow users to interact with our auto-mixing system. The initial mixing page screenshot is depicted in Figure 4.2. On the left panel, the user can upload songs that should be included in the mix. On the right panel, the user can configure the model to be used for the mix generation. The user

can also select the baseline [AMU](#) model for evaluation purposes. Our model *Mosaikbox* can be configured to use contextual information and stem modification. For the stem modification configuration, the user can choose between combinations of vocal removal in case of vocal clashing and drum removal in case of percussive incompatibilities.

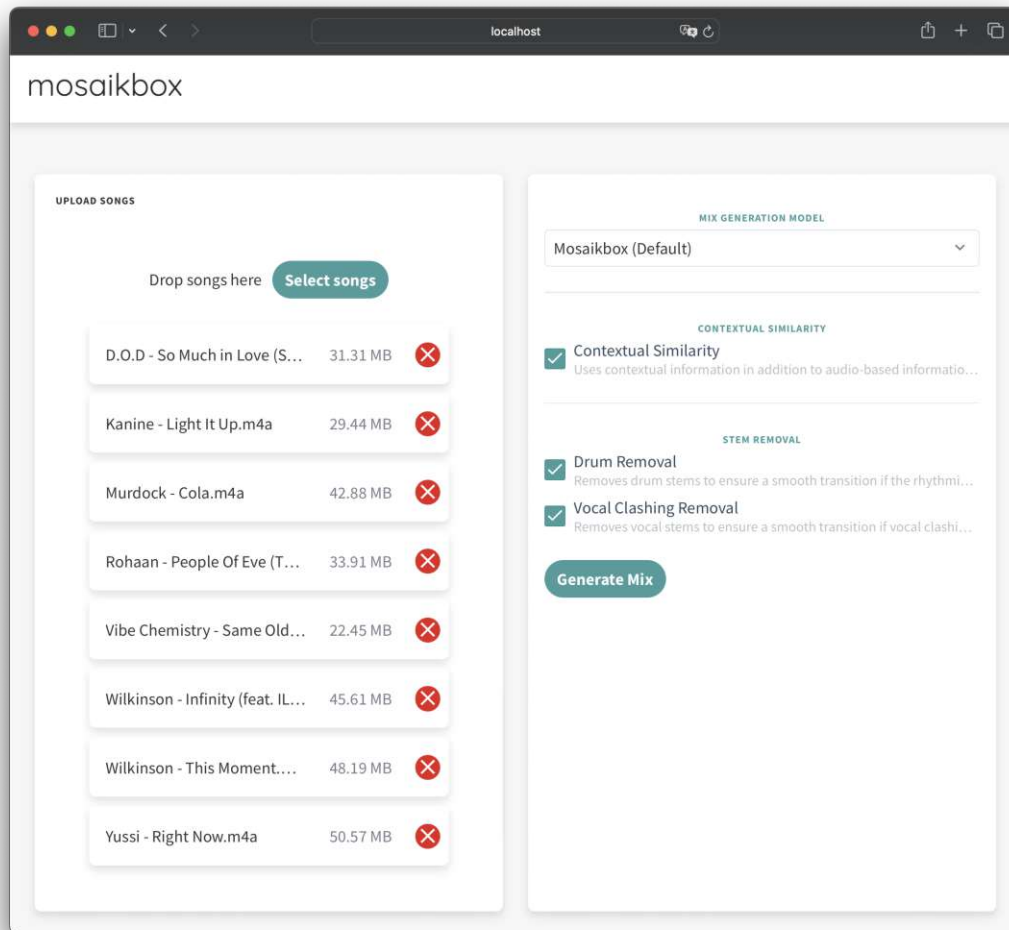


Figure 4.2: Screenshot of frontend after uploading songs.

Once clicking the "Generate Mix" button, the user is asked to select the initial song the mix should start with, as depicted in [Figure 4.3](#). Afterward, the mix generation process is started, and a progress bar that indicates the current stage of the mix generation is displayed.

[Figure 4.4](#) shows the user interface upon completion of the mix generation. The user is presented with a song schedule that visualizes the mix using the analogy of two turntables (decks), labeled "Deck 1" and "Deck 2". This view simulates a real-world DJ setup

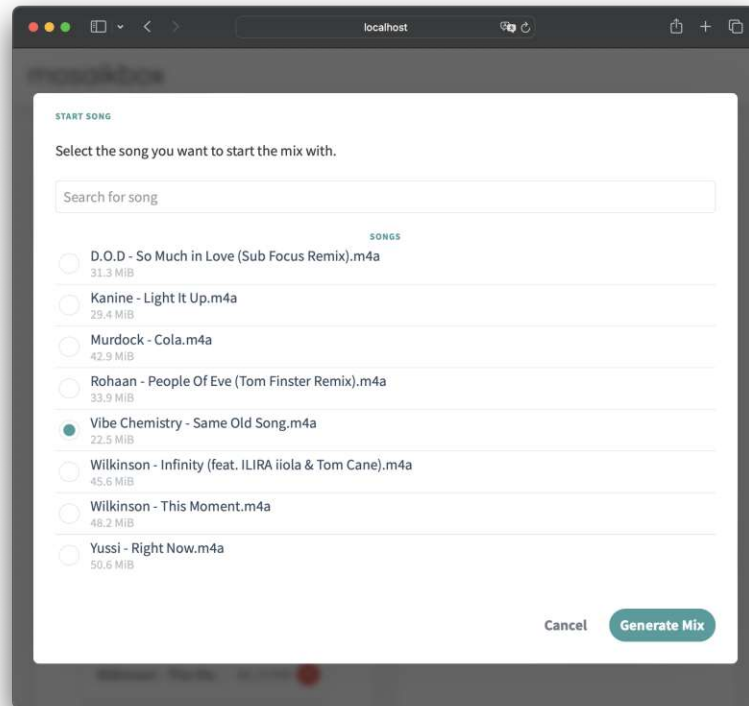


Figure 4.3: Screenshot of the frontend during the song selection step.

with two decks alternating in playing song excerpts. The horizontal overlap between song excerpts depicts the transition overlap, i.e., the time when both songs are playing simultaneously. Hovering over a song in the schedule reveals the song title together with the start and end time of the song excerpt.

Moreover, the user is presented with a visualization of the mix as a waveform, an integrated audio player for listening, and the option to download the mix. Clicking on a song within the schedule navigates the audio player to the corresponding transition point in the mix. Further, the user can access details about the selected song excerpt in the lower left panel. The details include mixability scores, key and tempo information. For instance, the screenshot demonstrates that the selected song "Yussi - Right Now" was pitch-shifted by +2 semitones from the original key $A\flat$ -Minor to $B\flat$ -Minor, or in the *Camelot* notation, from $1A$ to $3A$. Additionally, it highlights that the tempo remained unchanged.

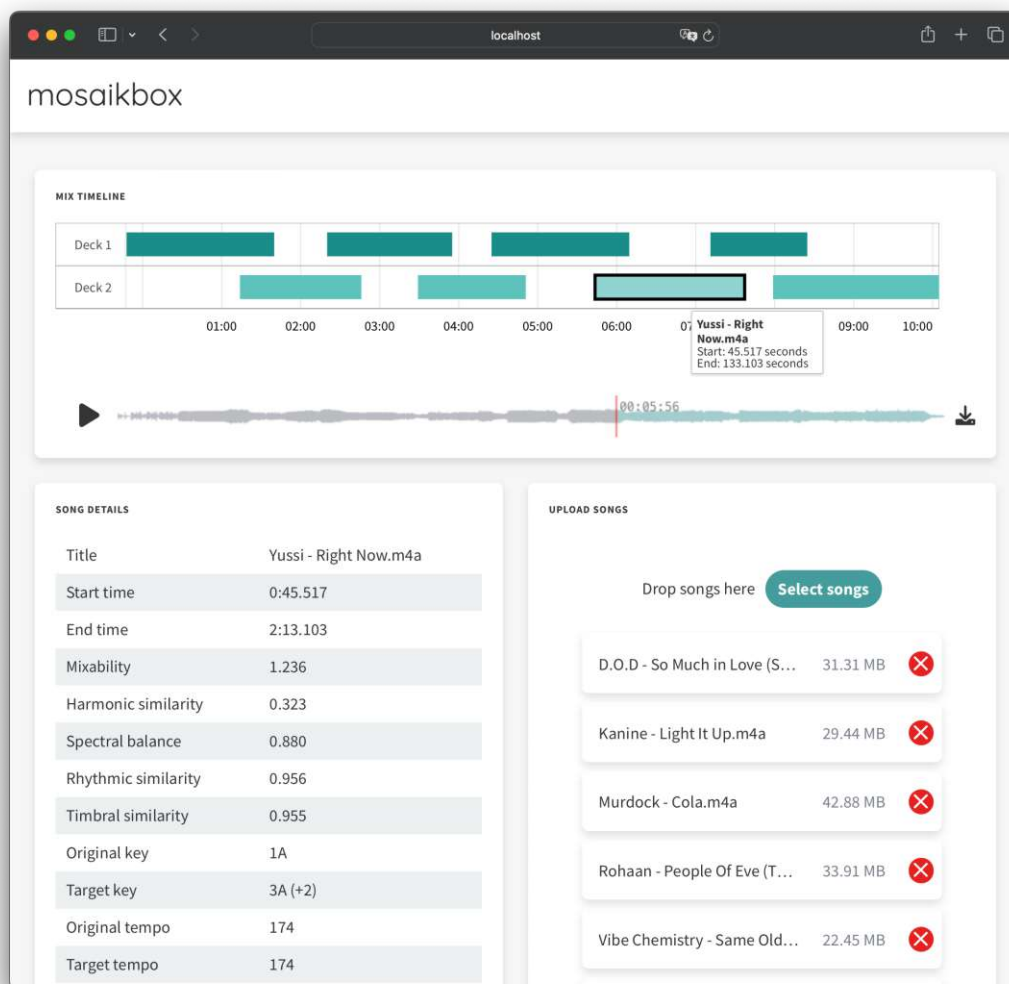


Figure 4.4: Screenshot of the frontend after the mix generation finished.

Listening Experiment

Due to the subjective nature of mixes [Hua+21] and the lack of ground truth corpora for similarity ratings between songs [RBH13], we will evaluate the performance of our proposed solutions with subjective methods. In particular, we will conduct a listening experiment to evaluate the performance of our system.

This chapter outlines the setup of our listening experiment. In section 5.1 we describe the survey structure and present the user interface for the listening experiment. We then detail our data acquisition and the sampled dataset used for the mix generation in section 5.2. Finally, we present the generated song schedule of all models in section 5.3.

5.1 Setup

To assess how musical knowledge influences the evaluation, we first ask the participants about their musical background and whether they have any prior experience in DJing. We split the following survey for each model into two parts. In the first part, we gather song-pair compatibility (SPC) ratings by asking the participants to rate how well individual songs that the models have mixed, fit together. This allows us later on to compare the song selection of the models to the collected SPC ratings. For all pairs of songs, the participants are asked to assess the compatibility based on the following categories:

- **Timbre:** Are the songs similar regarding timbre?
- **Rhythm:** Do the songs have a similar rhythmic pattern?
- **Harmony:** Do the songs have a similar harmonic structure?
- **Overall Mixable:** Are the songs overall mixable?

In the second part, the participants are presented with the generated mix of a model and are asked to rate the overall quality of each transition of the mix on a scale of 1 (awful) to 5 (excellent). The models are presented in random order to prevent bias from the order of presentation. Further, no information about the model type is given to the participants.

5.1.1 User Interface

We developed a web-based user interface for the listening experiment to streamline the evaluation process. To minimize the cognitive load on the participants and to speed up the evaluation process, we designed the user interface with simplicity in mind.

The user is first presented with a short introductory text, explaining the purpose of the listening experiment and its structure. After the introduction, the user is asked for their musical background and DJing experience. For each of the four models, the user is then presented with binary questions to evaluate the compatibility of the song pairs, as shown in [Figure 5.1](#). The integrated waveform audio player can be used to listen to the song excerpts and quickly jump to specific parts of the song.

After all song pairs of a model are evaluated, the user is presented with the generated mix of the model, as depicted in [Figure 5.2](#). It is possible to jump between transitions by clicking on the respective scheduled song excerpt in the mix timeline. Unlike in the mixing interface, the user cannot view any information about the scheduled song excerpt, such as mixability score or pitch shift. This is to prevent further bias from the participants and to ensure a fair evaluation. In the evaluation box below the mix timeline, the user can rate the quality of the transition on a scale of 1 (awful) to 5 (excellent).

At the end of the survey, the users are asked to provide an optional email address to receive the results of the listening experiment.

5.1.2 Implementation

The user interface was implemented using *Vue.js*, reusing components from the mixing interface, and *SurveyJS* as a survey framework. The Python backend was extended to support the evaluation of the models, and survey results were stored in a *MongoDB* NoSQL database. Due to the length of the survey, we are continuously saving the participants' progress and offer the possibility to continue the survey later in case any issues from the participants' side arise. The application was deployed on a [VPS](#) and made accessible to the participants via a public URL. To reduce the load on the server and improve response time, we used a [content delivery network \(CDN\)](#) to cache the audio files and serve them to the participants from edge locations.

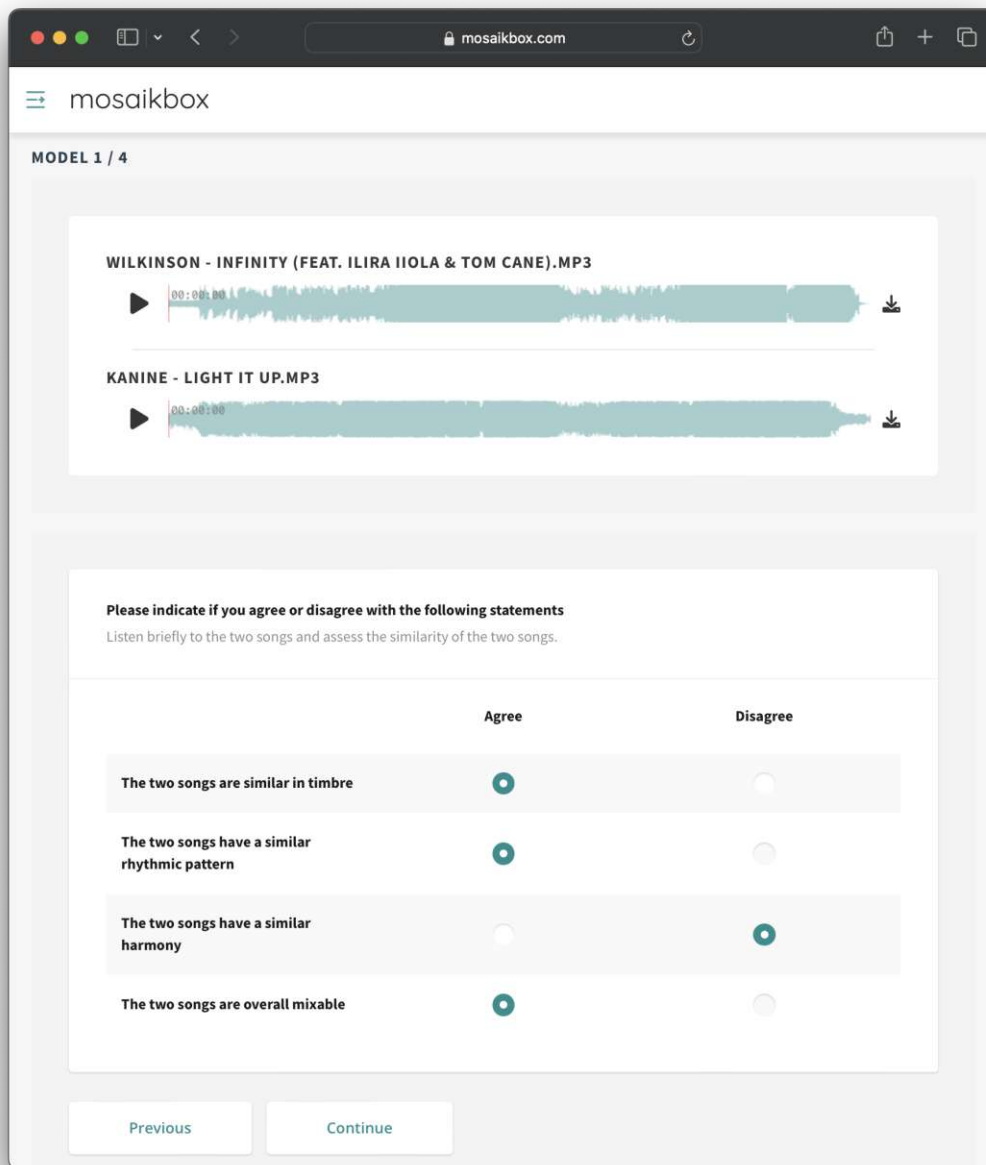


Figure 5.1: Screenshot of the user interface for the evaluation of song pairs.

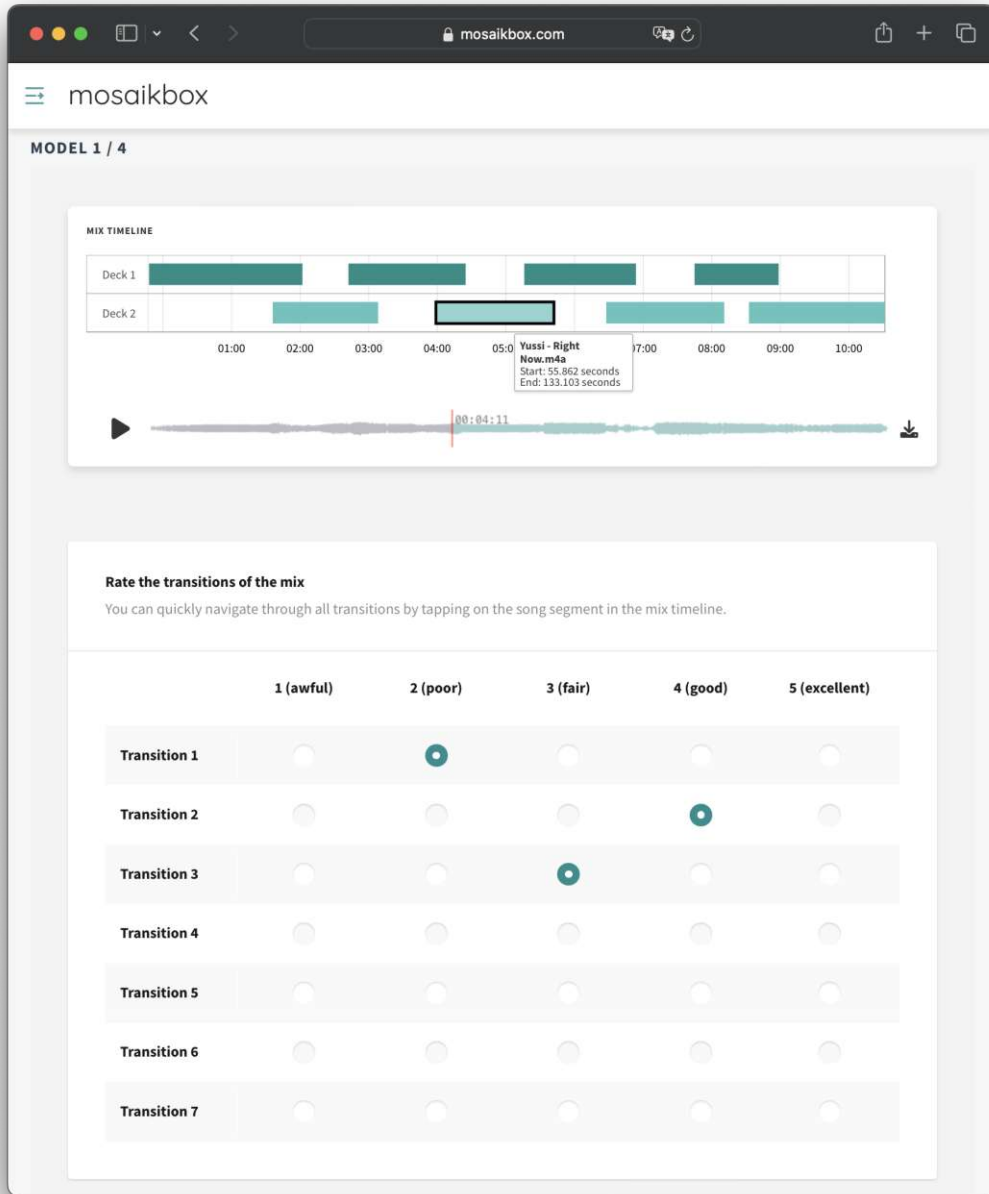


Figure 5.2: Screenshot of the user interface for the evaluation of the generated mix.

5.2 Dataset

Due to the tempo "lock-in", only songs with a tempo tolerance of maximum $\pm 8\%$ are considered. This commonly results in a genre "lock-in" as well, as songs of the same genre usually have a similar tempo. We, therefore, focus on one genre instead of including a wide variety of genres, which the tempo tolerance would rule out. Based on the musical tastes of potential participants, we decided to focus on the drum and bass genre (DnB). This gives the evaluation more weight and relevance, as the participants are likely more familiar with music and mixes of this genre.

We collected a dataset of 250 songs from the most popular DnB playlists on Tidal (e.g., "Super Sharp") and Spotify (e.g. "Drum and Bass Top 100"). As previously benchmarked in [section 4.3](#), using all 250 songs for the mix generation is infeasible due to the exponential growth of the search space. We, therefore, randomly sampled 16 songs from this collection and used them as input for the mix generation of the models. The 16 songs are depicted in [Table 5.1](#).

Title	Artist	Length
Chant	1991	03:45
Champion (Andromedik Remix)	Andromedik	03:27
The Edge of Time	Break	05:21
Dissolve me (feat. klei)	Camo & Krooked	03:30
Flashback	Cartoon	02:26
Breathing (fabric VIP Mix)	Chase & Status	04:01
Back & Forth	Circumference	03:04
So Much in Love (Sub Focus Remix)	D.O.D	03:56
Bittersweet Goodbye (Lense Remix)	Issey Cross	03:21
Light it Up	Kanine	03:41
Cola	Murdock	03:22
People Of Eve (Tom Finster Remix)	Rohaan	02:51
Same Old Song	Vibe Chemistry	03:19
Infinity (feat. ILIRA iiola & Tom Cane)	Wilkinson	03:34
This Moment	Wilkinson	03:38
Right Now	Yussi	03:38

Table 5.1: The dataset of 16 randomly sampled songs used for the mix generation of the models.

5.3 Mix Generation

To highlight the song selection aspect of the models and keep the survey at an acceptable length, we used a top-k approach for the song selection instead of forcefully mixing all 16

songs and thus taking bad scheduling into account. In particular, we use the first eight songs of the song schedule for the mix generation of the models.

As the starting song for all mixes, we select the song "Champion (Andromedik Remix) - Andromedik" by randomly sampling from the dataset. Maintaining the same starting song for all models ensures a fair comparison of the song selection of the models.

All songs are provided in lossless quality (ALAC format) and 44.1kHz sample rate as input to the models. The resulting generated mix of each model is then converted to a 320kbps MP3 file for evaluation to achieve a reasonable file size and to ensure consistent audio quality across all models.

The generated schedule of the AMU baseline model is depicted in Table 5.2. The schedule of the MB_{base} and MB_{stem} model is depicted in Table 5.3. The schedule of the MB_{full} model is depicted in Table 5.4. The start and end times of the song excerpts indicate the transition point of the individual songs, i.e., the time the switch from one song to another occurs. To derive a transition's actual start and end time, eight downbeats would need to be subtracted/added to the start/end time of the song excerpt (cf. section 3.5).

Song	Mix-ability	Start time	End time	Original tempo	Target tempo	Key shift
Andromedik - Champion (Andromedik Remix)	-	0.000	78.621	174	174	0
Issey Cross - Bittersweet Goodbye (Lens Remix)	0.770	113.103	180.690	174	174	-4
Wilkinson - Infinity (feat. ILIRA iola & Tom Cane)	0.745	33.103	97.931	174	174	-1
Camo & Krooked - Dissolve Me (feat. Klei)	0.760	27.586	91.034	175	174	-5
Cartoon - Flashback	0.713	37.241	116.552	174	174	4
Circumference - Back & Forth	0.726	107.586	151.724	172	174	-1
Rohaan - People Of Eve (Tom Finster Remix)	0.727	89.709	153.157	172.1	174	6
Wilkinson - This Moment	0.690	13.103	217.931	174	174	2

Table 5.2: The song schedule of the **AMU** baseline model.

Song	Mix-ability	Start time	End time	Original Tempo	Target Tempo	Original Key	Target Key	Key shift
Andromedik - Champion (Andromedik Remix)	-	0.000	108.966	174	174	1A	1A	0
Wilkinson - Infinity (feat. ILIRA iola & Tom Cane)	2.763	31.724	97.931	174	174	8A	1A	-1
Kanine - Light It Up	2.756	75.862	152.414	174	174	6A	1A	1
Yussi - Right Now	2.758	55.862	133.103	174	174	1A	1A	0
Chase & Status - Breathing (fabric VIP Mix)	2.731	143.750	215.474	175	174	6A	1A	1
Break - The Edge of Time (Workforce Remix)	2.654	69.302	146.543	172	174	6A	1A	1
Murdock - Cola	2.631	131.894	179.480	174	174	7A	12A	-1
Camo & Krooked - Dissolve Me (feat. klei)	2.620	116.552	209.516	175	174	7B	12B	-1

Table 5.3: The song schedule of the **MB_{base}** and **MB_{stem}** models.

Song	Mix-ability	Start time	End time	Original tempo	Target tempo	Original key	Target key	Key shift
Andromedik - Champion (Andromedik Remix)	-	0.000	108.966	174	174	1A	1A	0
Wilkinson - Infinity (feat. ILIRA iola & Tom Cane)	2.763	31.724	97.931	174	174	8A	1A	-1
Murdock - Cola	2.784	90.514	179.480	174	174	7A	12A	-1
Vibe Chemistry - Same Old Song	2.854	97.241	165.517	174	174	4A	11A	1
D.O.D - So Much in Love (Sub Focus Remix)	2.958	21.379	77.241	175	174	11A	11A	0
Wilkinson - This Moment	2.831	143.448	198.621	174	174	4A	11A	1
Circumference - Back & Forth	2.875	33.103	153.103	172	174	4A	11A	1
Chase & Status - Breathing (fabric VIP Mix)	2.711	154.784	240.984	175	174	6A	11A	-1

Table 5.4: The song schedule of the **MB_{full}** model.

Results and Discussion

This chapter summarizes the results of the listening experiment. We first conduct an exploratory analysis to identify general trends and patterns between the different models and the listening experiment results. We then test the significance of our results to support our observations. Finally, we conclude our findings from the data and discuss the implications of our results.

For the sake of transparency, it is worth mentioning that the participants of the listening experiment were mainly academics aged 23-30, with a majority having backgrounds in STEM (science, technology, engineering, and math) and economics, and most were familiar with the drum and bass genre. The raw results of the listening experiment are available on GitHub¹.

6.1 Exploratory Analysis

To get a deeper understanding of the results of the listening experiment, we conduct an exploratory analysis. This analysis will serve as a basis for establishing hypotheses regarding the performance of our models and the relationships between them.

6.1.1 Musical Knowledge and Experience

In total, 28 participants took part in the listening experiment, of which 8 had prior experience in DJing. Most participants stated they have an intermediate or advanced understanding of music theory and its concepts. In contrast, 10 participants stated they have little to no understanding of music concepts, with none rating themselves as experts. The participants' musical knowledge distribution is depicted in [Figure 6.1](#).

¹<https://github.com/robaerd/mosaikbox-survey-results>

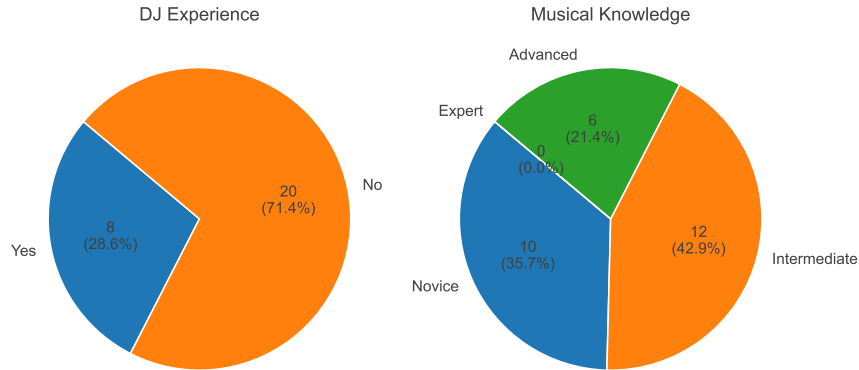


Figure 6.1: Ratio of participants with and without DJing experience and the distribution of the participants’ musical knowledge.

6.1.2 Song-Pair Compatibility (SPC)

The average `SPC` ratings for all models, averaged over all transitions, are depicted in [Table 6.1](#).

Model	Timbre	Rhythm	Harmony	Overall Mixable
<code>AMU</code>	0.459	0.515	0.429	0.628
<code>MB_{base, stem}</code>	0.474	0.643	0.515	0.658
<code>MB_{full}</code>	0.500	0.628	0.551	0.719

Table 6.1: The average `SPC` ratings for all models, averaged over all transitions.

Since the song schedule of our base model `MBbase` and our model with stem modification `MBstem` is the same, we will refer to both models as `MBbase, stem` in the following. All of our models received a better average rating over all aspects compared to the baseline model `AMU`. Additionally, a strong improvement can be observed for all our models’ rhythm and timbral aspects compared to `AMU`. `MBfull` received the highest average `SPC` ratings across all rating aspects except for rhythm, where it was outperformed by `MBbase, stem`.

A visualization and further correlation analysis of the `SPC` ratings over individual results is depicted in [Figure 6.2](#). We can observe that the song pairs for `AMU` have a U-shaped distribution. Furthermore, 5 out of 7 song pairs proved to have a dominant rhythmic aspect, corresponding to the higher weighting of the rhythm in the mixability calculation, compared to our models. The poor harmonic compatibility is surprising, given that `AMU`’s main mixability component is based on harmonic similarity. Harmony and rhythm correlate very highly with the overall mixability, which matches the high weighting of both aspects in the mixability component of `AMU`. What is also interesting to observe is

the high correlation between timbre and overall mixability, even though timbre is not part of the mixability component of **AMU**. Additionally, timbre is highly correlated with harmony, which might explain the high correlation with overall mixability.

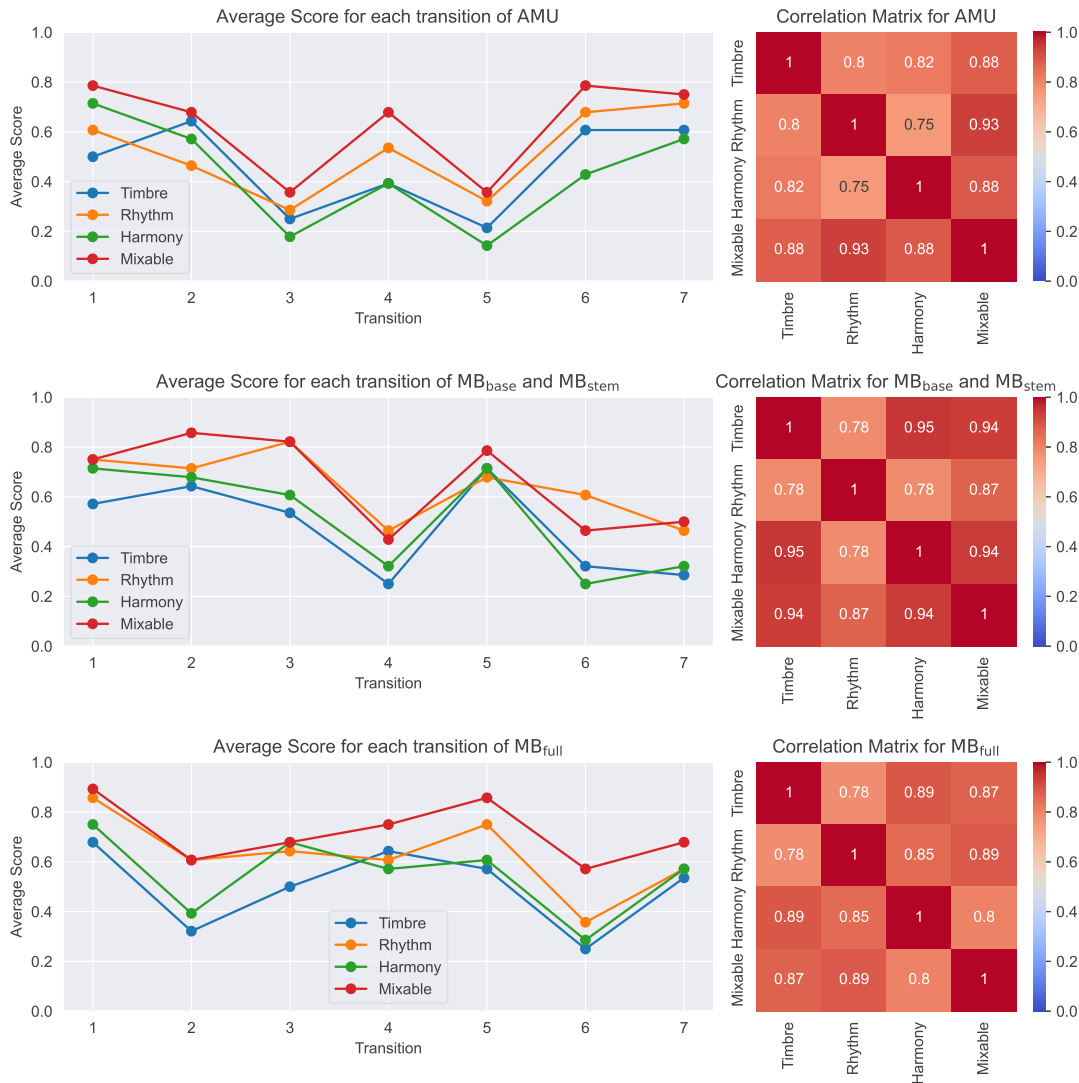


Figure 6.2: The individual **SPC** ratings, averaged over each transition for all models (left) and the correlation between the **SPC** ratings (right).

MB_{base, stem} achieved higher scores on individual **SPC** ratings compared to **AMU** and **MB_{full}**, but was followed by **SPC** ratings with significantly lower scores. Both timbre and harmony are very highly correlated with overall mixability, which matches the high weighting of the timbre and harmony aspects in the mixability component of MB_{base, stem}. We also observe a high correlation between the rhythm and the overall

mixability, matching the weighting of the rhythm aspect in the mixability component of $MB_{\text{base, stem}}$.

The trend of MB_{full} is less variable than that of the other models. Notably, the overall mixability remains high even if timbre, harmony, and rhythm are rated lower, indicating the contribution of the additional contextual information to the mixability of the songs. This is also reflected in the lower correlation between timbre and harmony with the overall mixability compared to the other models.

In all models, we can observe a high correlation between timbre and harmony, while timbre and rhythm were only moderately correlated.

6.1.3 Transition Ratings

The average transition ratings calculated over all models' transitions are depicted in Table 6.2. AMU received the lowest average transition rating, while MB_{stem} received the highest. To our surprise, the extension MB_{full} received a similar rating to MB_{base} .

Model	Average Transition Rating
AMU	2.571
MB_{base}	3.097
MB_{stem}	3.469
MB_{full}	3.036

Table 6.2: The average transition ratings for all models.

Figure 6.3 additionally depicts the frequency of the ratings for each transition by the bubble size. All transitions' average SPC ratings are plotted as a dotted line. To make the average SPC ratings comparable with the transition ratings, we mapped the SPC scale 0-4 to our 1-5 rating scale.

We can observe that the transition ratings for AMU are roughly aligned with the average SPC ratings, except for the first and sixth transitions, where the transition underperforms the expected SPC rating. It is also worth noticing that very few transitions received a rating of 5, while a significant number received a rating of 1.

The transition ratings for MB_{base} also follow the trend of the average SPC ratings, significantly outperforming them during the first four transitions. The fourth and sixth transitions follow the trend of the average SPC ratings and experience a substantial drop in the transition ratings, receiving an average rating of 2.25 and 1.71, respectively. Notable is the increase in 4 and 5 ratings across the first transitions.

Since MB_{stem} should improve upon rhythmic and vocal incompatibilities during the transitions, we expected an overall increase in the transition ratings compared to MB_{base} . However, we can observe a slight decrease in the transition ratings during the first two transitions. We can notice an increase in the transition ratings on all the other transitions.



Figure 6.3: The transition ratings for each transition over all models. The size of the bubbles represents the number of ratings for the respective transition. Each transition's average rating and the SPC are depicted as a line.

Most notable is the shift from 1 and 2 ratings to 3 and 4 ratings, such as for the fourth and sixth transition.

The transition ratings for **MB_{full}** do not follow the trend of the average **SPC** ratings and mostly underperform them. However, similarly to the average **SPC** ratings, the transition ratings for **MB_{full}** are less variable than the other models. Notable is the downward trend of the transition ratings. The first transitions received more positive ratings, while the ones from the second half tended to get lower ratings as the mix progressed.

6.1.4 Relationship between Song Pair Compatibility and Transition Ratings

We performed a correlation analysis to investigate the relationship between the **SPC** and the transition ratings. The results of the correlation analysis are depicted in **Figure 6.4**.

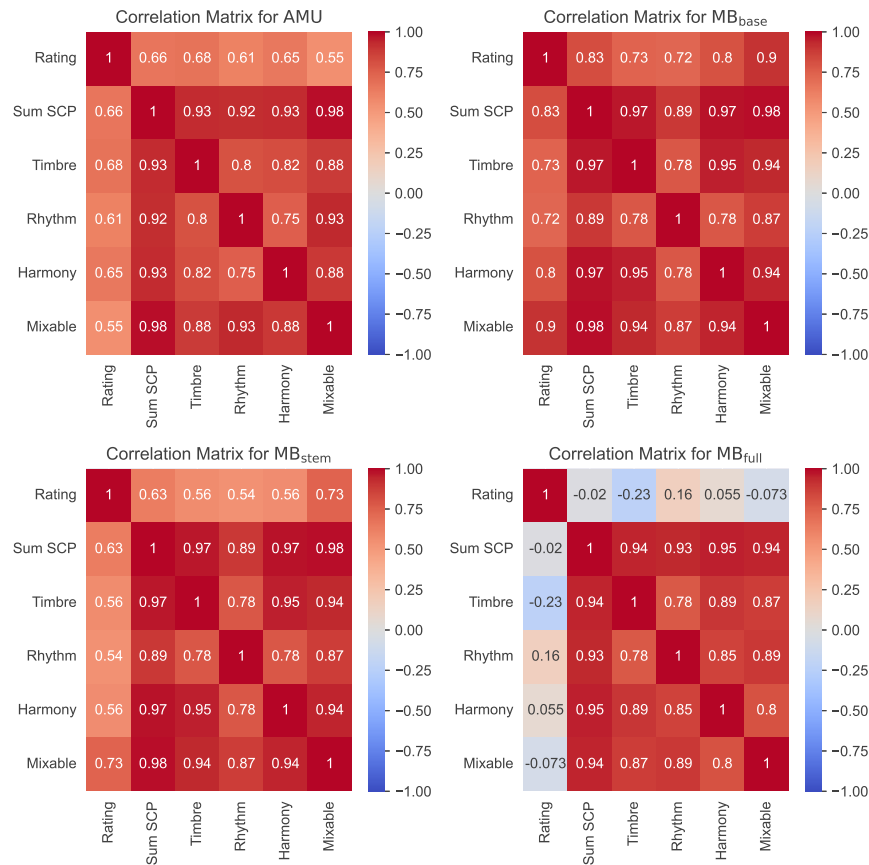


Figure 6.4: Correlation between the transition and the **SPC** ratings for all models.

We can observe similar, moderately strong correlation values across all aspects of the **SPC** and the transition ratings for **AMU**, with Pearson coefficients ranging from 0.55 to 0.68.

Our base model MB_{base} received a very strong correlation between all aspects except timbre and rhythm. This is especially interesting since timbre and rhythm are the main components of our mixability estimation. Also worth noting is the strong correlation between overall mixability and transition ratings, with a Pearson correlation coefficient of 0.90.

The correlation between all aspects of the SPC and the transition ratings for MB_{stem} dropped compared to MB_{base} . This matches our expectations based on the observations in subsection 6.1.3, where the transition for MB_{stem} received significantly better ratings compared to MB_{base} and its SPC ratings.

For MB_{full} , we can observe a weak negative correlation between timbre and transition ratings. We observe a weak or nonexistent correlation with the transition ratings for all other aspects. This is especially interesting since the contextual weighting factor ω_C is set to a lower value compared to other similarity metrics of our mixability estimation. However, the negative correlation between timbre and transition ratings, along with the particularly weak correlation between the transition ratings and other aspects, might also suggest that the contextual weight is too high.

6.1.5 Differences in Experience and Musical Knowledge

We observed a substantial difference in the transition ratings between participants with and without DJing experience, as well as among those with different musical knowledge. Figure 6.5 visualizes the transition ratings for each transition over all models, separated by the participants' DJing experience and musical knowledge. The figure additionally depicts the confidence interval for each transition rating.

Participants with DJ experience rated the transitions of AMU considerably lower than those without DJing experience. The average transition ratings for participants with DJing experience are 2.054 and 2.779 for those without. This trend is consistent with the visualization across nearly all transitions. A similar trend is observable for musical knowledge. Participants with less musical knowledge rated the transitions by the AMU model more favorably than those with more musical knowledge. A significant decline in ratings is observable in participants with advanced musical knowledge.

The MB_{base} model was rated similarly by participants with and without DJing experience for all transitions with the exception of the fourth, sixth, and seventh. For these transitions, participants with DJing experience gave around a whole step lower ratings than those without DJing experience. Remarkable is the smaller confidence interval for negative ratings compared to positive ratings by participants with DJ experience and the consensus on a 1 rating for the sixth transition. Participants with advanced musical knowledge rated the transitions similarly to participants with DJ experience, except for the sixth transition, where they awarded a slightly higher rating.

Participants with DJing experience rated all transitions significantly better than the MB_{base} model. While participants without DJing experience rated most of the transitions

6. RESULTS AND DISCUSSION

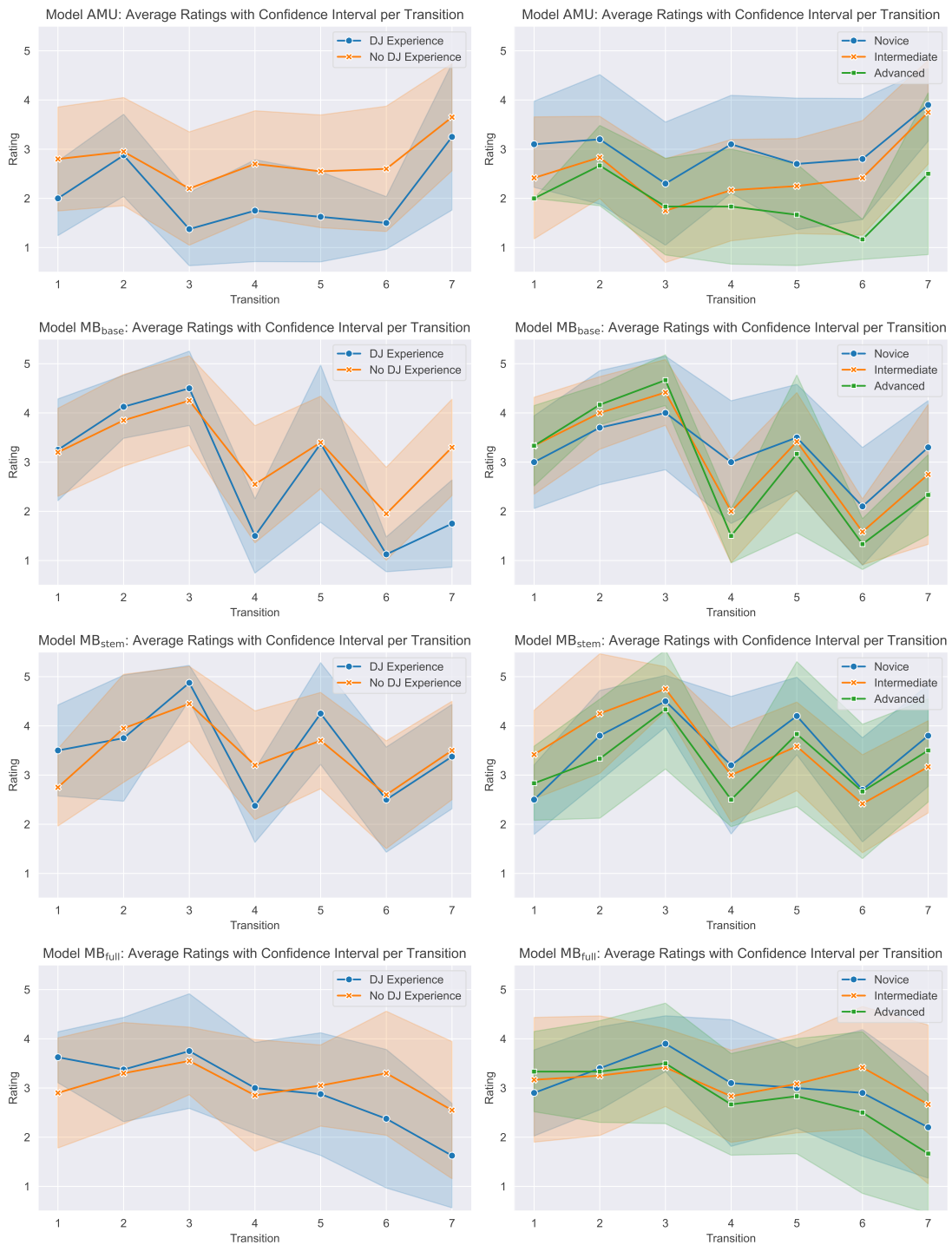


Figure 6.5: Averaged transition ratings over all models for participants with and without DJing experience (left) and participants with different musical knowledge (right).

higher compared to the MB_{base} model, the increase in ratings is less significant compared to participants with DJing experience. What is interesting to observe is the significantly higher confidence interval for participants with advanced musical knowledge. On average, participants with advanced musical knowledge rated the transitions lower than participants with DJ experience.

The ratings decline more steeply for participants with DJing experience compared to those without throughout the mix for the MB_{full} model. This is also reflected by the participants with intermediate and advanced musical knowledge, whose ratings decline more steeply than those with basic musical knowledge.

6.2 Statistical Significance

Before we can draw any conclusions from the results, we need to test if there is a significant difference between the performance of the models.

6.2.1 Overall Performance

To form an overall conclusion on model performance, we aggregated transition ratings for each model. A Shapiro-Wilk test for the normality of the transition ratings for each model reveals that the transition ratings are not normally distributed. Corresponding p-values of all models for the Shapiro-Wilk test are $p < 0.00001$.

Next, we perform a Kruskal-Wallis test to look for significant differences between the transition ratings of the models. The Kruskal-Wallis test reveals a significant difference between the four models with $p < 0.00001$.

We perform a Mann-Whitney U test for the following model pairs to further investigate the differences between the models. We start by comparing all our models against the baseline AMU model to verify the hypothesis that our models outperform the baseline model. To evaluate the effect of stem modification, we then compare MB_{stem} against our base model MB_{base} . Finally, to conclude the effect of the additional contextual information, we compare MB_{full} against the base model MB_{base} and our model with stem modification MB_{stem} . In total, we will conduct 6 Mann-Whitney U tests.

When performing multiple hypothesis tests, the likelihood of committing a Type I error, where a null hypothesis is incorrectly rejected, increases significantly with the number of tests performed [Sha95]. This increase in the error probability can lead to wrong conclusions about the significance of the results. To counteract this, we will apply the Holm-Bonferroni correction [Hol79] to adjust the p-values of the Mann-Whitney U tests. The Holm-Bonferroni correction can be seen as a sequential Bonferroni correction, where the p-values are ordered in ascending order and then adjusted stepwise. Compared to the Bonferroni correction, the Holm-Bonferroni correction achieves the same Type I error rate while reducing the Type II error rate, where a valid null hypothesis is incorrectly rejected [HB90].

We compute the adjusted p-values as follows,

$$\hat{p}_i = \max_{j \leq i} (\min(p_j \cdot (m - j + 1), 1)), \quad (6.1)$$

where m is the number of tests performed and p_j is the p-value of the j -th test sorted in ascending order.

Let $F(u)$ and $G(u)$ be the underlying cumulative distribution functions of the transition ratings of the models *Model 1* and *Model 2*, respectively. We test for the alternative hypothesis that the distribution of *Model 1* is stochastically greater than the distribution of *Model 2*, i.e., $F(u) < G(u)$, for all u . We reject the null hypothesis if the adjusted p-value \hat{p} is less than the significance level $\alpha = 0.05$. The results of the Mann-Whitney U test and the adjusted p-values are depicted in [Table 6.3](#).

Model 1 (F)	Model 2 (G)	\hat{p} -value $_{F(u) < G(u)}$
MB _{base}	AMU	0.00014
MB _{stem}	AMU	0.00000 (1.2696e-11)
MB _{full}	AMU	0.00019
MB _{stem}	MB _{base}	0.00991
MB _{full}	MB _{base}	1.00000
MB _{full}	MB _{stem}	1.00000

Table 6.3: Corrected p-values of the Mann-Whitney U tests testing if any of the models significantly outperforms another model.

From the results, we can conclude that our models MB_{base}, MB_{stem}, MB_{full} significantly outperform the baseline model AMU. We can also conclude that stem modification significantly improves the transition ratings of our base model MB_{base}. We failed to reject the null hypothesis for the MB_{full} model. We thus cannot conclude that additional contextual information leads to any significant improvement in the transition ratings.

6.2.2 Differences in SPC and Transition Ratings

In [subsection 6.1.3](#), we observed that average SPC ratings for all models have a similar trend to the average transition ratings. Furthermore, we found in [subsection 6.1.4](#) that there is a strong correlation between SPC and transition ratings for the three models MB_{base}, MB_{stem} and AMU and no correlation for the MB_{full} model. To draw any conclusion from these observations, we need to test whether there is a significant difference between the SPC and the transition ratings of the models.

We calculate the overall SPC ratings for each transition by summing the average SPC aspects of the corresponding song pairs and map the scale 0-4 to our 1-5 rating scale. Since we have seven transitions, our sample size is too small to conclude the normality of the SPC or transition ratings. Thus, we choose to perform a Mann-Whitney U test to test for significant differences between the SPC and the transition ratings of the models.

Our null hypothesis is that the **SPC** and the transition ratings of the models are equal, i.e., $F(u) = G(u)$, for all u . To account for Type I errors, we employ the Holm-Bonferroni correction again and adjust the p-values of the Mann-Whitney U tests. We reject the null hypothesis if the adjusted p-value \hat{p} is less than the significance level $\alpha = 0.05$.

The results of the Mann-Whitney U test and the adjusted p-values are shown in **Table 6.4**. We failed to reject the null hypothesis for all models and thus cannot conclude any significant difference between the **SPC** and the transition ratings of the models.

Model	\hat{p} -value
AMU	0.83450
MB _{base}	1.00000
MB _{stem}	1.00000
MB _{full}	0.83450

Table 6.4: Corrected p-values of the Mann-Whitney U tests testing for significant differences between the **SPC** ratings and the transition ratings of the models.

6.2.3 DJ Experience

Based on the results of the exploratory analysis, we hypothesize that there is a significant difference in the transition ratings between participants with and without DJing experience. In particular, we suggest that those with DJing experience rate the transitions significantly lower compared to those without.

We tested again for the normality of the transition ratings using a Shapiro-Wilk test. We found that the transition ratings are not normally distributed with a p-value of $p < 0.0001$ for all models. We then carried out a Mann-Whitney U test to evaluate the alternative hypothesis that $F(u) > G(u)$, for all u , where $F(u)$ and $G(u)$ represent the underlying cumulative distribution functions of the transition ratings from participants with and without DJing experience, respectively. To account for Type I errors, we again employ the Holm-Bonferroni correction and adjust the p-values of the Mann-Whitney U tests. We reject the null hypothesis if the adjusted p-value \hat{p} is less than the significance level $\alpha = 0.05$.

The results of the Mann-Whitney U test and the adjusted p-values are displayed in **Table 6.5**. We reject the null hypothesis only for the **AMU** model. We thus can only conclude that participants with DJing experience rate the transitions by the **AMU** model significantly worse than those without.

6.2.4 Musical Knowledge

To further investigate variations in transition ratings, we hypothesize a significant difference exists among participants with differing levels of musical knowledge. We conduct a Kruskal-Wallis test over all models to test for significant differences between

Model	\hat{p} -value
AMU	0.00017
MB _{base}	0.10840
MB _{stem}	0.67239
MB _{full}	0.56869

Table 6.5: Corrected p-values of the Mann-Whitney U test, testing the alternative hypothesis that participants with DJing experience rated the transitions significantly worse than those without DJing experience.

the transition ratings of participants with different musical knowledge. To address Type I errors resulting from multiple comparisons, we once again utilize the Holm-Bonferroni correction and adjust the p-values of the Kruskal-Wallis test. We reject the null hypothesis if the adjusted p-value \hat{p} is less than the significance level $\alpha = 0.05$.

The results of the Kruskal-Wallis test and the adjusted p-values are depicted in Table 6.6. We reject the null hypothesis for the baseline model AMU. We thus can conclude that there is a significant difference for AMU in the transition ratings between participants with different musical knowledge.

Model	\hat{p} -value
AMU	0.00009
MB _{base}	1.00000
MB _{stem}	1.00000
MB _{full}	1.00000

Table 6.6: Corrected p-values of the Kruskal-Wallis test, testing for significant differences between the transition ratings of participants with varying musical knowledge.

To investigate these differences in more depth, we perform a pairwise Mann-Whitney U test to determine which groups significantly differ by testing for the alternative hypothesis that $F(u) \neq G(u)$, for at least one u . We again employ the Holm-Bonferroni correction and use the significance level $\alpha = 0.05$. The results of the Mann-Whitney U test and the adjusted p-values are displayed in Table 6.7. We can conclude that a significant difference exists for participants across all musical knowledge levels for the AMU model.

To further examine the differences, especially regarding which musical knowledge levels rated the transitions more or less favorably, we compare the mean ranks of the transition ratings. Participants with novice, intermediate, and advanced musical knowledge received mean ranks of 118.79, 96.14, and 69.40, respectively. Therefore, we can conclude that participants with novice musical knowledge rated the transitions of AMU significantly better than those with intermediate and advanced musical knowledge. Additionally, participants with intermediate musical knowledge rated the transitions significantly more favorably than those with advanced musical knowledge.

Model 1 (F)	Model 2 (G)	\hat{p} -value $_{F(u) \neq G(u)}$
AMU _{Novice}	AMU _{Intermediate}	0.01696
AMU _{Novice}	AMU _{Advanced}	0.00000
AMU _{Intermediate}	AMU _{Advanced}	0.01696

Table 6.7: Corrected p-values of the Mann-Whitney U tests testing for significant differences between the transition ratings of participants with different musical knowledge for the `AMU` model.

6.3 Discussion

The outcomes of the listening experiment and the statistical significance tests offer us valuable insights regarding the performance of our models. We will summarize these findings in the following section before addressing our research questions.

Apart from the baseline model `AMU`, we could not derive any difference in ratings across music knowledge levels or DJing experience. For the baseline model alone, we determine that the more experienced participants rated the transitions significantly worse than the less experienced ones. We will therefore not make distinctions based on levels of musical knowledge or DJ experience in the subsequent discussion.

Our `SPC` ratings give us a good indicator of the potential compatibility and mixability of songs. After comparing the `SPC` ratings to the transition ratings, we found no significant difference between the two rating categories. We further successfully identified and matched correlating aspects of `SPC` to the mixability estimates of our models.

Let us now consider the overall performance of our models. All of our models significantly outperformed the baseline model `AMU`. Using our model with stem modification `MBstem` greatly improved the transition ratings compared to our base model `MBbase`. This leads us to conclude our first research question (RQ1) that stem modification significantly improves the overall quality of a mix. Additionally, we discovered that using the rhythmic compatibility from our mixability measure and the vocal intersection duration is sufficient to formulate effective rules for stem removal (cf. `section 3.4`).

We were not able to prove that the additional contextual information of our model `MBfull` significantly improves the transition ratings compared to our models `MBbase` or `MBfull`. Even though the overall mixability and rhythm aspects of the `SPC` ratings were higher than those of the other models, timbre and harmony were rated lower. As already hypothesized in `subsection 6.1.3`, we believe we set the contextual weighting factor ω_C too high. Another reason for the poor performance of the `MBfull` model might be that contextual relevance differs between genres of music. Since we only evaluated our models on one genre, we cannot conclude that the additional contextual information is not useful for other genres of music. We therefore conclude our second research question (RQ2) that we could not prove that the additional contextual information significantly improves the overall quality of a mix.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Conclusion

In this thesis we present a novel automatic music-mixing framework that surpasses existing state-of-the-art automatic music-mixing systems. This approach differs from existing systems by employing state-of-the-art [MIR](#) techniques and incorporating a more comprehensive mixability measure that captures additional audio aspects to better match the actual DJ music selection techniques. As part of this framework, we proposed a novel beat-grid estimation algorithm that serves as a foundation for various sub-tasks in the automatic music-mixing process.

We implemented a rule-based stem modification approach to answer our first research question, whether the mixability of tracks can be increased by selectively removing stems, and if so, what measures can be used to formulate such rules. Using rhythmic similarity and vocal intersection as measures to define stem modification rules, we showed that our stem modification approach significantly improves the quality of the mix and increases the mixability of tracks. To answer our second research question, whether contextual information such as lyrics can be used to improve the quality of the mix, we incorporated the semantic similarity of the lyrics into our mixability calculation. However, this proved to have no significant impact on the quality of the mix.

Given the challenges of objectively assessing music mix quality, we were constrained to opt for a subjective evaluation, which entailed certain limitations. A particular restriction was the limited amount of mixes that could be evaluated due to the time commitment required from the participants. Participants reported spending 45 - 60 minutes on the survey, which made evaluating additional mixes infeasible. To collect enough data to draw meaningful conclusions, we had to limit the mix to one genre, which may have introduced a certain bias. Additionally, the homogeneity of participant's demographic backgrounds (cf. [chapter 6](#)), may have further compounded this bias.

We consider the song segmentation one of the primary limitations of our mixing system, as it sometimes predicted song boundaries to end either too early or too late. Although

7. CONCLUSION

our stem modification approach made the sudden changes in the mix less noticeable, we believe that a more accurate song segmentation algorithm, such as by Wang et al. [WHS22], would improve the overall quality of the mix. Apart from this limitation, our work shows possibilities for improvement and future work in many areas. For instance, incorporating a feature that captures a song excerpt's perceived intensity and activity could improve the scheduling aspect, enabling the mixing system to account for breaks and build-ups. Moreover, since the quality of a mix is highly dependent on personal preference, we believe that our mixability measure could be further improved by incorporating similarity measures obtained by collaborative filtering techniques. Finally, a more sophisticated transition model supporting different transition techniques, such as *double drops*, *switch*, *filter fade*, or popular effects, such as *reverb* and *dub echo* might further improve the overall quality of the mix.

List of Figures

2.1	This figure depicts the three-step beat detection process. (a) shows the original audio signal x . (b) shows the feature vector F_i of a specific feature, in this case, the chroma vectors. (c) illustrates the downbeat detection function d . (d) shows the discrete downbeat position sequence s . [Dur+17]	6
2.2	Illustrations of the three-step DNN-based ADT approach. After extracting features from the audio, the features are fed into three separate DNNs, one for each drum instrument. The activations of the output layer of the DNNs are then peak-picked to obtain the individual drum onsets. [SSH16] . . .	8
2.3	The circle of fifths - also known as <i>Camelot Wheel</i> using a different key-coding [Mix] - is widely used by DJs to find a key-compatible song to mix.	9
2.4	Example of the most typical (even) drum patterns found in EDM [But06, p. 82] [PBH14].	12
2.5	Illustration of the hybrid transformer Demucs architecture. The architecture consists of two parallel stages. The input waveform is processed by a temporal encoder and simultaneously transformed into its spectral representation using the STFT and processed by a spectral encoder. Spectral encoders are prefixed with a Z, whereas temporal encoders are prefixed with a T. These two representations are fed into a Cross-domain Transformer Encoder using interleaved Transformer Encoder layers and cross-attention Encoder layers. Temporal and spectral decoders then process the output of the Cross-domain Transformer Encoder. The output spectrogram is then transformed back into the waveform dimension using the inverse-STFT (ISTFT) and summed with the output of the waveform outputs of the other stage, giving the final model output. [RMD23]	16
3.1	Comparison of our tempo estimation algorithm with a state-of-the-art tempo estimation algorithm by Böck and Davis [BD20] on the GiantSteps dataset for errors below 4%.	23
3.2	Song segments where the boundaries were detected by Serrà et al.'s algorithm [Ser+14] and the segments were labelled by Nieto and Bello's algorithm [NB14]. Segments are separated by a vertical line. The color of the segments represents the label of the segment. The song is <i>Falling</i> by <i>Camo & Krooked</i>	24

3.3	A three seconds excerpt from 03:66-06:66 of the song <i>Falling</i> by Camo & Krooked pitch shifted by 1 and 2 semitones using the pitch shifting algorithm <i>RubberBand</i> . The top plot shows the original audio, transformed into mono. The middle plot depicts the waveform of the original audio after pitch shifting by 1 semitone. The bottom plot shows the waveform of the original audio after pitch shifting by 2 semitones.	27
3.4	Mixability and its features per beat shift for a candidate song.	30
3.5	Frequency response of a low-pass Butterworth filter for the first 7 orders and a cutoff frequency of 2800 Hz.	33
3.6	Frequency response of the low-, mid- and high-shelving filters.	34
3.7	Equalization applied to both audio excerpts, in case of low rhythmic similarity, low timbral similarity and no drum stem modification.	35
3.8	Equalization applied to both audio excerpts, in case of high rhythmic similarity, high timbral similarity and drum stem modification.	36
3.9	Equalization applied to the vocal and drum stems to prevent vocal clashing and enable mixing of rhythmically incompatible songs.	37
4.1	Runtimes of the song analysis, scheduling, and mixing procedures for different amounts of songs.	42
4.2	Screenshot of frontend after uploading songs.	43
4.3	Screenshot of the frontend during the song selection step.	44
4.4	Screenshot of the frontend after the mix generation finished.	45
5.1	Screenshot of the user interface for the evaluation of song pairs.	49
5.2	Screenshot of the user interface for the evaluation of the generated mix.	50
6.1	Ratio of participants with and without DJing experience and the distribution of the participants' musical knowledge.	56
6.2	The individual SPC ratings, averaged over each transition for all models (left) and the correlation between the SPC ratings (right).	57
6.3	The transition ratings for each transition over all models. The size of the bubbles represents the number of ratings for the respective transition. Each transition's average rating and the SPC are depicted as a line.	59
6.4	Correlation between the transition and the SPC ratings for all models.	60
6.5	Averaged transition ratings over all models for participants with and without DJing experience (left) and participants with different musical knowledge (right).	62

List of Tables

3.1	Comparison of our tempo estimation algorithm with a state-of-the-art tempo estimation algorithm by Böck and Davis [BD20] on unseen data from the GiantSteps dataset.	22
3.2	Overview of different filter types' roll-off and phase response characteristics.	32
5.1	The dataset of 16 randomly sampled songs used for the mix generation of the models.	51
5.2	The song schedule of the AMU baseline model.	53
5.3	The song schedule of the MB _{base} and MB _{stem} models.	53
5.4	The song schedule of the MB _{full} model.	53
6.1	The average SPC ratings for all models, averaged over all transitions. . .	56
6.2	The average transition ratings for all models.	58
6.3	Corrected p-values of the Mann-Whitney U tests testing if any of the models significantly outperforms another model.	64
6.4	Corrected p-values of the Mann-Whitney U tests testing for significant differences between the SPC ratings and the transition ratings of the models. .	65
6.5	Corrected p-values of the Mann-Whitney U test, testing the alternative hypothesis that participants with DJing experience rated the transitions significantly worse than those without DJing experience.	66
6.6	Corrected p-values of the Kruskal-Wallis test, testing for significant differences between the transition ratings of participants with varying musical knowledge.	66
6.7	Corrected p-values of the Mann-Whitney U tests testing for significant differences between the transition ratings of participants with different musical knowledge for the AMU model.	67

List of Algorithms

3.1 Beat Grid Estimation Using 2D Constrained Minimization	21
3.2 Time linear Butterworth shelving filter algorithm	34

Glossary

MB_{base} Our base model, without stem modification or contextual information. The *MB* stands for the name we gave the application: *Mosaikbox*. [37](#), [52](#), [53](#), [56](#), [58](#), [61](#), [63-67](#), [73](#)

MB_{full} The extension of our stem modification model [MB_{stem}](#) with contextual similarity. [37](#), [42](#), [52](#), [53](#), [56-58](#), [60](#), [61](#), [63-67](#), [73](#)

MB_{stem} The extension of our base model [MB_{base}](#) with stem modification. [37](#), [52](#), [53](#), [56](#), [58](#), [61](#), [63-67](#), [73](#)

BM25 Best Matching 25 - A ranking function to rank matching documents according to their relevance to a given search query. [29](#)

TF-IDF Term Frequency-Inverse Document Frequency - Measures the importance of a term to a document in a collection. [14](#), [29](#)

Acronyms

- ADT** Automatic drum transcription. [7](#), [8](#), [39](#), [71](#)
- ALAC** Apple Lossless Audio Codec. [52](#)
- AMU** AutoMashUpper. [17](#), [18](#), [25](#), [37](#), [38](#), [43](#), [52](#), [53](#), [56-58](#), [60](#), [61](#), [63-67](#), [69](#), [73](#)
- API** application programming interface. [39](#), [40](#)
- CDN** content delivery network. [48](#)
- CNN** convolutional neural network. [6-8](#), [10](#), [11](#), [15](#), [31](#)
- CRNN** convolutional recurrent neural network. [6](#), [7](#)
- DNN** deep neural network. [7](#), [8](#), [39](#), [71](#)
- EDM** electronic dance music. [1](#), [12](#), [13](#), [19](#), [23-25](#), [34](#), [35](#), [42](#), [71](#)
- LSTM** long short-term memory. [15](#)
- LUFS** Loudness Units relative to Full Scale. [31](#)
- MFCCs** Mel-frequency cepstral coefficients. [10](#), [13](#), [25](#), [26](#), [39](#)
- MIR** music information retrieval. [5](#), [10](#), [69](#)
- MSS** music source separation. [2](#), [5](#), [14](#), [15](#), [31](#)
- PCM** pulse-code modulation. [41](#)
- REST** representational state transfer. [40](#)
- RNN** recurrent neural network. [6](#), [7](#), [15](#), [31](#)
- RPC** remote procedure call. [41](#)

SPC song-pair compatibility. [47](#), [56-61](#), [64](#), [65](#), [67](#), [72](#), [73](#)

STFT short-time Fourier transform. [16](#), [39](#), [71](#)

VPS virtual private server. [48](#)

Bibliography

- [APS05] Jean-Julien Aucouturier, Francois Pachet, and Mark Sandler. "The way it Sounds": timbre models for analysis and retrieval of music signals". In: *IEEE Transactions on Multimedia* 7.6 (Nov. 2005), pp. 1028–1035. DOI: [10.1109/TMM.2005.858380](https://doi.org/10.1109/TMM.2005.858380).
- [BB07] Frank Broughton and Bill Brewster. *How to DJ Right: The Art and Science of Playing Records*. Grove/Atlantic, Inc., Dec. 2007. ISBN: 9780802199744.
- [BD20] Sebastian Böck and Matthew Davies. "Deconstruct, analyse, reconstruct: How to improve tempo, beat, and downbeat estimation". In: *Proceedings of the 21st International Society for Music Information Retrieval Conference* (Montreal, Canada). ISMIR, Nov. 2020, pp. 574–582. DOI: [10.5281/zenodo.4245498](https://doi.org/10.5281/zenodo.4245498).
- [BHM20] Jan vom Brocke, Alan Hevner, and Alexander Maedche. "Introduction to Design Science Research". In: *Design Science Research. Cases*. Springer International Publishing, Sept. 2020, pp. 1–13. DOI: [10.1007/978-3-030-46781-4_1](https://doi.org/10.1007/978-3-030-46781-4_1).
- [BKW16] Sebastian Böck, Florian Krebs, and Gerhard Widmer. "Joint Beat and Downbeat Tracking with Recurrent Neural Networks". In: *Proceedings of the 17th International Society for Music Information Retrieval Conference*. ISMIR, 2016. DOI: [10.5281/zenodo.1415836](https://doi.org/10.5281/zenodo.1415836).
- [BM08] John Ashley Burgoyne and Stephen McAdams. "A Meta-analysis of Timbre Perception Using Nonlinear Extensions to CLASCAL". In: *Computer Music Modeling and Retrieval. Sense of Sounds*. Ed. by Richard Kronland-Martinet, Sølvi Ystad, and Kristoffer Jensen. Vol. 4969. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 181–202. DOI: [10.1007/978-3-540-85035-9_12](https://doi.org/10.1007/978-3-540-85035-9_12).
- [BNJ03] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation". In: *The Journal of Machine Learning Research* 3. Jan (2003), pp. 993–1022. DOI: [10.5555/944919.944937](https://doi.org/10.5555/944919.944937).
- [BPN96] Richard H. Byrd, Lu Peihuang, and Jorge Nocedal. *A limited-memory algorithm for bound-constrained optimization*. Report. Mar. 1996. DOI: [10.2172/204262](https://doi.org/10.2172/204262).

- [BS11] Sebastian Böck and Markus Schedl. “Enhanced Beat Tracking with Context-Aware Neural Networks”. In: *Proceedings of the 14th International Conference on Digital Audio Effects (DAFx-11)*. DAFx, 2011.
- [But06] Mark Jonathan Butler. *Unlocking the Groove: Rhythm, Meter, and Musical Design in Electronic Dance Music*. Indiana University Press, 2006. ISBN: 978-0-253-34662-9.
- [Cac+05] Anne Caclin et al. “Acoustic correlates of timbre space dimensions: A confirmatory study using synthetic tones”. In: *Journal of the Acoustical Society of America* 118.1 (2005). DOI: [10.1121/1.1929229](https://doi.org/10.1121/1.1929229).
- [CFG21] Tian Cheng, Satoru Fukayama, and Masataka Goto. “Joint Beat and Downbeat Tracking Based on CRNN Models and a Comparison of Using Different Context Ranges in Convolutional Layers”. In: *Proceedings of the ICMC July 25th-31st, 2021*. 2021.
- [Dav+14] Matthew Davies et al. “AutoMashUpper: Automatic Creation of Multi-Song Music Mashups”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22.12 (Dec. 2014), pp. 1726–1737. DOI: [10.1109/TASLP.2014.2347135](https://doi.org/10.1109/TASLP.2014.2347135).
- [DB19] Matthew E. P. Davies and Sebastian Böck. “Temporal convolutional networks for musical audio beat tracking”. In: *2019 27th European Signal Processing Conference. EUSIPCO*, Sept. 2019, pp. 1–5. DOI: [10.23919/EUSIPCO.2019.8902578](https://doi.org/10.23919/EUSIPCO.2019.8902578).
- [Déf+21] Alexandre Défossez et al. *Music Source Separation in the Waveform Domain*. arXiv preprint arXiv:1911.13254. Apr. 2021. DOI: [10.48550/arXiv.1911.13254](https://doi.org/10.48550/arXiv.1911.13254).
- [Déf21] Alexandre Défossez. “Hybrid Spectrogram and Waveform Source Separation”. In: *Proceedings of the ISMIR 2021 Workshop on Music Source Separation*. 2021.
- [Dro07a] Wim van Drongelen. “11 - Filters: Analysis”. In: *Signal Processing for Neuroscientists*. Burlington: Academic Press, Jan. 2007, pp. 177–188. ISBN: 978-0-12-370867-0. DOI: [10.1016/B978-012370867-0/50011-5](https://doi.org/10.1016/B978-012370867-0/50011-5).
- [Dro07b] Wim van Drongelen. “13 - Filters: Digital Filters”. In: *Signal Processing for Neuroscientists*. Ed. by Wim van Drongelen. Burlington: Academic Press, 2007, pp. 205–217. ISBN: 978-0-12-370867-0. DOI: [10.1016/B978-012370867-0/50013-9](https://doi.org/10.1016/B978-012370867-0/50013-9).
- [Dur+16] Simon Durand et al. “Feature adapted convolutional neural networks for downbeat tracking”. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Shanghai: IEEE, Mar. 2016, pp. 296–300. DOI: [10.1109/ICASSP.2016.7471684](https://doi.org/10.1109/ICASSP.2016.7471684).

- [Dur+17] Simon Durand et al. “Robust Downbeat Tracking Using an Ensemble of Convolutional Networks”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25.1 (Jan. 2017), pp. 76–89. DOI: [10.1109/TASLP.2016.2623565](https://doi.org/10.1109/TASLP.2016.2623565).
- [Ell07] Daniel P. W. Ellis. “Beat Tracking by Dynamic Programming”. In: *Journal of New Music Research* 36.1 (Mar. 2007), pp. 51–60. DOI: [10.1080/09298210701653344](https://doi.org/10.1080/09298210701653344).
- [Far+16] Ángel Faraldo et al. “Key Estimation in Electronic Dance Music”. In: *Advances in Information Retrieval*. Ed. by Nicola Ferro et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 335–347. ISBN: 978-3-319-30671-1. DOI: [10.1007/978-3-319-30671-1_25](https://doi.org/10.1007/978-3-319-30671-1_25).
- [FJH17] Ángel Faraldo, Sergi Jordà, and Perfecto Herrera. “A Multi-Profile Method for Key Estimation in EDM”. In: *Audio Engineering Society Conference: 2017 AES International Conference on Semantic Audio*. 2017. DOI: [10.5281/zenodo.3855499](https://doi.org/10.5281/zenodo.3855499).
- [Fle+08] Arthur Flexer et al. “Playlist Generation using Start and End Songs”. In: *Proceedings of the 9th International Society for Music Information Retrieval Conference*. ISMIR, Jan. 2008, pp. 173–178. DOI: [10.5281/zenodo.1418272](https://doi.org/10.5281/zenodo.1418272).
- [Foo00] Jonathan Foote. “Automatic audio segmentation using a measure of audio novelty”. In: *2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings*. Vol. 1. July 2000, 452–455 vol.1. DOI: [10.1109/ICME.2000.869637](https://doi.org/10.1109/ICME.2000.869637).
- [FP19] Hadrien Foughmand and Geoffroy Peeters. “Deep-Rhythm for Global Tempo Estimation in Music”. In: *Proceedings of the 20th International Society for Music Information Retrieval Conference*. Delft, The Netherlands: ISMIR, Nov. 2019, pp. 636–643. DOI: [10.5281/zenodo.3527890](https://doi.org/10.5281/zenodo.3527890).
- [Fue+18] Magdalena Fuentes et al. “Analysis of Common Design Choices in Deep Learning Systems for Downbeat Tracking”. In: *Proceedings of the 19th International Society for Music Information Retrieval Conference*. Paris, France: ISMIR, Sept. 2018. DOI: [10.5281/zenodo.1492354](https://doi.org/10.5281/zenodo.1492354).
- [GM95] Masataka Goto and Yoichi Muraoka. “A Real-time Beat Tracking System for Audio Signals”. In: *ICMC Proceedings 1995*. 1995. DOI: [2027/spo.bbp2372.1995.052](https://doi.org/10.2027/spo.bbp2372.1995.052).
- [Gou+06] Fabien Gouyon et al. “An experimental comparison of audio tempo induction algorithms”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 14.5 (Sept. 2006), pp. 1832–1844. DOI: [10.1109/TSA.2005.858509](https://doi.org/10.1109/TSA.2005.858509).
- [GR08] Olivier Gillet and Gael Richard. “Transcription and Separation of Drum Signals From Polyphonic Music”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 16.3 (Mar. 2008), pp. 529–540. DOI: [10.1109/TASL.2007.914120](https://doi.org/10.1109/TASL.2007.914120).

- [GSU14] Thomas Grill, Jan Schlüter, and Karen Ullrich. “Boundary Detection in Music Structure Analysis using Convolutional Neural Networks”. In: *Proceedings of the 15th International Society for Music Information Retrieval Conference*. ISMIR, Jan. 2014. DOI: [10.5281/zenodo.1415885](https://doi.org/10.5281/zenodo.1415885).
- [HAH01] Jürgen Herre, Eric Allamanche, and Oliver Hellmuth. “Robust matching of audio signals using spectral flatness features”. In: *Proceedings of the 2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics*. New Platz, NY, USA: IEEE, 2001, pp. 127–130. DOI: [10.1109/ASPAA.2001.969559](https://doi.org/10.1109/ASPAA.2001.969559).
- [HB90] Yosef Hochberg and Yoav Benjamini. “More powerful procedures for multiple significance testing”. In: *Statistics in Medicine* 9.7 (July 1990), pp. 811–818. DOI: [10.1002/sim.4780090710](https://doi.org/10.1002/sim.4780090710).
- [HCD21] Mojtaba Heydari, Frank Cwitkowitz, and Zhiyao Duan. “BeatNet: CRNN and Particle Filtering for Online Joint Beat Downbeat and Meter Tracking”. In: *Proceedings of the 22nd International Society for Music Information Retrieval Conference*. ISMIR, 2021. DOI: [10.5281/zenodo.5624577](https://doi.org/10.5281/zenodo.5624577).
- [HD21] Mojtaba Heydari and Zhiyao Duan. “Don’t look back: an online beat tracking method using RNN and enhanced particle filtering”. In: *2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 236–240. DOI: [10.1109/ICASSP39728.2021.9413915](https://doi.org/10.1109/ICASSP39728.2021.9413915).
- [HDE09] Xiao Hu, J. Downie, and Andreas Ehmann. “Lyric Text Mining in Music Mood Classification”. In: *Proceedings of the 10th International Society for Music Information Retrieval Conference*. ISMIR, Jan. 2009, pp. 411–416. DOI: [10.5281/zenodo.1416789](https://doi.org/10.5281/zenodo.1416789).
- [HDM15] Tatsunori Hirai, Hironori Doi, and Shigeo Morishima. “MusicMixer: computer-aided DJ system based on an automatic song mixing”. In: *Proceedings of the 12th International Conference on Advances in Computer Entertainment Technology*. Association for Computing Machinery, Nov. 2015, pp. 1–5. DOI: [10.1145/2832932.2832942](https://doi.org/10.1145/2832932.2832942).
- [HDM16] Tatsunori Hirai, Hironori Doi, and Shigeo Morishima. “MusicMixer: Automatic DJ System Considering Beat and Latent Topic Similarity”. In: *MultiMedia Modeling - 22nd International Conference*. Ed. by Qi Tian et al. Springer International Publishing, 2016, pp. 698–709. DOI: [10.1007/978-3-319-27671-7_59](https://doi.org/10.1007/978-3-319-27671-7_59).
- [Hol79] Sture Holm. “A Simple Sequentially Rejective Multiple Test Procedure”. In: *Scandinavian Journal of Statistics* 6.2 (1979), pp. 65–70.
- [Hua+21] Jiawen Huang et al. “Modeling the Compatibility of Stem Tracks to Generate Music Mashups”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.1 (May 2021), pp. 187–195. DOI: [10.1609/aaai.v35i1.16092](https://doi.org/10.1609/aaai.v35i1.16092).

- [IHT09] Hiromi Ishizaki, Keiichiro Hoashi, and Yasuhiro Takishima. “Full-Automatic DJ Mixing System with Optimal Tempo Adjustment based on Measurement Function of User Discomfort”. In: *Proceedings of the 10th International Society for Music Information Retrieval Conference*. ISMIR, Jan. 2009, pp. 135–140. DOI: [10.5281/zenodo.1418230](https://doi.org/10.5281/zenodo.1418230).
- [Int15] International Telecommunication Union. *Algorithms to measure audio programme loudness and true-peak audio level*. Recommendation ITU-R BS.1770-4. Series R BS.1770-4. 2015. URL: https://assets.corusent.com/wp-content/uploads/2021/10/ITUR_BS_1770_4_Audio_Program_Loudness_En.pdf.
- [Jan+17] Andreas Jansson et al. “Singing Voice Separation with Deep U-Net Convolutional Networks”. In: *Proceedings of the 18th International Society for Music Information Retrieval Conference*. ISMIR, Oct. 2017. DOI: [10.5281/zenodo.1414933](https://doi.org/10.5281/zenodo.1414933).
- [Jeh05a] Tristan Jehan. “Creating Music by Listening”. PhD thesis. Massachusetts Institute of Technology, Jan. 2005.
- [Jeh05b] Tristan Jehan. “Downbeat prediction by listening and learning”. In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005*. New Paltz, NY, USA: IEEE, 2005, pp. 267–270. ISBN: 978-0-7803-9154-3. DOI: [10.1109/ASPAA.2005.1540221](https://doi.org/10.1109/ASPAA.2005.1540221).
- [JLL19] Bijue Jia, Jiancheng Lv, and Dayiheng Liu. “Deep Learning-Based Automatic Downbeat Tracking: A Brief Review”. In: *Multimedia Systems* 25.6 (Dec. 2019), pp. 617–638. DOI: [10.1007/s00530-019-00607-x](https://doi.org/10.1007/s00530-019-00607-x).
- [Kim14] Yoon Kim. “Convolutional Neural Networks for Sentence Classification”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Oct. 2014, pp. 1746–1751. DOI: [10.3115/v1/D14-1181](https://doi.org/10.3115/v1/D14-1181).
- [KKP08] Florian Kleedorfer, Peter Knees, and Tim Pohle. “Oh Oh Oh Whoah! Towards Automatic Topic Detection In Song Lyrics”. In: *Proceedings of the 9th International Society for Music Information Retrieval Conference*. ISMIR, Sept. 2008, pp. 287–292. DOI: [10.5281/zenodo.1416153](https://doi.org/10.5281/zenodo.1416153).
- [Kne+18] Peter Knees et al. “Two Data Sets for Tempo Estimation and Key Detection in Electronic Dance Music Annotated from User Corrections.” In: *Proceedings of the 16th International Society for Music Information Retrieval Conference (Málaga, Spain)*. ISMIR, Sept. 2018, pp. 364–370. DOI: [10.5281/zenodo.1414996](https://doi.org/10.5281/zenodo.1414996).
- [KS13] Peter Knees and Markus Schedl. “A survey of music similarity and recommendation from music context data”. In: *ACM Transactions on Multimedia Computing, Communications, and Applications* 10.1 (Dec. 2013), pp. 1–21. DOI: [10.1145/2542205.2542206](https://doi.org/10.1145/2542205.2542206).

- [KT13] Thor Kell and George Tzanetakis. “Empirical Analysis of Track Selection and Ordering in Electronic Dance Music Using Audio Feature Extraction”. In: *Proceedings of the 14th International Society for Music Information Retrieval Conference*. ISMIR, 2013. DOI: [10.5281/zenodo.1415653](https://doi.org/10.5281/zenodo.1415653). URL: <https://api.semanticscholar.org/CorpusID:5938602>.
- [KW18] Filip Korzeniowski and Gerhard Widmer. “Genre-Agnostic Key Classification with Convolutional Neural Networks”. In: *Proceedings of the 19th International Society for Music Information Retrieval Conference*. ISMIR, 2018. DOI: [10.5281/zenodo.1492399](https://doi.org/10.5281/zenodo.1492399).
- [Lai16] Steven Geoffrey Laitz. *The Complete Musician: An Integrated Approach to Theory, Analysis and Listening*. Oxford University Press, 2016. ISBN: 978-0-19-934709-4.
- [LI+24] Yizhi LI et al. “MERT: Acoustic Music Understanding Model with Large-Scale Self-supervised Training”. In: *The Twelfth International Conference on Learning Representations*. ICLR, June 2024.
- [LLT09] Heng-Yi Lin, Yin-Tzu Lin, and Ming-Chun Tien. “Music Paste: Concatenating Music Clips based on Chroma and Rhythm Features”. In: *Proceedings of the 10th International Society for Music Information Retrieval Conference*. ISMIR, Jan. 2009, pp. 213–218. DOI: [10.5281/zenodo.1415757](https://doi.org/10.5281/zenodo.1415757).
- [LSW18] Bernhard Lehner, Jan Schlüter, and Gerhard Widmer. “Online, loudness-invariant vocal detection in mixed music signals”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26.8 (2018), pp. 1369–1380. DOI: [10.1109/TASLP.2018.2825108](https://doi.org/10.1109/TASLP.2018.2825108).
- [LT07] Olivier Lartillot and Petri Toiviainen. “A Matlab Toolbox for Musical Feature Extraction from Audio”. In: *Proceedings of the 10th International Conference on Digital Audio Effects (DAFx-07)*. DAFx, 2007.
- [Lu+21] Wei-Tsung Lu et al. “SpecTNT: a Time-Frequency Transformer for Music Audio”. In: *Proceedings of the 22nd International Society for Music Information Retrieval Conference*. ISMIR, Oct. 2021. DOI: [10.5281/zenodo.5624502](https://doi.org/10.5281/zenodo.5624502).
- [Mah+05] Jose Mahedero et al. “Natural language processing of lyrics”. In: *Proceedings of the 13th annual ACM international conference on Multimedia*. Association for Computing Machinery, Nov. 2005, pp. 475–478. DOI: [10.1145/1101149.1101255](https://doi.org/10.1145/1101149.1101255).
- [Mah16] Robert Mahieu. *Detecting Musical Key with Supervised Learning*. Tech. rep. Department of Electrical Engineering, Stanford University, 2016.
- [McC19] Matthew C. McCallum. “Unsupervised Learning of Deep Features for Music Segmentation”. In: *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, May 2019, pp. 346–350. DOI: [10.1109/ICASSP.2019.8683407](https://doi.org/10.1109/ICASSP.2019.8683407).

- [Mik+13a] Tomas Mikolov et al. “Distributed Representations of Words and Phrases and their Compositionality”. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. Curran Associates Inc., 2013, pp. 3111–3119. DOI: [10.5555/2999792.2999959](https://doi.org/10.5555/2999792.2999959).
- [Mik+13b] Tomas Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. In: *International Conference on Learning Representations*. ICLR, Jan. 2013.
- [Mit+22] Yuki Mitsufuji et al. “Music Demixing Challenge 2021”. In: *Frontiers in Signal Processing* 1 (2022). DOI: [10.3389/frsip.2021.808395](https://doi.org/10.3389/frsip.2021.808395).
- [Mix] Mixed In Key. *Harmonic Mixing Guide*. URL: <https://mixedinkey.com/harmonic-mixing-guide/> (visited on 12/13/2023).
- [MMF+23] Brian McFee, Matt McVicar, Daniel Faronbi, et al. *librosa/librosa: 0.10.1*. Aug. 2023. DOI: [10.5281/zenodo.8252662](https://doi.org/10.5281/zenodo.8252662).
- [MNR08] Rudolf Mayer, Robert Neumayer, and Andreas Rauber. “Rhyme and Style Features for Musical Genre Classification by Song Lyrics”. In: *Proceedings of the 9th International Society for Music Information Retrieval Conference*. ISMIR, Sept. 2008, pp. 337–342. DOI: [10.5281/zenodo.1416757](https://doi.org/10.5281/zenodo.1416757).
- [NB14] Oriol Nieto and Juan Pablo Bello. “Music segment similarity using 2D-Fourier Magnitude Coefficients”. In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Florence, Italy: IEEE, May 2014, pp. 664–668. DOI: [10.1109/ICASSP.2014.6853679](https://doi.org/10.1109/ICASSP.2014.6853679).
- [NB16] Oriol Nieto and Juan Pablo Bello. “Systematic exploration of computational music structure research”. In: *Proceedings of the 17th International Society for Music Information Retrieval Conference*. ISMIR, 2016. DOI: [10.5281/zenodo.1417660](https://doi.org/10.5281/zenodo.1417660).
- [Orf96] Sophocles J. Orfanidis. *Introduction to signal processing*. Prentice Hall signal processing series. Englewood Cliffs, N.J: Prentice Hall, 1996. ISBN: 978-0-13-209172-5.
- [Pam06] Elias Pampalk. “Computational models of music similarity and their application in music information retrieval”. PhD thesis. Technische Universität Wien, 2006. DOI: [20.500.12708/12837](https://doi.org/20.500.12708/12837).
- [Pan+17] Maria Panteli et al. “A model for rhythm and timbre similarity in electronic dance music”. In: *Musicae Scientiae* 21.3 (Sept. 2017), pp. 338–361. DOI: [10.1177/1029864916655596](https://doi.org/10.1177/1029864916655596).
- [PBH14] Maria Panteli, Niels Bogaards, and Aline Honingh. “Modeling Rhythm Similarity for Electronic Dance Music”. In: *Proceedings of the 15th International Society for Music Information Retrieval Conference*. ISMIR, 2014, pp. 537–542. DOI: [10.5281/zenodo.1416663](https://doi.org/10.5281/zenodo.1416663).

- [Pee+11] Geoffroy Peeters et al. “The Timbre Toolbox: extracting audio descriptors from musical signals”. In: *The Journal of the Acoustical Society of America* 130.5 (Nov. 2011), pp. 2902–2916. DOI: [10.1121/1.3642604](https://doi.org/10.1121/1.3642604).
- [PM95] John G. Proakis and Dimitris G. Manolakis. *Digital Signal Processing: Principles, Algorithms and Applications*. 3rd Edition. Upper Saddle River: Prentice Hall, Oct. 1995. ISBN: 978-0-13-373762-2.
- [Poh+09] Tim Pohle et al. “On Rhythm and General Music Similarity”. In: *Proceedings of the 10th International Society for Music Information Retrieval Conference*. ISMIR, Jan. 2009, pp. 525–530. DOI: [10.5281/zenodo.1418228](https://doi.org/10.5281/zenodo.1418228).
- [PP11] H el ene Papadopoulou and Geoffroy Peeters. “Joint estimation of chords and downbeats from an audio signal”. In: *IEEE Transactions on Audio, Speech and Language Processing* 19.1 (Jan. 2011), pp. 138–152. DOI: [10.1109/TASL.2010.2045236](https://doi.org/10.1109/TASL.2010.2045236).
- [RBH13] Bruno Rocha, Niels Bogaards, and Aline Honingh. *Segmentation and timbre- and rhythm-similarity in Electronic Dance Music*. Tech. rep. University of Amsterdam, Elephantcandy, Apr. 2013. DOI: [11245/1.402415](https://doi.org/11245/1.402415).
- [RG19] Nils Reimers and Iryna Gurevych. “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Nov. 2019, pp. 3982–3992. DOI: [10.18653/v1/D19-1410](https://doi.org/10.18653/v1/D19-1410).
- [RMD23] Simon Rouard, Francisco Massa, and Alexandre D efossez. “Hybrid Transformers for Music Source Separation”. In: *2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023. DOI: [10.1109/ICASSP49357.2023.10096956](https://doi.org/10.1109/ICASSP49357.2023.10096956).
- [Roy19] Th eo Royer. “Pitch-shifting algorithm design and applications in music”. MA thesis. KTH Royal Institute of Technology, School of Electrical Engineering and Computer Science, 2019.
- [RSD13] Andrew Robertson, Adam Stark, and Matthew E P Davies. “Percussive Beat tracking using real-time median filtering”. In: *Proc. 6th Int. Workshop Mach. Learn. Music (MML 13)*. ECML PKDD, 2013, pp. 71–75.
- [SED18] Daniel Stoller, Sebastian Ewert, and Simon Dixon. “Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation”. In: *Proceedings of the 19th International Society for Music Information Retrieval Conference*. ISMIR, 2018. DOI: [10.5281/zenodo.1492416](https://doi.org/10.5281/zenodo.1492416).
- [Ser+12] Joan Serr a et al. “Unsupervised Detection of Music Boundaries by Time Series Structure Features”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 26.1 (2012), pp. 1613–1619. DOI: [10.1609/aaai.v26i1.8328](https://doi.org/10.1609/aaai.v26i1.8328).

- [Ser+14] Joan Serrà et al. “Unsupervised Music Structure Annotation by Time Series Structure Features and Segment Similarity”. In: *IEEE Transactions on Multimedia* 16.5 (Aug. 2014), pp. 1229–1240. DOI: [10.1109/TMM.2014.2310701](https://doi.org/10.1109/TMM.2014.2310701).
- [Sha11] Ibrahim Sha’ath. “Estimation of key in digital music recordings, MSc Computer Science Project Report”. MA thesis. Birkbeck College, University of London, 2011.
- [Sha95] Juliet Popper Shaffer. “Multiple Hypothesis Testing”. In: *Annual Review of Psychology* 46.1 (1995), pp. 561–584. DOI: [10.1146/annurev.ps.46.020195.003021](https://doi.org/10.1146/annurev.ps.46.020195.003021).
- [Shi07] John Shiga. “Copy-and-Persist: The Logic of Mash-Up Culture”. In: *Critical Studies in Media Communication* 24.2 (June 2007), pp. 93–114. DOI: [10.1080/07393180701262685](https://doi.org/10.1080/07393180701262685).
- [SM18] Hendrik Schreiber and Meinard Müller. “A Crowdsourced Experiment for Tempo Estimation of Electronic Dance Music”. In: *Proceedings of the 19th International Society for Music Information Retrieval Conference*. ISMIR, 2018. DOI: [10.5281/zenodo.1492436](https://doi.org/10.5281/zenodo.1492436).
- [SSH16] Carl Southall, Ryan Stables, and Jason Hockman. “Automatic Drum Transcription Using Bi-Directional Recurrent Neural Networks.” In: *Proceedings of the 17th International Society for Music Information Retrieval Conference* (New York City, United States). ISMIR, Sept. 2016, pp. 591–597. DOI: [10.5281/zenodo.1416244](https://doi.org/10.5281/zenodo.1416244).
- [SSH17] Carl Southall, Ryan Stables, and Jason Hockman. “Automatic drum transcription for polyphonic recordings using soft attention mechanisms and convolutional neural networks”. In: *Proceedings of the 18th International Society for Music Information Retrieval Conference*. ISMIR, 2017. DOI: [10.5281/zenodo.1415615](https://doi.org/10.5281/zenodo.1415615).
- [Ste10] John Steventon. *DJing For Dummies*. 2nd Edition. For Dummies, Sept. 2010. ISBN: 978-0-470-66372-1.
- [SWB20] Shashank Sonkar, Andrew E. Waters, and Richard G. Baraniuk. “Attention Word Embedding”. In: *Proceedings of the 28th International Conference on Computational Linguistics*. International Committee on Computational Linguistics, Dec. 2020, pp. 6894–6902. DOI: [10.18653/v1/2020.coling-main.608](https://doi.org/10.18653/v1/2020.coling-main.608).
- [SWP10] Klaus Seyerlehner, Gerhard Widmer, and Tim Pohle. “Fusing Block-level Features for Music Similarity Estimation”. In: *Proceedings of the 13th International Conference on Digital Audio Effects (DAFx-10)*. DAFx, 2010.

- [TGM18] Naoya Takahashi, Nabarun Goswami, and Yuki Mitsufuji. “MMDenseLSTM: An efficient combination of convolutional and recurrent neural networks for audio source separation”. In: *2018 16th International workshop on acoustic signal enhancement (IWAENC)*. IEEE, 2018, pp. 106–110. DOI: [10.1109/IWAENC.2018.8521383](https://doi.org/10.1109/IWAENC.2018.8521383).
- [TM21] Naoya Takahashi and Yuki Mitsufuji. “Densely connected multidilated convolutional networks for dense prediction tasks”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2021, pp. 993–1002. DOI: [10.1109/CVPR46437.2021.00105](https://doi.org/10.1109/CVPR46437.2021.00105).
- [Vas01] Pantelis Vassilakis. “Perceptual and Physical Properties of Amplitude Fluctuation and their Musical Significance”. Doctoral Dissertation. University of California, Los Angeles, Jan. 2001. ISBN: 978-0-493-45511-2.
- [Vas07] Pantelis N Vassilakis. “SRA: A Web-based Research Tool for Spectral and Roughness Analysis of Sound Signals”. In: *Proceedings of the 4th Sound and Music Computing (SMC) Conference*. SMC, 2007, pp. 319–325. DOI: [10.5281/zenodo.849474](https://doi.org/10.5281/zenodo.849474).
- [VD18] Len Vande Veire and Tijl De Bie. “From raw audio to a seamless mix: creating an automated DJ system for Drum and Bass”. In: *EURASIP Journal on Audio, Speech, and Music Processing* 2018.1 (Sept. 2018), p. 13. DOI: [10.1186/s13636-018-0134-8](https://doi.org/10.1186/s13636-018-0134-8).
- [Vog+17] Richard Vogl et al. “Drum Transcription via Joint Beat and Drum Modeling Using Convolutional Recurrent Neural Networks”. In: *Proceedings of the 18th International Society for Music Information Retrieval Conference*. ISMIR, 2017. DOI: [20.500.12708/57031](https://doi.org/20.500.12708/57031).
- [VP20] Michaela Vystrčilová and Ladislav Peška. “Lyrics or Audio for Music Recommendation?” In: *Proceedings of the 10th International Conference on Web Intelligence, Mining and Semantics*. ACM, June 2020, pp. 190–194. DOI: [10.1145/3405962.3405963](https://doi.org/10.1145/3405962.3405963).
- [VTH23] Julian Von Der Mosel, Alexander Trautsch, and Steffen Herbold. “On the Validity of Pre-Trained Transformers for Natural Language Processing in the Software Engineering Domain”. In: *IEEE Transactions on Software Engineering* 49.4 (Apr. 2023), pp. 1487–1507. DOI: [10.1109/TSE.2022.3178469](https://doi.org/10.1109/TSE.2022.3178469).
- [WHS22] Ju-Chiang Wang, Yun-Ning Hung, and Jordan B. L. Smith. “To Catch A Chorus, Verse, Intro, or Anything Else: Analyzing a Song with Structural Functions”. In: *2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, May 2022, pp. 416–420. DOI: [10.1109/ICASSP43922.2022.9747252](https://doi.org/10.1109/ICASSP43922.2022.9747252).

- [Won+20] Minz Won et al. “Data-Driven Harmonic Filters for Audio Representation Learning”. In: *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, May 2020, pp. 536–540. ISBN: 978-1-5090-6631-5. DOI: [10.1109/ICASSP40776.2020.9053669](https://doi.org/10.1109/ICASSP40776.2020.9053669).
- [WSW21] Hei-Chia Wang, Sheng-Wei Syu, and Papis Wongchaisuwat. “A method of music autotagging based on audio and lyrics”. In: *Multimedia Tools and Applications* 80.10 (Apr. 2021), pp. 15511–15539. DOI: [10.1007/s11042-020-10381-y](https://doi.org/10.1007/s11042-020-10381-y).
- [Wu+18] Chih-Wei Wu et al. “A Review of Automatic Drum Transcription”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26.9 (Sept. 2018), pp. 1457–1483. DOI: [10.1109/TASLP.2018.2830113](https://doi.org/10.1109/TASLP.2018.2830113).
- [WWS21] I-Chieh Wei, Chih-Wei Wu, and Li Su. “Improving Automatic Drum Transcription Using Large-Scale Audio-to-Midi Aligned Data”. In: *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, June 2021, pp. 246–250. ISBN: 978-1-72817-605-5. DOI: [10.1109/ICASSP39728.2021.9414409](https://doi.org/10.1109/ICASSP39728.2021.9414409).
- [Xia+13] Yang Xiang et al. “Generalized Simulated Annealing for Global Optimization: The GenSA Package”. In: *The R Journal* 5.1 (2013), pp. 13–28. DOI: [10.32614/RJ-2013-002](https://doi.org/10.32614/RJ-2013-002).
- [YLL21] Shingchern D You, Chien-Hung Liu, and Jia-Wei Lin. “Improvement of Vocal Detection Accuracy Using Convolutional Neural Networks.” In: *KSII Transactions on Internet & Information Systems* 15.2 (2021). DOI: [10.3837/tiis.2021.02.019](https://doi.org/10.3837/tiis.2021.02.019).
- [Zhu+97] Ciyong Zhu et al. “Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization”. In: *ACM Transactions on Mathematical Software* 23.4 (1997), pp. 550–560. DOI: [10.1145/279232.279236](https://doi.org/10.1145/279232.279236).