Jazz Drum Machine: A Novel Interface for Data-Driven Remixing of Music

Ben Lacker
327 Bedford Avenue, #B6, Brooklyn, NY 11211
benlacker@gmail.com

Abstract

Jazz Drum Machine (http://jazzdrummachine.com) is a web application written in Javascript that uses the Echo Nest Remix API to intelligently remix music. An input song is segmented into discrete musical events, which are then grouped according to pitch. Users are presented with an interface that allows them to select the number of segments from each group that will be heard in the final mix. The final mix is generated in real time and preserves rhythmic features extracted from the original song while reordering segments based on user input.

Introduction

Advances in the fields of Music Information Retrieval and Machine Listening have made it possible for computers to rapidly extract musically meaningful data from digital audio signals. These techniques have been used successfully for applications such as audio identification, score following, and music recommendation. Computerized audio analysis also provides a wealth of possibilities for interacting with and composing music.

Several examples exist of software focused on creating new music by cutting up and dramatically re-sequencing existing music. Among the most notable are Nick Collins's BBCut library, which provides a framework for automatically cutting up audio input (Collins 2002). While BBCut library offers powerful tools for cutting up music, it intentionally provides little guidance in the re-sequencing of these cuts. CataRT, on the other hand, is a software instrument that focuses more on the interface for re-sequencing: it analyzes a corpus of sounds and plots them on a two-dimensional grid, allowing users to "play" the corpus by navigating through the grid (Schwarz 2008).

The Echo Nest Remix API[1] is a toolkit for using automatic audio analysis data to compose and interact with music. This API makes available detailed information about any digital audio file's pitch, timbre, and rhythmic structure (Jehan 2011). This data offers a wealth of possibilities for musical interaction, and the API offers an accessible way to both cut and re-sequence music algorithmically.

Developers have used the Echo Nest Remix API to build applications that automatically create dubstep remixes[2] and create never-ending and always changing versions of songs[3]. These existing applications tend to produce results in keeping with a more traditional concept of remixing: a new version of an existing song where the original is completely recognizable. Jazz Drum Machine seeks to explore a different subset of the possibilities offered by this API, to use audio analysis data to create a simple musical instrument out of an existing song, and to preserve certain aspects of the original music while discarding many others.

Description of Web Application for Intelligent Musical Remixing

Overview

Jazz Drum Machine (http://jazzdrummachine.com) is a web application written entirely in Javascript. It presents a novel interface for musical remixing in the form of a single web page. The application remixes audio files according to musical features extracted through audio analysis. These features are used to split the input file into distinct musical events, which are then grouped into clusters according to pitch. These clusters of sounds form the basis of rhythmic

[1] The Echo Nest Remix API: http://echonest.github.io/remix/
[2] Peter Sobot's *The Wub Machine*: https://the.wubmachine.com/
[3] Paul Lamere's *Infinite Jukebox*: http://infinitejuke.com

loops, where the user can control how many sounds from each cluster are present during playback. The onset times for playback of musical events are derived from the original audio file, while the decision of which events to play back is controlled by the user. Thus Jazz Drum Machine provides an interface for exploring existing audio recordings, manipulating their playback to reveal subtleties in the existing material and to create new music.

Audio Analysis

Jazz Drum Machine takes as input a single digital audio file. While non-musical audio may be used and may produce interesting effects, the application was designed to manipulate musical audio, with a particular focus on rhythmic characteristics. Users can upload an audio file of their choice for remixing, or use one of several example songs provided. The example songs, which are recordings of solos by famous jazz drummers, were chosen to highlight the application's tendency to bring a song's rhythmic peculiarities to the foreground.

Once an input file has been selected, it is analyzed using The Echo Nest's Audio Analysis API[4]. This API provides automatic description of the musical features of an audio file, including tempo, key, pitch, timbre, and detailed rhythmic information. The rhythmic features include the onset times of every section, bar, beat, and segment in a song. As defined by the Echo Nest, a segment is the smallest possible musical subdivision of a beat, and is relatively uniform in terms of timbre and harmony. Each segment is characterized by its pitch, timbre, loudness, and duration.

Clustering of Audio Segments

The audio features provided by the Echo Nest API are used first to split the input file into distinct musical events. For example, when a recording of a drum solo is analyzed, each successive attack will be described as an audio segment with a distinct pitch and timbre. If the performer hits a floor tom, that segment's pitch will reflect the tuning of the drum; if she then hits a cymbal, a kick drum, and snare drum a the same time, that segment will be considerably more noisy, and the pitch descriptor will reflect that.

Once the input file has been split into musical segments, these segments are then divided into eight groups based on their pitch content. A k-means clustering algorithm is used to place segments into clusters based on similarities in pitch. This algorithm causes harmonically similar sounds to end up in the same group. To use again the example of a jazz drum solo, this approach yields clusters of segments that loosely correspond to the elements of a drum kit: snare drum hits, low tom hits, kick drum hits, kick drum and cymbal hits together, rim shots. A more traditionally melodic song, such as a solo flute recording, would be split into clusters corresponding to pitch classes: the tonic note, the dominant note, etc.

This simple approach to clustering is sufficient for this application. The goal of Jazz Drum Machine is not to classify a song's sounds into precise, intelligible groups. It is enough that the segments in each group are audibly similar, as well as audibly different from the segments in the other groups. Fuzziness about the real-world meaning of the clusters does not detract from the artistic purposes of the application.

User Interface

The user interface consists of a Play button, a Stop button, and eight sliders. Each slider corresponds to a single cluster of segments, and the slider's value determines how many segments from that cluster will be heard during playback. Raising a slider halfway will cause half of the segments from the corresponding cluster to be heard during playback; raising it just a little bit will cause only the first segment in the cluster to be heard; raising it all the way will cause all the segments in that cluster to be heard during playback. The order in which segments are chosen from a cluster is based on the segments' distance from the center of the cluster. The first segment chosen for playback from a cluster will be the most central, or the most representative of its cluster, in terms of pitch content. As the slider is raised further, the segments selected include ones further and further from the center of the cluster.

The simplicity of the interface is intentional; it is designed to be accessible to people with no musical background. The design also encourages users to approach the application like a found musical instrument or toy: to understand it by playing with it.

Playback

When the user clicks the Play button, Jazz Drum Machine begins playing a loop whose length corresponds to one bar of the input song. This loop will contain all the segments selected for playback by the user's manipulation of sliders. Each segment is played back according to its relative position within the bar in the original song. In other words, a segment's start time within the playback loop is determined by taking the segment's start time in the original song and subtracting the start time of the bar that contains it. This is perhaps the most important aspect of Jazz Drum Machine: the preservation of the input song's rhythmic information, even while the order of its segments is changed.

[4] The Echo Nest API: http://developer.echonest.com/

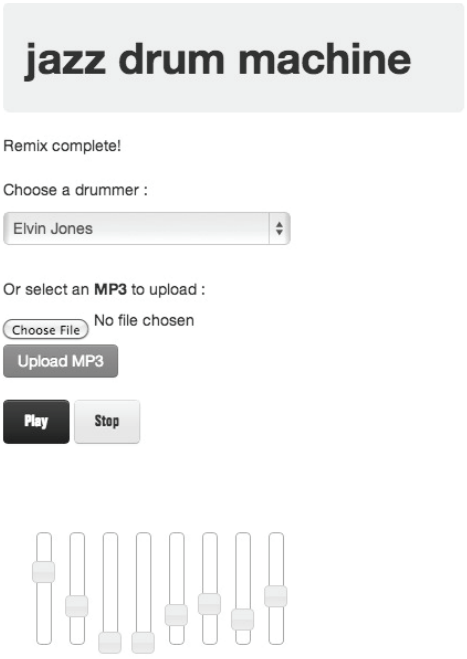

Figure 1. User Interface of Jazz Drum Machine

For example, if a particular segment originally occurred on the second beat of a measure, it will be heard on the second beat of the playback loop. If another segment originally occurred slightly before the third beat of a different measure, it will be heard just before beat three in the playback loop. This is where the jazz drum solos chosen as example tracks prove especially illuminating: drum solos illustrate the tendency of sound clusters based on pitch to exhibit different rhythmic properties. Because the segment clusters derived from drum solos tend to correspond loosely to different elements of a drum kit, each cluster tends to maintain the rhythmic characteristics of that element of the kit. For instance, in jazz and pop music the snare drum is frequently played on the second and fourth beats of a measure. If a user raises the slider of the cluster associated with snare drum sounds, the segments will mostly be played back on beats two and four of the playback loop.

Segments from every bar of the original song are superimposed onto the one-bar playback loop. If every slider is raised to its maximum height, every segment from the original song will be heard in the playback loop. This tends to sound cacophonous, but illustrates the workings of Jazz Drum Machine: every bar from the original input song is superimposed onto a one-bar loop, and by controlling the eight sliders the user can choose which types of sounds will be heard in the playback loop.

By manipulating the sliders in real time, the user can create a great variety of loops, all consisting of sounds from the original song. These loops completely discard the original ordering of segments while maintaining the rhythmic feeling of the original song. Because segments are played back according to their original times within the measure, Jazz Drum Machine preserves the rhythmic idiosyncrasies of its input: techno music sounds just as quantized, reggae music maintains its slight swing, and jazz drum solos bear the rhythmic imprints of the soloists.

Conclusions

This simple interface has proven to be remarkably expressive, teasing interesting nuances out of familiar music. It highlights the way certain pitches and types of sounds tend to occur at certain beats within the measure. The rhythmic "feel" of an input song is brought to the foreground, while the original structure, melody, and harmonic progression are obliterated. Jazz Drum Machine offers a way for musicians and non-musicians alike to explore existing sounds and to use them as raw material for creating new music. The author sees this application as an early step in the work of using the wealth of data offered by automatic musical analysis for creative purposes.

References

Collins, N. 2002b. The BBCut Library. In Proceedings of the International Computer Music Conference, 313-316. Goteborg, Sweden.

Jehan, T., and DesRoches, D. 2011. Analyzer Documentation. The Echo Nest. Somerville, MA.

Schwarz, D.; Cahen, R.; and Britton, S. 2008. Principles and Applications of Interactive Corpus-Based Concatenative Synthesis. In *Journées d'Informatique Musicale (JIM)*, Albi, France: GMEA. [Online]. Available: http://articles.ircam.fr/textes/Schwarz08a/