Finding "The Drop": Recognizing the climax in electronic music using classification models.

Koen van den Brink
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
k.vandenbrink@student.utwente.nl

ABSTRACT

Electronic dance music (in short: *EDM*) is a collection of music genres often listened to at festivals and clubs by younger audiences. DJs often use structural information of *EDM* music to transition seamlessly between songs. A key structural component that is used often by DJs, and therefore has to be labeled by hand often is the climax of the songs (also referred to as the *drop*. This paper proposes two machine learning model designs, a hand-labeled dataset and training data pre-processing techniques to identify the *drop* in *EDM* songs. I process the curated dataset in three different ways and train two different classification models with all three resulting training sets. After this, I evaluate the six different combinations and compare them against each other. I evaluate all combinations on my own evaluation set and an evaluation set of an earlier research that looks promising. The proposed models outperform existing models with similar window sizes. A convolutional neural network trained with a specific data processing technique outperforms an existing model with a much larger (275% increase) window size. The models proposed in this paper achieve better results than earlier models, even when presenting the models with a fraction of the audio context.

Keywords

Music structure annotation, Music information retrieval (MIR), segment labelling, data pre-processing, convolutional neural network (CNN), support vector machine (SVM), electronic dance music (*EDM*).

1. INTRODUCTION

Electronic dance music is an umbrella term for a wide range of music genres, often listened to during larger social gatherings, such as *EDM* festivals. An iconic moment in *EDM* music is the climax of the song. This section in a song is comparable to a chorus of a song in a more traditional genre. The exact starting point (or on-set) of the chorus in *EDM* songs is referred to as *the drop*, named after the fact that this event is almost always preceded by a *build-up*. This is a section that builds up tension, all of which is released (*dropped*) at *the drop*.

33rd Twente Student Conference on IT July. 3rd, 2020, Enschede, The Netherlands.

Many music information retrieval (*MIR*) implementations are used in DJ software. An example of this is beat on-set detection [1]. This provides information to the DJ to provide a better music listening experience for their audience. This paper proposes a system using machine learning that provides the DJ with extra information about song structure of *EDM* songs, so the DJ can focus more on expressing creativity through their sets.

1.1 Objectives

The goal for this research is to classify the on-set (starting timestamp) of the chorus(ses) in a song. In other words: the paper aims to identify the *drop* of *EDM* songs.

Previous research has been with rather small datasets, which is something this research also improves upon. These datasets often did not have more than one- to two hundred songs. This research hopes to improve by using a dataset of around five hundred curated songs. The impact of using this new dataset for training models is not specifically tested and discussed in this paper. However, the proposed models of this paper will be evaluated on external (previously proposed) datasets.

This research also aims to find the impact of different ways of processing the training data on the performance of the model. Three forms of processing are used and compared to a model that was trained with data that went through minimal processing.

1.2 Related Works

Models aiming to find a certain event in music often follow a two-step process of which the first is to run a segmentation algorithm. These algorithms groups similar audio frames together and therefore generates segment boundaries in the audio. The second step is to classify these segments and generate the contextual information about the boundaries. The issue with finding a certain part of a song is the act of correctly labelling these boundaries.

First, segmentation is the process of splitting a song into parts that are significantly different from other parts. For example, segmentation separates two verses and the build-up from the chorus. Most implementations for segmentation follow a process where a figure is created, often referred to as a similarity matrix [9].

The similarity matrices are created in multiple different ways, one of which is based on histogram clustering [7]. This technique has a better performance in aligning with human hearing. Other techniques try to use structures that humans often use to visualize music. For example, spectrograms, chroma features (what notes are played over what time period), RGB-frequency-split (band frequency filters combined into an RGB picture) waveforms (raw signal), etc. [3]. Features such as chroma and MFCCs can also be 'fused' together [11]. Fusion is combining multiple similarity matrices to make sure multiple audio features are similar at specific points in time. This fusion makes sure multiple structural components of the audio are similar, which can be used to determine contextual context of a segment segment. For example, vocals usually have quite similar chroma features, as the frequency of voice is more or less constant. However, MFCCs will still be quite different as the sung words are different. Many of these techniques have been implemented by open-source Python libraries such as Librosa, which was first presented in 2015 [6]. Librosa integrates many of these state-of-the-art segmentation principles such as temporal segmentation [5] and laplacian segmentation [4].

A promising implementation of this is from Yadati et al. [13], where they used a *binary support vector machine* (SVM) classifier. This classifier received the boundaries in the track which came from segmentation, as well as some other audio features like the *mel frequency cepstral coefficients* (MFCC). A classification model was trained to distinguish segments on whether it they are likely to be the on-set of a *drop* or not.

Most often, the models used for contextual labelling are SVMs or convolutional neural networks (CNNs). LSTMs have also been tried in previous research [12]. However, the latter model type did not have promising outcomes.

2. RESEARCH QUESTIONS

The research questions were formulated aimed at the task of labelling segment boundaries as follows:

RQ 1: What type of classification model, between SVMs and CNNs is most suitable to find the on-set of a *drop* in *EDM* songs?

RQ 2: Do different pre-processing techniques have an impact on the performance of the models presented in this paper?

RQ 3: Does one of the proposed models outperform Yadati's (2014) + implementation on their testing dataset?

3. DATASET

This paper proposes a new dataset that I designed. This dataset contains around 500 *EDM* songs with their manually labeled ground-truth labels of the on-set of the *drops*. This dataset will be referred to as the *curated dataset*. In the curated dataset, all song's *drops* are noted as timestamps.

This curated dataset was hand-labeled by two curators. Custom labeling software was made by me beforehand to make the labelling as quick as possible. All songs' drops were labeled twice (by each curator). All *drop* timestamps were tested to be within one second from each other. If the labels were within this distance from each other, the average off these labels was taken to be the true value and inserted in the final curated dataset, along with the raw audio data. Whenever the two labels are within the given distance from each other, the standard deviation between the two labels in the dataset is 0.09 seconds. This means that on average the two curators independently labeled drops within 0.09 seconds from other.

In figure 1, you can see an example of the raw audio (as a waveform) of an electronic dance music track with the curated drops.

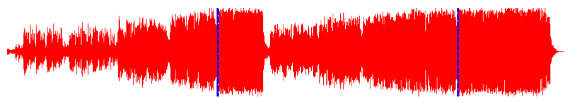

Figure 1. Waveform of an *EDM* track with the two drops marked as dotted blue lines.

4. METHODOLOGY

4.1 Pre-Processing

Training data is automatically generated from the curated dataset. The training data is presented in the form of spectrograms (as images). A spectrograms testing a certain on-set will range from 2 seconds before the on-set until 2 seconds after. This is done to give models some audio context of the audio event that is taking place at the middle of the spectrogram. This means all training data entries are labeled 4-second spectrograms. All class labels are whether the spectrogram either defines a *drop* or not.

4.1.1 Base Training Set

The base training set has no pre-processing of audio or onsets of the spectrograms. The spectrograms representing the drops are directly created from the raw audio and the *drop* timestamps in the curated dataset. When the spectrogram is created, it will be a spectrogram representing 4 seconds of audio, of which we want the model to identify the exact middle to be an on-set of a *drop*. The other spectrograms, of which we would like the model to identify the middle as a negative sample, are created with a randomly chosen timestamp anywhere within the song. However, these timestamp are not chosen completely at random, as they need to adhere to the following two constraints:

1. The timestamp may not lie within the first or last two seconds of a song, as this would clip the spectrogram.
2. The timestamp may not lie within two seconds of a *drop* timestamp for the respective song.

In figure 2 below, you can see examples of two spectrograms representing a randomly chosen timestamp and a *drop* on-set.

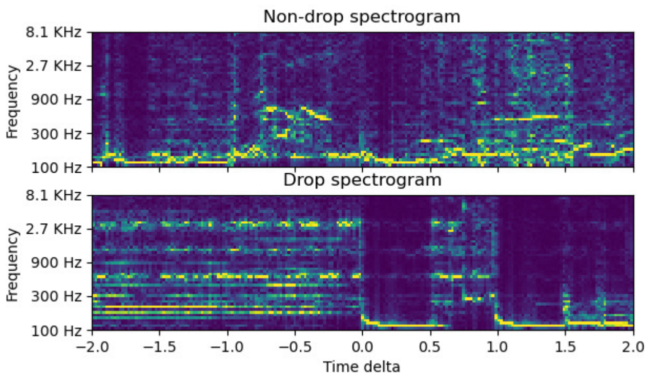

Figure 2. Examples of spectrograms for the base training set.

4.1.2 *Aligned Training Set*

The aligned training set is similar to the base training set. It uses two types of spectrograms, representing *drop* on-sets and other timestamps in the audio. However, for this training set, I have used some pre-processing to make false spectrograms less random and more representative of what a model could expect in an implementation.

First, the song audio is segmented using a previous (state of the art) implementation. This produces segment boundaries, which are iterated and checked against the hand-curated *drop* on-sets. If a segment is within 0.1 seconds of a *drop* on-set, then this boundary is marked as a *drop* on-set boundary and a spectrogram is created and labeled to be a drop. If no boundary was found to be within that constraint, spectrograms are created from the *drop* on-sets in the curated dataset. These spectrograms are then labeled to be representing *drop* on-sets.

After this, false spectrograms are created (similar to the base training set). However, the chosen timestamps have to follow certain constraints. In addition to complying with the two rules for the base training set, these timestamps have to be chosen between the segment boundaries that resulted from the song segmentation. This way, we are sure that every spectrogram represents a significant change within the song.

In figure 3 below, you can see two example spectrograms with this type of pre-processing.

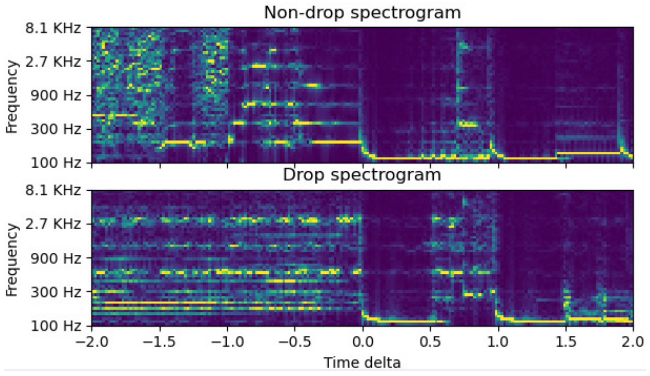

Figure 3. Examples of spectrograms for the aligned training set.

What can be noted in these spectrograms is that the structure of the audio is clearly different left and right of the middle in both spectrograms. Therefore, both spectrograms clearly represent some change in audio context at 0.0 seconds.

4.1.3 *Equalised Training Set*

Lastly, the equalised training set follow the exact same steps as the base training set. However, instead of the raw audio being used, this training set is created from filtered (equalised) audio. The audio is filtered with a function that applies a gain (volume multiplier) to specific frequency ranges. This frequency-gain function can be seen in figure 4.

As seen in figure 4, low frequencies are kept at full gain, while middle-range and higher-range frequencies are lowered and even cut off respectively. Human DJs often have waveforms split into the waveforms of three different frequency ranges: low, mid and high frequencies. These waveforms are then combined into one by combining the three waveforms into the three different colour channels of the final waveform. Of this RGB waveform, DJs usually

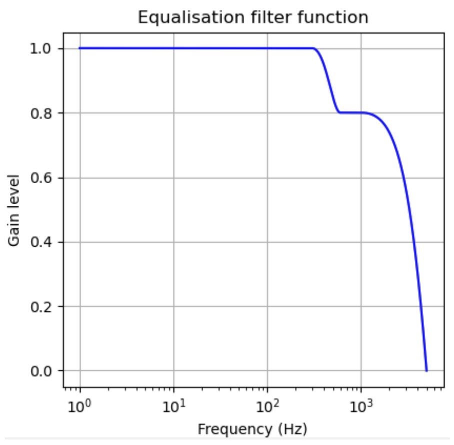

Figure 4. Frequency-gain function for the equalised training set.

only consider the waveform corresponding to the low frequencies most [10]. Therefore, this filtering could help the model learn the problem more quickly and effectively, as humans do.

4.1.4 *Evaluation Set*

For evaluation purposes, a separate evaluation set is created. This evaluation set is generated from the curated dataset, similar to the training sets. However, the evaluation set does not do any pre processing of any data, and only contains labeled spectrograms. These spectrograms are generated from the resulting boundaries from running segmentation. These boundaries are then labeled based on their proximity to *drop* timestamps from the curated dataset. An important difference between the valuation data sets and any of the training sets is that some tracks might not have any *drop* spectrograms. This is because the segmentation sometimes misses the segment boundary representing the *drop* of a song. When this happens, a song in the evaluation set will only have non-*drop* spectrograms.

4.2 Model design

As stated in the research questions, we're looking at 2 different machine learning models in this paper: a support vector machine and a convolutional neural network. The code that was used to create the models can be found on this paper's GitLab repository [1].

4.2.1 *Support Vector Machine*

I implemented the SVM and optimised it slightly based on some systematic trial-and-error. The chosen kernel function is $\exp(-\gamma\|x-x'\|^2)$, where γ is defined as $\frac{1}{n\_features}$. This function is often referred to as the *RBF* function. The tolerance was set to $1e^{-3}$. All other parameters, such as the shrinking heuristic, were not experimented with extensively and therefore kept to the defaults of *Scikit-Learn*'s SVM defaults [8].

4.2.2 *Convolutional Neural Network*

As spectrograms are time-series, I used a 1-dimensional CNN. The convolutions' kernels and the other layers were defined by systematic trial-and-error. However, optimisations of the network design are most likely possible for

[1]The GitLab repository is hosted by the University of Twente and can be found with the following link: `https://git.snt.utwente.nl/s1932195/finding-the-drop`.

future works. The network consists of a small stack of convolutional- and pooling layers. The first convolutional layer has 32 filters and a kernel size of 3. The max-pooling layer after this has a pool size of 2. The third layer (and second convolutional layer) has 64 filters and a kernel size of 3. The fourth layer (and second max-pooling layer) has a pool size of 2. The input is then flattened and fed into the second stage of the neural network. There are two fully connected layers, the latter of which marks the classification output. To prevent over-fitting, a dropout layer was introduced between these two. All layers have a rectified linear unit (*ReLU*) activation function, except the output layers. The output layer has a softmax activation function. In figure 5, you can see the layout of the model, as defined in Keras [2]:

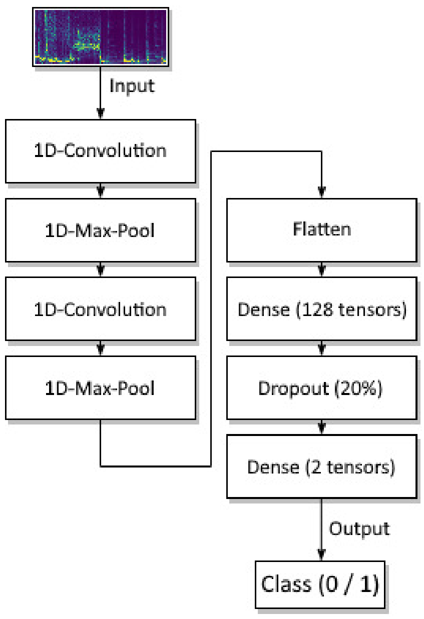

Figure 5. Design of the proposed convolutional neural network.

Multiple different combinations of layers have been tried. Kernels of the convolution layers have been fine-tuned by hand until the model seemed to perform best for this specific evaluation set.

5. RESULTS

5.1 Evaluation Metrics

As this is a binary classification problem, true positives (*tp*), false positives (*fp*), true negatives (*tn*) and false negatives (*fn*) were measured. A true positive is when the model is given an input that represents a *drop*, which is classifies as a *drop*. True negatives are another type of spectrogram that is classified as not being a *drop*.

From these measurement, the F$_1$-score was calculated. This score is calculated as follows: $F_1score = \frac{2 \cdot tp}{2 \cdot tp + fn + fp}$. In addition to this, the *Matthews correlation coefficient* (MCC) was calculated. This coefficient is calculated as follows: $MCC = \frac{tp \cdot tn - fp \cdot fn}{\sqrt{(tp+fp)(tp+fn)(tn+fp)(tn+fn)}}$. The latter was calculated, as the F$_1$-score does not make use of the true negative class, which is a very prominent class in the evaluation set. The F$_1$-score still had to be calculated, however, to be able to compare the models to Yadati's models.

After this, I can fairly compare models and accurately describe the impact of certain training sets or model designs.

In table 1, you can see all the results of the experiments: the true positives, false positives, true negatives and false negatives. In addition to this, the table contains the calculated F1 scores and MCC of all combinations.

Table 1. Results on the evaluation set.

	SVM		
	Base	Aligned	Equalised
TP	110	114	114
FP	77	70	81
TN	260	267	256
FN	26	22	22
F$_1$-score	0.68	0.71	0.69
MCC	0.54	0.55	0.56
	CNN		
	Base	Aligned	Equalised
TP	124	122	127
FP	100	43	96
TN	237	294	241
FN	12	14	9
F$_1$-score	0.69	0.81	0.71
MCC	0.56	0.73	0.59

Please note that table 1 shows results for the evaluation set. This is an important note as there may be bias in the music I chose to be in the dataset.

5.2 Cross-validation

To measure the extent in which the models can be generalised, I also evaluate the proposed models on Yadati's dataset. Please note that the proposed models are still trained on my training sets, and Yadati's models are trained on their dataset. The results can be seen in table 2. Please note that Yadati trained his model with different window lengths: 3, 5, 7, 9, 11, 13 and 15 seconds long. Only 3, 9 and 15 seconds are shown in the table, as the window lengths in between follow the linear growth of the F$_1$-score that can be seen by just these three results. Notice that the true- and false positives and negatives were not recorded in this table, as this data is not directly known from Yadati's (2014) implementation.

Table 2. Results of the proposed models and Yadati's models on his music set.

	SVM		
	Base	Aligned	Equalised
F$_1$-score	0.65	0.70	0.65
	CNN		
	Base	Aligned	Equalised
F$_1$-score	0.68	0.73	0.65
	Yadati's model		
	3s window	9s window	15s window
F$_1$-score	0.61	0.66	0.71

5.3 Segmentation Accuracy

Lastly, it was measured how many tracks in the evaluation set do not contain a positive labeled spectrogram. This is an important measure to predict a full system's competence to identify a *drop* when only presented raw audio of *EDM* tracks. This is because the proposed classifier merely classifies segment boundaries and thus would not be able to pinpoint certain timestamps in the audio by itself. Therefore, if the chosen segmentation implementation does not find any boundary for a *drop* of an *EDM* track, the proposed models will be unable to identify a drop. The percentage of tracks on which segmentation could not detect an event at any *drop* was 34% in my dataset, while almost 48% in Yadati's (2014) music set.

6. DISCUSSION

Firstly, I would like to compare the two different model types: SVMs and CNNs. The results clearly suggest that CNNs are slightly better for this problem. However, when using specific kinds of data processing, CNNs excel and become much better for the problem. On both evaluation sets, a neural network trained with the aligned training set seems to outperform any other implementation. This makes me conclude that CNNs are clearly better at correctly labelling the *drop* of electronic music given spectrograms of segment boundaries.

Secondly, I would like to discuss the pre-processing techniques and which one fits the problem best. As we have already established, CNNs seem better suited for the problem. With this model design, it excelled on both evaluation sets when trained with the aligned training set. Its effectiveness seems to come from its ability to classify input as a false positive much less often than any other model-set combination. In addition to this, the aligned training set performed marginally better than the others when used to train the SVM model. This makes me conclude that the aligned training set is best suited for this problem. This can be explained by the fact that events in electronic music very often appear on the beat of a song (usually the start of a musical phrase, even). However, it can also be noted that the equalised training set seems to perform marginally better when evaluated on my dataset. When evaluating the models on Yadati's (2014) music set, the base training set seems to perform marginally better than the equalised training set.

Thirdly, I would like to discuss the final research question whether any model would be able to outperform Yadati's (2014) implementation. There are a number of important notes that should be made for this discussion:

1. The segmentation used in my and Yadati's systems are very similar and are both state of the art segmentation implementations: very few optimisations here are possible. However, since Yadati used state of the art segmentation from 2014, it could explain minor improvements of my system compared to his.
2. The approaches used were very similar and do not fundamentally differ: first segmentation, then labelling segment boundaries.
3. The proposed models were not trained on any music from Yadati's music set, but were tested on this. Therefore, this shows a form of generalisation for the proposed models, while this is not directly apparent to Yadati's implementation.
4. The segmentation finding an event for the drops worked significantly better or worse depending on the music set it was released on. This suggests that the two music sets are very different. (I can confirm this, my dataset seems to be biased towards house, while his dataset seems to be more balanced).
5. Yadati's models tried to label in more classes than the proposed models, and therefore might be less optimised just for finding the drop.
6. Yadati experimented with window sizes ranging from 3 seconds to 15 seconds. My window size is always 4 seconds.

Considering all these points, I would like to make some conclusions. It is apparent that the proposed models can be generalised, as they performed well on an evaluation set that was created by another author, even though they were trained with my training sets.

In addition to this, the CNN trained with the aligned dataset seems to outperform Yadati's best model with a window size of 15 seconds, even though my spectrograms have a window size of 4 seconds in all cases. The linear growth in performance of Yadati's implementation suggests larger window sizes tend to make models perform better. This could be because sometimes models need more audio context to determine what type of event is displayed in the input. For future work, someone could look at the impact of window size specifically with this model design and training data pre-processing. An important note here, is that it is known that spectrograms have to be of such a resolution that machine learning models can clearly identify micro-events (beats, chords, etc.). Therefore, increasing the window size is possible, but it must increase the input size and therefore the size of the network.

7. CONCLUSIONS

Firstly, between SVMs and CNNs, the latter can be concluded to be the best suitable for this problem. Both model designs showed a form of generalisation, but the CNN design excelled with the aligned training data.

Since the final system in which the model is evaluated has segment boundaries as its input, it is not surprising that both model designs excel with the aligned training set. The equalisation seems to have marginal impact on the models' performance in the best case scenario. Therefore is most likely not an investment that has to be made when pre-processing the training data.

Lastly, when trained on the proposed dataset and evaluated on Yadati's (2014) dataset, the models show generalisation and in some cases outperform Yadati's proposed models. The proposed models (with a 4 second window) always outperform Yadati's proposed model with a 3 second model. The proposed CNN trained with the aligned training set (still 4 second window) outperforms Yadati's best proposed model.

8. ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Nicola Strisciuglio for his great amount of help before and during the research. He greatly helped me to shape the research; provided me with more and better feedback than I could've hoped for; and kept me enthusiastic throughout the entire process.

I would like to thank Matthias te Braake for his help with curating the data that was needed for this research.

Lastly, I would like to thank Karthrik Yadati and his superiors and co-authors for swiftly responding to me and providing me their music set to evaluate the proposed models and compare the proposed models with theirs.

9. REFERENCES

[1] N. Instruments. Traktor pro 3 - professional 4-deck dj software.

[2] Keras. The python deep-learning api.

[3] R. Khulusia, J. Kusnich, C. Meinecke, C. Gillmann, J. Focht, and S. Jänicke. A survey on visualizations for musical data. *Computer Graphics Forum*, 0(2020):1–28, 2020.

[4] Librosa. Segmentation manual: Laplacian segmentation.

[5] Librosa. Segmentation manual: Temporal segmentation.

[6] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenbergk, and O. Nieto. librosa: Audio and music signal analysis in python. *Python In Science Conference*, 14(1):18–24, 2015.

[7] S. Rongshu, Z. Jingiing, J. Wei, and H. Yuexin. Segmentation of pop music based on histogram clustering. *CISP BMEI*, pages 1–5, 2018.

[8] Scikit-Learn. *User Manual Ch. 1.4.6: Kernel Functions.*

[9] J. Serrà, M. Müller, P. Grosche, and J. L. Arcos. Unsupervised music structure annotation by time series structure features and segment similarity. *IEEE Transactions on Multimedia*, 16(5):1229–1240, 2020.

[10] J. Steventon. *DJing for Dummies.* For Dummies, 2007.

[11] C. J. Tralie and B. McFee. Enhanced hierarchical music structure annotations via feature level similarity fusion. *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 201–205, 2019.

[12] Q. Wang, F. Su, and Y. Wang. Hierarchical attentive deep neural networks for semantic music annotation through multiple music representations. *International Journal of Multimedia Information Retrieval*, 9(1):3–16, 2020.

[13] K. Yadati, M. Larson, C. C. Liem, and A. Hanjalic. Detecting drops in electronic dance music: Content based approaches to a socially significant music event similarity. *International Society for Music Information Retrieval Conference*, 15(1):143–148, 2014.