PRUNING SUBSEQUENCE SEARCH WITH ATTENTION-BASED EMBEDDING

Colin Raffel and Daniel P. W. Ellis

LabROSA
Columbia University
New York, NY

ABSTRACT

Finding the most similar sequence to a query in a database of sequences can be prohibitively expensive when the database size is large. Typically, the scalability of different approaches are dependent on various "pruning" techniques, which use heuristics to avoid expensive comparisons against a large subset of the database. We present an approximate pruning technique which involves embedding sequences in a Euclidean space. Sequences are embedded using a convolutional network with a form of attention which integrates over time, which is trained using matching and non-matching pairs of sequences. By using fixed-length embeddings, our pruning method effectively runs in constant time, making it many orders of magnitude faster when compared to full Dynamic Time Warping-based matching. We demonstrate the effectiveness of our approach on a large-scale musical score-to-audio recording retrieval task.

***Index Terms*—** Sequence Retrieval, Attention-Based Models, Convolutional Networks, Dynamic Time Warping, Pruning

1. INTRODUCTION

Sequences[1] are a natural way of representing data in a wide range of fields, including multimedia, environmental sciences, natural language processing, and biology. The proliferation of sensors and decreasing cost of storage in recent years has resulted in the creation of very large databases of sequences. Given such a database, a fundamental task is to find the database entry which is most similar to a query. However, this task can be prohibitively expensive depending on both the method used to compare sequences and the size of the database.

An effective way to compare sequences is the Dynamic Time Warping (DTW) distance, first proposed in the context of speech utterances [1]. DTW first computes an optimal alignment of the two sequences of feature vectors being compared, and then reports their distance as the total distance among aligned feature vectors. The alignment step makes DTW much more robust to phase or offset errors than typical distance metrics, such as the Euclidean distance [2]. DTW also naturally extends to the setting where subsequence matching is allowed. Using dynamic programming, the optimal DTW alignment can be found in time quadratic in the length of the sequences. Despite this optimization, naive application of DTW to nearest-neighbor retrieval of the most similar sequence in a database can be prohibitively expensive. Other dynamic programming-based "edit distance" metrics are used when appropriate, and are similar in both their effectiveness and inefficiency.

[1]In this work, we refer to "signals" by the slightly more generic term "sequences", but the two can be used interchangeably.

To mitigate this issue, a variety of "pruning methods" have been proposed. The idea behind these techniques is to use heuristics to avoid computing the full DTW alignment for a large proportion of the database. [2] gives a broad overview of different pruning methods, and shows that their application makes exact retrieval feasible in extremely large (e.g. trillions of sequences) databases. However, these methods rely on various assumptions, including that the query sequence is a subsequence of its correct match and the length of the alignment path is known a priori.

An alternative approach is to replace the DTW search with a surrogate problem which is faster to solve. For example, [3] proposes a technique which maps sequences to a fixed-length embedding, which avoids DTW calculation when matching the query to the database. The embedding is constructed by pre-computing the DTW distance between each sequence in the database and a small collection of "reference" sequences, which are chosen to optimize retrieval accuracy. Then, to match a query sequence to the database, its DTW distance is computed against the same collection of reference sequences, and the resulting vector of DTW distances is matched against the database of embeddings using a standard Euclidean distance-based nearest-neighbor search. The resulting algorithm is approximate, but provided state-of-the-art rsults both in terms of accuracy and speed. This technique relies on the same assumptions as the pruning methods described in [2]. In addition, it presents a trade-off between efficiency and accuracy where more reference sequences generally improve accuracy but require more full DTW calculations to be made, which can be prohibitively expensive for very long or very high-dimensional sequences.

In [4], a more general approximate method is proposed where a neural network model is used to map and downsample sequences of feature vectors to shorter sequences of bit vectors. Computing the distance between bit vectors can be achieved by a single exclusive-or operation followed by a table lookup, so this approach vastly improves the efficiency of the local distance calculations involved in DTW. Furthermore, downsampling the sequences gives quadratic speed gains. In an experiment matching transcriptions of musical pieces to audio recordings in a large database, using this method allowed for 99% of the entries in the database to be pruned with high confidence. However, this approach still relies on DTW which is quadratic-time and therefore provides room for efficiency improvements.

Motivated by the above issues, we propose a learning-based system for producing sequence embeddings for approximate matching. Our approach is similar in spirit to [3], except that it is fully general, i.e. it does not rely on any assumptions about the alignment length or whether the query is a subsequence of its match. This is thanks to the fact that we utilize an attention-based neural network model which can adapt to any problem setting according to the training data provided. Our approach is also constant-time: By mapping sequences to

a fixed-length embedding, comparing a query to each database entry is a single Euclidean distance calculation, which does not become less efficient for longer or higher-dimensional sequences.

In the following section, we discuss embedding and attention-based neural network models, which we use to motivate our proposed model in Section 3. In Section 4, we evaluate the effectiveness of our proposed model on the task of matching musical transcriptions to audio recordings of their corresponding songs in a large database. Finally, we discuss results and possibilities for improvement in Section 5.

2. EMBEDDING AND ATTENTION

The idea of "embedding" is an attractive one: Given data which is provided in a representation which is either inefficient or does not readily reveal factors of interest, can we produce a mapping to fixed-length vectors for which our data has the desired properties? Because many simple machine learning methods are very effective given data where Euclidean distance corresponds to some notion of similarity, embedding can make data more amenable to learning and comparison.

As a concrete example, in [5], a neural network architecture is used to embed images of faces such that images of the same person have small pairwise distances in the embedded space images of different people have large distances. The authors were able to achieve state-of-the-art results on the "labeled faces the wild" dataset by simply thresholding the resulting distances to denote "same person" or "different person". In addition, the embedded representation produced qualitatively useful results when performing unsupervised clustering using k-means. Another highly effective application of embedding is word2vec, which is a family of models for mapping words to a Euclidean space which has desireable properties [6]. For example, the difference between "China" and "Beijing" in the embedded space is roughly equivalent to the difference between "Rome" and "Italy".

The idea of embedding has also been applied to sequences. Notably, in [7], a recurrent network was trained to map a sequence of words in a source language to a fixed-length vector, which was then used to generate a sequence of words in a target language, resulting in a quality translation of the original sentence. The resulting source-sentence embeddings encoded high-level linguistic characteristics, such as invariance to word order in sentences with the same meaning. A benefit of using recurrent networks is their ability to summarize an entire sequence of arbitrary length as a fixed-length vector, which brings the possibility of learning embeddings to sequences.

One issue with using recurrent networks is their difficulty in modeling long-term dependencies [8]. In the embedding setting, this may result in the end of the sequence having more of an effect on the embedding than the beginning. A recently proposed technique for mitigating this issue is "attention" [9]. Conceptually, an attention-based model includes a mechanism which learns to assign a weighing at each sequence step based on the current and all previous feature vectors. When used with recurrent networks, the addition of attention has proven effective in a wide variety of domains including speech recognition, machine translation, and image captioning [10].

At a high level, attention-based models are similar to the common technique of summarizing a sequence of feature vectors by the mean, variance, and occasionally covariance of feature dimensions. This simple approach has seen frequent application in the domain of music information retrieval, where it has proven to be surprisingly effective at genre classification [11, 12], instrument classification [13], artist recognition [14], and similarity rating prediction [15]. Attention-based models can be seen as a generalization of this technique where both the feature representation and a weighted average are both optimized and data-adaptive.

3. FEED-FORWARD ATTENTION

While previous work on attention-based models has focused on recurrent networks, in the present work we will use feed-forward networks for the following reasons: First, Recurrent networks have difficulty modelling very long-term dependencies [8]. In the experiment described in Section 4, sequences may have thousands of time steps, which is prohibitively long even for sophisticated models such as long short-term memory networks [16]. Second, feed-forward networks are much more efficient to train and evaluate than recurrent networks because their operations can be completely parallelized, whereas recurrent networks must evaluate each time step sequentially. Finally, attention provides its own form of temporal modelling due to the fact that it integrates over the entire sequence. We therefore propose a "feed-forward" variant of attention, which computes a weighting for each sequence step independent of all other steps.

Our simplified variant of attention can be formulated as follows: Given a matrix which comprises a sequence of M D-dimensional vectors $X \in \mathbb{R}^{M \times D}$, we compute weights $\alpha \in \mathbb{R}^M$ using the following transformation:

$$\alpha = \mathrm{softmax}(\sigma(Xw + b)) \tag{1}$$

where

$$\mathrm{softmax}(x)_j = \frac{e^{x_j}}{\sum_{i=1}^{M} e^{x_i}}$$

and σ is some nonlinear function (throughout this work, we will use $\sigma = \tanh$). $w \in \mathbb{R}^D$ and $b \in \mathbb{R}$ are parameters of the transformation, which can be learned as part of a larger feed-forward network model. After computing α, the weighted average of the vectors in X is computed by

$$\hat{X} = \sum_{i=1}^{M} \alpha_i X_{i,:}$$

Conceptually, this attention mechanism can be thought of as using w, b, and σ to compute scalar "importance" values for each vector in X, which are then normalized using the softmax function and used to compute a weighted mean $\hat{X}$ over all vectors in X.

The main difference between this formulation and the one proposed in [9, 10] is that here we are only producing a single weighting α for the entire sequence rather than a different α at each time step. This is primarily because our goal in embedding is to produce a single vector for the entire sequence whereas previous applications of attention have mainly focused on sequence-to-sequence transduction. As a result, Equation 1 does not contain any terms for a previous attention vector or a previous hidden state, because in the present contexts these quantities do not exist. The main disadvantage to using attention in this way is that it ignores temporal order by computing an average over the entire sequence. However, motivated by the surprising effectiveness of the unweighted/non-adaptive Gaussian results discussed in Section 2, we submit that this approach will be effective and verify its utility in the following section.

4. PRUNING EXPERIMENT

Recall that the goal of this work is to utilize embeddings to speed up large-scale sequence similarity searches by avoiding comparisons

to most sequences in a database. Towards this end, we will utilize embeddings as follows: First, given a training set of matching sequences, we will train a feed-forward network with attention to embed each sequence in a Euclidean space where similar sequences have a small distance and dissimilar ones have a large distance. Then, we will utilize this network to pre-compute embeddings for all sequences in a database. In order to match a query sequence, we will compute the distance between its embedding and all pre-computed embeddings from the database. Given a test set of correct matches which are known a priori, the effectiveness of this approach can be measured by determining how often the correct match appears in the N entries with the N smallest embedded distances for varying values of N.

We will evaluate this approach on the problem proposed in [4]. This work seeks to efficiently match a transcription of a piece of music to a recording of the same piece in a large database of recordings. The resulting problem boils down to a large-scale subsequence matching problem, where the sequences are particularly long (up to thousands of time steps), the feature dimensionality is relatively high, and the correct match may be a subsequence of the query and vice versa. These properties make the traditional DTW pruning methods outlined in Section 1 ill-suited and provide a setting to evaluate completely general-purpose methods like the one proposed in this work. We will outline the details of the experiment below; for a complete discussion, we refer the reader to [4].

The training set consists of a collection of MIDI files which have been matched to audio files. MIDI files can be thought of as transcriptions, and can be synthesized in order to obtain an audio recording which is qualitatively similar to the song they are a transcription of. In [4], the MIDI and audio pairs are also aligned in time to one another using DTW before being used as training data; the system proposed in this work doesn't require sequences which are aligned in time so we skipped this step. The resulting training set contains 5,536 sequence pairs.

As a representation, we use log-magnitude constant-Q spectrograms [17] of the audio files and MIDI syntheses. Each constant-Q spectra spanned four octaves, using twelve bins per octave starting from MIDI note C3 (65.4 Hz). Each frequency bin in all feature sequences was z-scored (standardized) by its mean and standard deviation over the training set. In [4], using shorter sequences was beneficial so constant-Q spectra were aggregated over estimated beat locations. The present approach is meant to be able to cope with sequences which may be oversampled in time, so we did not beat-aggregate feature vectors and instead used spectra computed every 46 milliseconds. All feature analysis and processing was accomplished with `librosa` [18, 19]

As a model, we also mimic [4] and use a siamese architecture consisting of two parallel convolutional networks followed by two parallel fully-connected networks. The use of a convolutional "front end" allows our model to learn exploit both time and frequency invariances in the constant-Q spectra while reducing the number of model parameters. The main difference in models is the addition of the attention mechanism proposed in Section 3, which we place between the convolutional network and the fully-connected network. In this way, the convolutional network can be seen as transforming a "raw" input sequence to a sequence of vectors in a learned representation, which are then aggregated using the attention mechanism and mapped to the embedded space using the fully-connected network. We used an embedding space dimensionality of 128.

To train the networks, we pass pairs of "matching" and "non-matching" sequences and utilize a loss function which encourages matching pairs to have similar embeddings and non-matching pairs

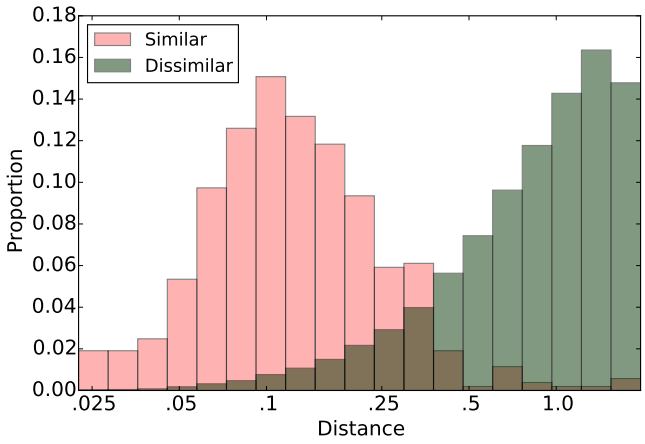

Fig. 1. Histograms of embedded distances for positive and negative example pairs in the validation set for our best-performing embedding network. Note that the x-axis is log-scaled.

to have dissimilar ones. We utilize the same loss function as in [4]: Given a pair of positive sequences X_p, Y_p and a set of negative sequences X_n, Y_n, we attempt to minimize

$$\mathcal{L} = \|f(X_p) - g(Y_p)\|_2^2 - \alpha \max(0, m - \|f(X_n) - g(Y_n)\|_2)^2$$

where f and g are the functions of the two networks being trained, α is a parameter controlling the importance of separating dissimilar pairs, and m is the target separation for dissimilar pairs. To construct negative pair examples, we simply randomly choose non-matched MIDI/audio sequence pairs from the training set.

As in [4] we utilize rectified linear units throughout the network, except on the final output layer which used $\tanh$ units. All weights were initialized using He et. al.'s method [20] and all biases were initialized to zero. The networks were optimized with respect to $\mathcal{L}$ using Nesterov's accelerated gradient [21] with mini-batches of 20 similar and dissimilar pairs. All sequences were either randomly cropped or padded to the median sequence length in each modality. Every 100 minibatches, we computed the mean loss $\mathcal{L}$ on a held-out validation set consisting of 10% of the training set; when it was less than .99 times the previous lowest validation loss, we trained for 500 more minibatches.

To choose hyperparameter settings, we used the Bayesian optimization software package `spearmint` [22]. We optimized the number of convolutional and pooling layers, the number of fully-connected layers, the learning rate and momentum, and the loss parameters α and m. We also experimented with using RMSProp [23] and adam [24] for optimization and dropout [25] in the fully-connected layers for regularization, but did not find them to be beneficial. As an objective, we computed the Bhattacharrya coefficient [26] between the distributions of matching and non-matching pairs in the validation set. The best-performing network configuration achieved a Bhattacharrya coefficient of .471 and used a single convolutional layer with 16 filters which were 5 time steps wide by 12 semitones tall, one 2x2 max-pooling layer, three fully connected layers (including the final output layer) with 2048 units each, a learning rate of 10^{-4}, a momentum of 0.997, $\alpha = 0.935$ and $m = 1$. The validation set distance distributions produced by this network after training can be seen in Figure 1.

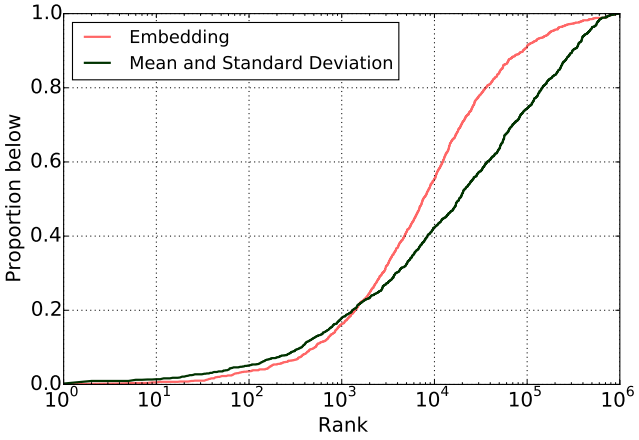

Fig. 2. Proportion of MIDI files in the test set whose correct match appeared below a given rank, based on embedding distances.

5. RESULTS

To evaluate our system's performance on the task of pruning large-scale subsequence search, we used it to match a held-out test set of 1,282 MIDI files to entries in the Million Song Dataset (MSD) [27]. Our test set was the same used in [4], for which the correct match in the MSD is known for each MIDI file. We first computed embeddings of constant-Q spectrograms for all of the short (30 to 60 second) preview recordings of each entry in the MSD provided by 7digital [28]. Then, we computed a constant-Q spectrogram and its embedding for each MIDI file in the test set. To determine performance, we computed the Euclidean distance between the embedding of each MIDI file and the embeddings of every entry in the MSD and determined how many entries had a distance that was smaller than the correct match. As a baseline, we performed the same evaluation except using the concatenated mean and standard deviation of the feature dimensions of constant-Q spectrograms as the embedding. The results can be seen in Figure 2.

To summarize these results, using the proposed method the correct match appeared in the top 100,000 matches 91.3% of the time using embedding distance and only 74.4% of the time using the mean and standard deviation of spectrograms. As noted in [4], matching a single MIDI file to the MSD using dynamic time warping distance would take about three hours; the proposed approach takes about 360 milliseconds per file using a non-optimized Python implementation. Another way of stating our result, then, is that we can discard 90% (900,000 entries) of the MSD with 91.3% confidence using a pruning method which is about 30,000 times faster than the standard DTW-based matching approach. Compared to the approach proposed in [4] which can prune 99% of the MSD with 95.9% confidence, it is substantially less accurate, but it is also over 300 times faster. In addition, our approach is fully general, and was able to achieve good performance on a task that would not be possible using the methods proposed in [2] or [3].

While our proposed approach has proven effective, there is room for improvement. The model may be able to further differentiate between similar and dissimilar sequences if the dimensionality of the embedded space was increased, at the cost of a linear increase in runtime. Alternatively, choosing negative example pairs more carefully (e.g. using the "harmonic embedding" approach of [5]) may further improve discrimination. Finally, evaluating this approach on more traditional sequence retrieval tasks which allow more stringent assumptions to be made about the correct match would facilitate comparison to existing pruning techniques. For researchers interested in utilizing and reproducing our results, code is available online.[2]

6. REFERENCES

[1] Hiroaki Sakoe and Seibi Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 26, no. 1, pp. 43–49, 1978.

[2] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 262–270.

[3] Panagiotis Papapetrou, Vassilis Athitsos, Michalis Potamias, George Kollios, and Dimitrios Gunopulos, "Embedding-based subsequence matching in time-series databases," *ACM Transactions on Database Systems (TODS)*, vol. 36, no. 3, pp. 17, 2011.

[4] Colin Raffel and Daniel P. W. Ellis, "Large-scale content-based matching of MIDI and audio files," in *Proceedings of the 16th International Society for Music Information Retrieval Conference*, 2015.

[5] Florian Schroff, Dmitry Kalenichenko, and James Philbin, "Facenet: A unified embedding for face recognition and clustering," *arXiv preprint arXiv:1503.03832*, 2015.

[6] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[7] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[8] Yoshua Bengio, Patrice Simard, and Paolo Frasconi, "Learning long-term dependencies with gradient descent is difficult," *Neural Networks, IEEE Transactions on*, vol. 5, no. 2, pp. 157–166, 1994.

[9] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[10] Kyunghyun Cho, Aaron C. Courville, and Yoshua Bengio, "Describing multimedia content using attention-based encoder-decoder networks," *arXiv preprint arXiv:1507.01053*, 2015.

[11] James Bergstra, Norman Casagrande, Dumitru Erhan, Douglas Eck, and Balázs Kégl, "Aggregate features and adaboost for music classification," *Machine learning*, vol. 65, no. 2-3, pp. 473–484, 2006.

[12] George Tzanetakis and Perry Cook, "Musical genre classification of audio signals," *Speech and Audio Processing, IEEE transactions on*, vol. 10, no. 5, pp. 293–302, 2002.

[2] `http://github.com/craffel/sequence-embedding`

[13] Jeremiah D Deng, Christian Simmermacher, and Stephen Cranefield, "A study on feature analysis for musical instrument classification," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 38, no. 2, pp. 429–438, 2008.

[14] Michael I. Mandel and Daniel P. W. Ellis, "Song-level features and support vector machines for music classification," in *Proceedings of the 6th International Conference on Music Information Retrieval*. Queen Mary, University of London, 2005, pp. 594–599.

[15] Peter Foster, Matthias Mauch, and Sam Dixon, "Sequential complexity as a descriptor for musical similarity," *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, vol. 22, no. 12, pp. 1965–1977, 2014.

[16] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[17] Judith C Brown, "Calculation of a constant q spectral transform," *The Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1991.

[18] Brian McFee, Matt McVicar, Colin Raffel, Dawen Liang, Oriol Nieto, Josh Moore, Dan Ellis, Douglas Repetto, Petr Viktorin, Joo Felipe Santos, and et al., "`librosa`: v0.4.0," `https://github.com/bmcfee/librosa`, 2015.

[19] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th Python in Science Conference*, 2015.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *arXiv preprint arXiv:1502.01852*, 2015.

[21] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th international conference on machine learning (ICML-13)*, 2013, pp. 1139–1147.

[22] Jasper Snoek, Hugo Larochelle, and Ryan P Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[23] Tijmen Tieleman and Geoffrey Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, vol. 4, 2012.

[24] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[25] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[26] Anil Kumar Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distributions," *Bulletin of the Calcutta Mathematical Society*, vol. 35, pp. 99–109, 1943.

[27] Thierry Bertin-Mahieux, Daniel P. W. Ellis, Brian Whitman, and Paul Lamere, "The million song dataset," in *Proceedings of the 12th International Society for Music Information Retrieval Conference*, 2011, pp. 591–596.

[28] Alexander Schindler, Rudolf Mayer, and Andreas Rauber, "Facilitating comprehensive benchmarking experiments on the million song dataset," in *Proceedings of the 13th International Society for Music Information Retrieval Conference*, 2012, pp. 469–474.