Learning Content Similarity for Music Recommendation

Brian McFee, *Student Member, IEEE*, Luke Barrington, and Gert Lanckriet, *Member, IEEE*

***Abstract*—Many tasks in music information retrieval, such as recommendation, and playlist generation for online radio, fall naturally into the *query-by-example* setting, wherein a user queries the system by providing a song, and the system responds with a list of relevant or similar song recommendations. Such applications ultimately depend on the notion of *similarity* between items to produce high-quality results. Current state-of-the-art systems employ *collaborative filter* methods to represent musical items, effectively comparing items in terms of their constituent users. While collaborative filter techniques perform well when historical data is available for each item, their reliance on historical data impedes performance on novel or unpopular items. To combat this problem, practitioners rely on content-based similarity, which naturally extends to novel items, but is typically outperformed by collaborative filter methods. In this paper, we propose a method for optimizing content-based similarity by learning from a sample of collaborative filter data. The optimized content-based similarity metric can then be applied to answer queries on novel and unpopular items, while still maintaining high recommendation accuracy. The proposed system yields accurate and efficient representations of audio content, and experimental results show significant improvements in accuracy over competing content-based recommendation techniques.**

***Index Terms*—Audio retrieval and recommendation, collaborative filters (CFs), music information retrieval, query-by-example, structured prediction.**

I. Introduction

AN effective notion of similarity forms the basis of many applications involving multimedia data. For example, an online music store can benefit greatly from the development of an accurate method for automatically assessing similarity between two songs, which can in turn facilitate high-quality recommendations to a user by finding songs which are similar to her previous purchases or preferences. More generally, high-quality similarity can benefit any *query-by-example* recommendation system, wherein a user presents an example of an item that she likes, and the system responds with, e.g., a ranked list of recommendations.

The most successful approaches to a wide variety of recommendation tasks—including not just music, but books, movies, etc.—use *collaborative filters* (CF). Systems based on collaborative filters exploit the "wisdom of crowds" to infer similarities between items, and recommend new items to users by representing and comparing these items in terms of the people who use them [1]. Within the domain of music information retrieval, recent studies have shown that CF systems consistently outperform alternative methods for playlist generation [2] and semantic annotation [3]. However, collaborative filters suffer from the dreaded "cold start" problem: a new item cannot be recommended until it has been purchased, and it is less likely to be purchased if it is never recommended. Thus, only a tiny fraction of songs may be recommended, making it difficult for users to explore and discover new music [4].

The cold-start problem has motivated researchers to improve *content-based* recommendation systems. Content-based systems operate on music representations that are extracted automatically from the audio content, eliminating the need for human feedback and annotation when computing similarity. While this approach naturally extends to any item regardless of popularity, the construction of features and definition of *similarity* in these systems are frequently ad-hoc and not explicitly optimized for the specific task.

In this paper, we propose a method for optimizing content-based audio similarity by learning from a sample of collaborative filter data. Based on this optimized similarity measure, recommendations can then be made where no collaborative filter data is available. The proposed method treats similarity learning as an information retrieval problem, where similarity is learned to optimize the ranked list of results in response to a query example (Fig. 1). Optimizing similarity for ranking requires more sophisticated machinery than, e.g., genre classification for semantic search. However, the information retrieval approach offers a few key advantages, which we argue are crucial for realistic music applications. First, there are no assumptions of transitivity or symmetry in the proposed method. This allows, for example, that "The Beatles" may be considered a relevant result for "Oasis," but not vice versa. Second, CF data can be collected *passively* from users by mining their listening histories, thereby directly capturing their listening habits. Finally, optimizing similarity for ranking directly attacks the main quantity of interest: the ordered list of retrieved items, rather than coarse abstractions of similarity, such as genre agreement.

Manuscript received April 22, 2011; revised September 01, 2011; accepted April 24, 2012. Date of publication June 06, 2012; date of current version August 09, 2012. This work was supported in part by Qualcomm, Inc, eHarmony, Inc., Yahoo! Inc., the National Science Foundation (NSF) under Grants CCF-0830535 and IIS-1054960, an the UCSD FWGrid Project, NSF Research Infrastructure Grant Number EIA-0303622. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Bryan Pardo.

B. McFee and G. Lanckriet are with the Department of Computer Science and Engineering, University of California at San Diego, La Jolla, CA, 92093 USA (e-mail: bmcfee@cs.ucsd.edu; gert@ece.ucsd.edu).

L. Barrington is with the Tomnod, Inc., San Diego, CA 92126 USA (e-mail: luke@tomnod.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASL.2012.2199109

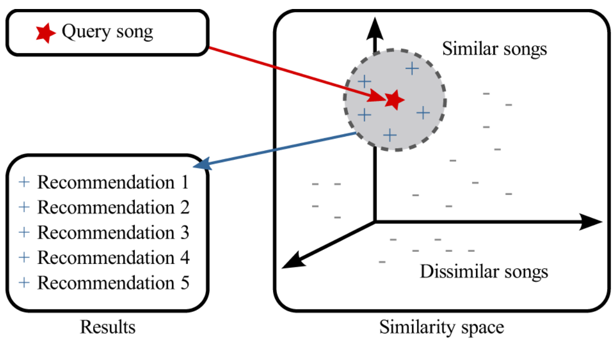

Fig. 1. Query-by-example recommendation engines allow a user to search for new items by providing an example item. Recommendations are formed by computing the most similar items to the query item from a database of potential recommendations.

A. Related Work

Early studies of musical similarity followed the general strategy of first devising a model of audio content (e.g., spectral clusters [5], Gaussian mixture models [6], or latent topic assignments [7]), applying some reasonable distance function (e.g., earth-mover's distance or Kullback–Leibler divergence), and then evaluating the proposed similarity model against some source of ground truth. Logan and Salomon [5] and Aucouturier and Pachet [6] evaluated against three notions of similarity between songs: same artist, same genre, and human survey data. Artist or genre agreement entail strongly binary notions of similarity, which due to symmetry and transitivity may be unrealistically coarse in practice. Survey data can encode subtle relationships between items, for example, triplets of the form *"A is more similar to B than to C"* [6], [8], [9]. However, the expressive power of human survey data comes at a cost: while artist or genre meta-data is relatively inexpensive to collect for a set of songs, similarity survey data may require human feedback on a quadratic (for pairwise ratings) or cubic (for triplets) number of comparisons between songs, and is thus impractical for large data sets.

Later work in musical similarity approaches the problem in the context of supervised learning: given a set of training items (songs), and some knowledge of similarity across those items, the goal is to *learn* a similarity (distance) function that can predict pairwise similarity. Slaney *et al.* [10] derive similarity from web-page co-occurrence, and evaluate several supervised and unsupervised algorithms for learning distance metrics. McFee and Lanckriet [11] develop a metric learning algorithm for triplet comparisons as described above. Our proposed method follows in this line of work, but is designed to optimize structured ranking loss (not just binary or triplet predictions), and uses a collaborative filter as the source of ground truth.

The idea to learn similarity from a collaborative filter follows from a series of positive results in music applications. Slaney and White [12] demonstrate that an item-similarity metric derived from rating data matches human perception of similarity better than a content-based method. Similarly, it has been demonstrated that when combined with metric learning, collaborative filter similarity can be as effective as semantic tags for predicting survey data [11]. Kim *et al.* [3] demonstrated that collaborative filter similarity vastly outperforms content-based methods for predicting semantic tags. Barrington *et al.* [2] conducted a user survey, and concluded that the iTunes Genius playlist algorithm (which is at least partially based on collaborative filters[1]) produces playlists of equal or higher quality than competing methods based on acoustic content or meta-data.

Finally, there has been some previous work addressing the cold-start problem of collaborative filters for music recommendation by integrating audio content. Yoshii *et al.* [13] formulate a joint probabilistic model of both audio content and collaborative filter data in order to predict user ratings of songs (using either or both representations), whereas our goal here is to use audio data to predict the similarities derived from a collaborative filter. Our problem setting is most similar to that of Stenzel and Kamps [14], wherein a CF matrix was derived from playlist data, clustered into latent "pseudo-genres," and classifiers were trained to predict the cluster membership of songs from audio data. Our proposed setting differs in that we derive similarity at the user level (not playlist level), and automatically learn the content-based song similarity that directly optimizes the primary quantity of interest in an information retrieval system: the quality of the rankings it induces.

B. Our Contributions

Our primary contribution in this work is a framework for improving content-based audio similarity by learning from a sample of collaborative filter data. Toward this end, we first develop a method for deriving item similarity from a sample of collaborative filter data. We then use the sample similarity to train an optimal distance metric over audio descriptors. More precisely, a distance metric is optimized to produce high-quality rankings of the training sample in a query-by-example setting. The resulting distance metric can then be applied to previously unseen data for which collaborative filter data is unavailable. Experimental results verify that the proposed methods significantly outperform competing methods for content-based music retrieval.

This paper expands on results described in a previous conference publication [15]. This paper investigates a broader class of audio representations, provides a more thorough evaluation of each component of the framework, and includes thorough comparisons to alternative methods. Notably, in comparison to the previous work, the audio representations described here significantly improve performance relative to semantic auto-tag representations.

C. Preliminaries

For a d-dimensional vector $u \in \mathbb{R}^d$ let $u[i]$ denote its ith coordinate; similarly, for a matrix A, let $A[ij]$ denote its ith row and jth column entry. A square, symmetric matrix $A \in \mathbb{R}^{d\times d}$ is *positive semi-definite* (PSD, denoted $A \succeq 0$) if each of its eigenvalues is nonnegative. For two matrices A, B of compatible dimension, the Frobenius inner product is defined as

$$\langle A, B\rangle_{\mathrm{F}} = \mathrm{tr}(A^{\mathsf{T}}B) = \sum_{i,j} A[ij]B[ij].$$

[1]http://www.apple.com/pr/library/2008/09/09itunes.html

Finally, let $\mathbb{1}[x]$ denote the binary indicator function of the event x.

II. Learning Similarity

The main focus of this work is the following information retrieval problem: given a *query* song q, return a ranked list from a database $\mathcal{X}$ of n songs ordered by descending similarity to q. In general, the query may be previously unseen to the system, but $\mathcal{X}$ will remain fixed across all queries. We will assume that each song is represented by a vector in $\mathbb{R}^d$, and similarity is computed by Euclidean distance. Thus, for any query q, a natural ordering of $x \in \mathcal{X}$ is generated by sorting according to increasing distance from q: $\|q - x\|$.

Given some side information describing the similarity relationships between items of $\mathcal{X}$, distance-based ranking can be improved by applying a *metric learning* algorithm. Rather than rely on native Euclidean distance, the learning algorithm produces a PSD matrix $W \in \mathbb{R}^{d\times d}$ which characterizes an optimized distance

$$\|q - x\|_W = \sqrt{(q-x)^\mathsf{T} W (q-x)}. \tag{1}$$

In order to learn W, we will apply the metric learning to rank (MLR) [16] algorithm (Section II-B). At a high level, MLR optimizes the distance metric W on $\mathcal{X}$, i.e., so that W generates optimal rankings of songs in $\mathcal{X}$ when using each song in $\mathcal{X}$ as a query. To apply the algorithm, we must provide a set of similar songs $x \in \mathcal{X}$ for each *training* query $q \in \mathcal{X}$. This is achieved by leveraging the side information that is available for items in $\mathcal{X}$. More specifically, we will derive a notion of similarity from collaborative filter data on $\mathcal{X}$. So, the proposed approach optimizes content-based audio similarity by learning from a sample of collaborative filter data.

A. Collaborative Filters

The term *collaborative filter* (CF) is generally used to denote to a wide variety of techniques for modeling the interactions between a set of items and a set of users [1], [17]. Often, these interactions are modeled as a (typically sparse) matrix F where rows represent the users, and columns represent the items. The entry $F[ij]$ encodes the interaction between user i and item j.

The majority of work in the CF literature deals with F derived from explicit user feedback, e.g., 5-star ratings [12], [13]. While rating data can provide highly accurate representations of user-item affinity, it also has drawbacks, especially in the domain of music. First, explicit ratings require active participation on behalf of users. This may be acceptable for long-form content such as films, in which the time required for a user to rate an item is miniscule relative to the time required to consume it. However, for short-form content (e.g., songs), it seems unrealistic to expect a user to rate even a fraction of the items consumed. Second, the scale of rating data is often arbitrary, skewed toward the extremes (e.g., 1- and 5-star ratings), and may require careful calibration to use effectively [12].

Alternatively, CF data can also be derived from *implicit* feedback. While somewhat noisier on a per-user basis than explicit feedback, implicit feedback can be derived in much higher volumes by simply counting how often a user interacts with an item (e.g., listens to an artist) [18], [19]. Implicit feedback differs from rating data, in that it is positive and unbounded, and it does not facilitate explicit negative feedback. As suggested by Hu *et al.* [19], binarizing an implicit feedback matrix by thresholding can provide an effective mechanism to infer positive associations.

In a binary CF matrix F, each column $F[\cdot j]$ can be interpreted as a *bag-of-users* representation of item j. Of central interest in this paper is the similarity between items (i.e., columns of F). We define the similarity between two items $i,\ j$ as the Jaccard index [20] of their user sets:

$$S(i,j) = \frac{|F[\cdot i] \cap F[\cdot j]|}{|F[\cdot i] \cup F[\cdot j]|} = \frac{F[\cdot i]^\mathsf{T} F[\cdot j]}{|F[\cdot i]| + |F[\cdot j]| - F[\cdot i]^\mathsf{T} F[\cdot j]} \tag{2}$$

which counts the number of users shared between items i and j, and normalizes by the total number of users of i and j combined.

Equation (2) defines a quantitative metric of similarity between two items. However, for information retrieval applications, we are primarily interested in the most similar (relevant) items for any query. We therefore define the *relevant* set $\mathcal{X}_q^+$ for any item q as the top k most similar items according to (2), i.e., those items which a user of the system would be shown first. Although binarizing similarity in this way does simplify the notion of relevance, it still provides a flexible language for encoding relationships between items. Note that after thresholding, transitivity and symmetry are not enforced, so it is possible, e.g., for *The Beatles* to be relevant for *Oasis* but not vice versa. Consequently, we will need a learning algorithm which can support such flexible encodings of relevance.

B. Metric Learning to Rank

Any query-by-example retrieval system must have at its core a mechanism for assessing the similarity of a query to a each item in a known database. Intuitively, the overall system should yield better results if the underlying similarity mechanism is optimized according to the chosen task. In classification tasks, for example, this general idea has led to a family of algorithms collectively known as *metric learning*, in which a feature space is optimized (typically by a linear transformation) to improve performance of nearest-neighbor classification [21]–[23]. While metric learning algorithms have been demonstrated to yield substantial improvements in classification performance, nearly all of them are fundamentally limited to classification, and do not readily generalize to asymmetric and non-transitive notions of similarity or relevance. Moreover, the objective functions optimized by most metric learning algorithms do not clearly relate to ranking performance, which is of fundamental interest in information retrieval applications.

Rankings, being inherently combinatorial objects, can be notoriously difficult to optimize. Performance measures of rankings, e.g., area under the ROC curve (AUC) [24], are typically non-differentiable, discontinuous functions of the underlying parameters, so standard numerical optimization techniques cannot be directly applied. However, in recent years, algorithms based on the structural SVM [25] have been developed which can efficiently optimize a variety of ranking

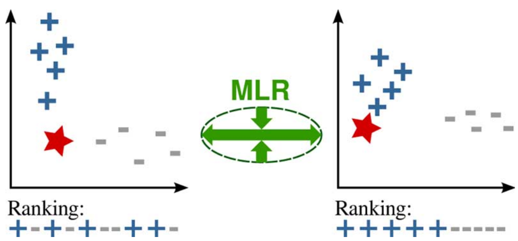

Fig. 2. Left: a query point $\star$ and its relevant $(+)$ and irrelevant $(-)$ results; ranking by distance from $\star$ results in poor retrieval performance. Right: after learning an optimal distance metric with MLR, relevant results are ranked higher than irrelevant results.

performance measures [26]–[28]. While these algorithms support general notions of relevance, they do not directly exploit the structure of query-by-example retrieval problems.

The metric learning to rank (MLR) algorithm combines these two approaches of metric learning and structural SVM, and is designed specifically for the query-by-example setting [16]. MLR learns a positive semi-definite matrix W such that rankings induced by learned distances (1) are optimized according to a ranking loss measure, e.g., AUC, mean reciprocal rank (MRR) [29], or normalized discounted cumulative gain (NDCG) [30]. In this setting, "relevant" results should lie close in space to the query q, and "irrelevant" results should be pushed far away.

For a query song q, the database $\mathcal{X}$ is ordered by sorting each $x \in \mathcal{X}$ according to increasing distance from q under the metric defined by $W \in \mathbb{R}^{d\times d}$ (see Fig. 2). The metric W is learned by solving a constrained convex optimization problem such that, for each input training query q, a higher score is assigned to a correct ranking y_q than to any other possible output ranking y in the set $\mathcal{Y}$ of all rankings (permutations of $\mathcal{X}$):

$$\forall y \in \mathcal{Y} : \langle W, \psi(q, y_q)\rangle_{\mathrm{F}} \geq \langle W, \psi(q, y)\rangle_{\mathrm{F}} + \Delta(y_q, y) - \xi_q. \tag{3}$$

Here, the "score" for an input–output (query-ranking) pair (q, y) is computed by the Frobenius inner product $\langle W, \psi(q, y)\rangle_{\mathrm{F}}$.[2] The function $\psi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^{d\times d}$ produces a matrix-valued representation of the input–output pair (q, y) (described below). $\Delta(y_q, y)$ computes the loss (e.g., decrease in AUC) incurred by predicting y instead of y_q for the query q, essentially playing the role of the "margin" between rankings y_q and y. Intuitively, the score for a correct output ranking y_q should exceed the score for any other ranking y by at least the loss $\Delta(y_q, y)$. (A correct ranking is any one which places all relevant results $\mathcal{X}_q^+$ before all irrelevant results $\mathcal{X}_q^-$.) As in support vector machines, a slack variable $\xi_q \geq 0$ is introduced for each query to allow for margin violations during training.

Having defined the margin constraints (3), what remains to be specified, to learn W, is the feature map ψ and the objective function of the optimization. To define the feature map ψ, we first observe that the margin constraints indicate that, for a query q, the predicted ranking y should be that which maximizes the score $\langle W, \psi(q, y)\rangle_{\mathrm{F}}$. Consequently, the (matrix-valued) feature map $\psi(q, y)$ must be chosen so that the score maximization coincides with the distance-ranking induced by W, which is, after all, the prediction rule we propose to use in practice, for query-by-example recommendation (1). To accomplish this, MLR encodes query-ranking pairs (q, y) by the *partial order* feature [26]:

$$\psi(q, y) = \sum_{i\in\mathcal{X}_q^+} \sum_{j\in\mathcal{X}_q^-} y_{ij} \frac{(\phi(q, x_i) - \phi(q, x_j))}{|\mathcal{X}_q^+| \cdot |\mathcal{X}_q^-|} \tag{4}$$

where $\mathcal{X}_q^+$ ($\mathcal{X}_q^-$) is the set of relevant (irrelevant) songs for q, the ranking y is encoded by

$$y_{ij} = \begin{cases} +1, & i \text{ precedes } j \text{ in } y \\ -1, & i \text{ succeeds } j \text{ in } y \end{cases}$$

and $\phi(q, x_i)$ is an auxiliary (matrix-valued) feature map that encodes the relationship between the query $q \in \mathbb{R}^d$ and an individual item $x_i \in \mathbb{R}^d$. Intuitively, $\psi(q, y)$ decomposes the ranking y into pairs $(i, j) \in \mathcal{X}_q^+ \times \mathcal{X}_q^-$, and computes a signed average of pairwise differences $\phi(q, x_i) - \phi(q, x_j)$.

If y places i before j (i.e., correctly orders i and j), the difference $\phi(q, x_i) - \phi(q, x_j)$ is added to $\psi(q, y)$, and otherwise it is subtracted. Note that under this definition of ψ, any two correct rankings y_q, y_q' have the same feature representation: $\psi(q, y_q) = \psi(q, y_q')$. It therefore suffices to only encode a single correct ranking y_q for each query q to construct margin constraints (3) during optimization.

Since ψ is linear in ϕ, the score also decomposes into a signed average across pairs:

$$\langle W, \psi(q, y)\rangle_{\mathrm{F}} = \sum_{i\in\mathcal{X}_q^+} \sum_{j\in\mathcal{X}_q^-} y_{ij} \frac{\langle W, \phi(q, x_i) - \phi(q, x_j)\rangle \mathrm{F}}{|\mathcal{X}_q^+| \cdot |\mathcal{X}_q^-|}. \tag{5}$$

This indicates that the score $\langle W, \psi(q, y_q)\rangle_{\mathrm{F}}$ for a correct ranking y_q (the left-hand side of (3)) will be larger when the point-wise score $\langle W, \phi(q, \cdot)\rangle_{\mathrm{F}}$ is high for relevant points i, and low for irrelevant points j, i.e.,

$$\forall i \in \mathcal{X}_q^+, j \in \mathcal{X}_q^- : \langle W, \phi(q, x_i)\rangle_{\mathrm{F}} > \langle W, \phi(q, x_j)\rangle_{\mathrm{F}}. \tag{6}$$

Indeed, this will accumulate only positive terms in the score computation in (5), since a correct ranking orders all relevant results i before all irrelevant results j and, thus, each y_{ij} in the summation will be positive. Similarly, for incorrect rankings y, point-wise scores satisfying (6) will lead to smaller scores $\langle W, \psi(q, y)\rangle_{\mathrm{F}}$. Ideally, after training, W is maximally aligned to correct rankings y_q (i.e., $\langle W, \psi(q, y_q)\rangle_{\mathrm{F}}$ achieves large margin over scores $\langle W, \psi(q, y)\rangle_{\mathrm{F}}$ for incorrect rankings) by (approximately) satisfying (6). Consequently, at test time (i.e., in the absence of a correct ranking y_q), the ranking for a query q is predicted by sorting $i \in \mathcal{X}$ in descending order of point-wise score $\langle W, \phi(q, x_i)\rangle_{\mathrm{F}}$ [26].

This motivates the choice of ϕ used by MLR:

$$\phi(q, x_i) = -(q - x_i)(q - x_i)^{\mathsf{T}} \tag{7}$$

[2]The margin constraint (3) is based on a generalization of multi-class SVM, in which the score of a correct labeling exceeds the score of an incorrect labeling by the classification error (typically 0–1 loss) [31].

which upon taking an inner product with W, yields the negative, squared distance between q and i under W:

$$\begin{aligned}\langle W, \phi(q, x_i)\rangle_{\mathrm{F}} &= -\mathrm{tr}\left(W(q - x_i)(q - x_i)^{\mathsf{T}}\right)\\ &= -(q - x_i)^{\mathsf{T}} W (q - x_i)\\ &= -\|q - x_i\|_W^2. \end{aligned} \quad (8)$$

Descending point-wise score $\langle W, \phi(q, x_i)\rangle_{\mathrm{F}}$ therefore corresponds to increasing distance from q. As a result, the ranking predicted by descending score is equivalent to that predicted by increasing distance from q, which is precisely the ranking of interest for query-by-example recommendation.

The MLR optimization problem is listed as Algorithm 1. As in support vector machines [32], the objective consists of two competing terms: a regularization term $\mathrm{tr}(W)$, which is a convex approximation to the rank of the learned metric, and $1/n \sum \xi_q$ provides a convex upper bound on the empirical training loss Δ, and the two terms are balanced by a tradeoff parameter C. Although the full problem includes a super-exponential number of constraints (one for each $y \in \mathcal{Y}$, for each q), it can be approximated by cutting plane optimization techniques [16], [33].

Algorithm 1 Metric Learning to Rank **[16]**

Input: data $\mathcal{X} = \{q_1, q_2, \ldots, q_n\} \subset \mathbb{R}^d$,

correct rankings $\{y_q : q \in \mathcal{X}\}$,

slack trade-off $C > 0$

Output: $d \times d$ matrix $W \succeq 0$

$$\min_{W \succeq 0, \xi} \quad \mathrm{tr}(W) + C \cdot \frac{1}{n} \sum_{q \in \mathcal{X}} \xi_q$$

$$\text{s.t.} \quad \forall q \in \mathcal{X}, \quad \forall y \in \mathcal{Y} \text{ (permutations of } \mathcal{X}) :$$
$$\langle W, \psi(q, y_q)\rangle_{\mathrm{F}} \geq \langle W, \psi(q, y)\rangle \mathrm{F} + \Delta(y_q, y) - \xi_q.$$

III. Audio Representation

In order to compactly summarize audio signals, we represent each song as a histogram over a dictionary of timbral *codewords*. This general strategy has been proven effective in computer vision applications [34], as well as audio and music classification [35]–[37]. The efficiency and ease of implementation of the codeword histogram approach makes it an attractive choice for audio representation.

As a first step, a *codebook* is constructed by clustering a large collection of feature descriptors (Section III-A). Once the codebook has been constructed, each song is summarized by aggregating vector quantization (VQ) representations across all frames in the song, resulting in *codeword histograms* (Section III-B). Finally, histograms are represented in a nonlinear kernel space to facilitate better learning with MLR (Section III-C).

A. Codebook Training

Our general approach to constructing a codebook for vector quantization is to aggregate audio feature descriptors from a large pool of songs into a single bag-of-features, which is then clustered to produce the codebook.

For each song x in the codebook training set $\mathcal{X}_C$—which may generally be distinct from the MLR training set $\mathcal{X}$—we compute the first 13 Mel frequency cepstral coefficients (MFCCs) [38] from each half-overlapping 23-ms frame. From the time series of MFCC vectors, we compute the first and second instantaneous derivatives, which are concatenated to form a sequence of 39-dimensional dynamic MFCC (ΔMFCC) vectors [39]. These descriptors are then aggregated across all $x \in \mathcal{X}_C$ to form an unordered bag of features Z.

To correct for changes in scale across different ΔMFCC dimensions, each vector $z \in Z$ is normalized according to the sample mean $\mu \in \mathbb{R}^{39}$ and standard deviation $\sigma \in \mathbb{R}^{39}$ estimated from Z. The ith coordinate $z[i]$ is mapped by

$$z[i] \mapsto \frac{z[i] - \mu[i]}{\sigma[i]}. \quad (9)$$

The normalized ΔMFCC vectors are then clustered into a set $\mathcal{V}$ of $|\mathcal{V}|$ codewords by k-means (specifically, an online variant of Hartigan's method [40]).

B. (Top-τ) Vector Quantization

Once the codebook $\mathcal{V}$ has been constructed, a song x is represented as a histogram h_x over the codewords in $\mathcal{V}$. This proceeds in three steps: 1) a bag-of-features is computed from x's ΔMFCCs, denoted as $x = \{x_i\} \subset \mathbb{R}^{39}$; 2) each $x_i \in x$ is normalized according to (9); 3) the codeword histogram is constructed by counting the frequency with which each codeword $v \in \mathcal{V}$ quantizes an element of x:[3]

$$h_x[v] = \frac{1}{|x|} \sum_{x_i \in x} \mathbb{1}\left[v = \operatorname*{arg\,min}_{u \in \mathcal{V}} \|x_i - u\|\right]. \quad (10)$$

Codeword histograms are normalized by the number of frames $|x|$ in the song in order to ensure comparability between songs of different lengths; h_x may therefore be interpreted as a multinomial distribution over codewords.

Equation (10) derives from the standard notion of vector quantization (VQ), where each vector (e.g., data point x_i) is replaced by its closest quantizer. However, VQ can become unstable when a vector has multiple, (approximately) equidistant quantizers (Fig. 3, left), which is more likely to happen as the size of the codebook increases.

To counteract quantization errors, we generalize (10) to support *multiple* quantizers for each vector. For a vector x_i, a codebook $\mathcal{V}$, and a *quantization threshold* $\tau \in \{1, 2, \ldots, |\mathcal{V}|\}$, we define the quantization set

$$\operatorname*{arg\,min}_{u \in \mathcal{V}}{}^{\tau} \|x_i - u\| \doteq \{u \text{ is a } \tau \text{ - nearest neighbor of } x_i\}.$$

The *top-*τ codeword histogram for a song x is then constructed as

$$h_x^{\tau}[v] = \frac{1}{|x|} \sum_{x_i \in x} \frac{1}{\tau} \mathbb{1}\left[v \in \operatorname*{arg\,min}_{u \in \mathcal{V}}{}^{\tau} \|x_i - u\|\right]. \quad (11)$$

[3]To simplify notation, we denote by $h_x[v]$ the bin of histogram h_x corresponding to the codeword $v \in \mathcal{V}$. Codewords are assumed to be unique, and the usage should be clear from context.

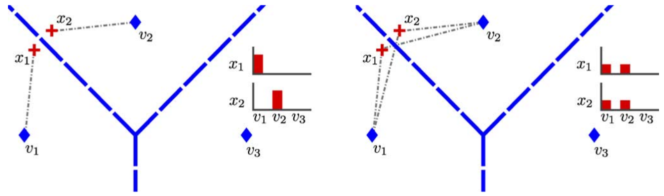

Fig. 3. Two close data points x_1, x_2 (+) and the Voronoi partition for three VQ codewords $v_1,\ v_2,\ v_3$ (♦). Left: hard VQ ($\tau = 1$) assigns similar data points to dissimilar histograms. Right: assigning each data point to its top $\tau = 2$ codewords reduces noise in codeword histogram representations.

Intuitively, (11) assigns $1/\tau$ mass to each of the τ closest codewords for each $x_i \in x$ (Fig. 3, right). Note that when $\tau = 1$, (11) reduces to (10). The normalization by $1/\tau$ ensures that $\sum_v h_x^\tau[v] = 1$, so that for $\tau > 1$, h_x^τ retains its interpretation as a multinomial distribution over $\mathcal{V}$.

It should be noted that top-τ is by no means the only way to handle over-quantization errors. In particular, the hierarchical Dirichlet process (HDP) method proposed by Hoffman, *et al.* addresses the quantization error problem (by using a probabilistic encoding), as well as the issue of determining the size of the codebook, and could easily be substituted into our framework [7]. However, as demonstrated in Section IV, Algorithm 1 adequately compensates for these effects. For the sake of simplicity and ease of reproducibility, we opted here to use the top-τ method.

C. Histogram Representation and Distance

After summarizing each song x by a codeword histogram h_x^τ, these histograms may be interpreted as vectors in $\mathbb{R}^{|\mathcal{V}|}$. Subsequently, for a query song q, retrieval may be performed by ordering $x \in \mathcal{X}$ according to increasing (Euclidean) distance $\|h_q^\tau - h_x^\tau\|$. After optimizing W with Algorithm 1, the same codeword histogram vectors may be used to perform retrieval with respect to the learned metric $\|h_q^\tau - h_x^\tau\|_W$.

However, treating codeword histograms directly as vectors in a Euclidean space ignores the simplicial structure of multinomial distributions. To better exploit the geometry of codeword histograms, we represent each histogram in a *probability product kernel* (PPK) space [41]. Inner products in this space can be computed by evaluating the corresponding *kernel function* k. For PPK space, k is defined as

$$k(h_q^\tau, h_x^\tau) = \sum_{v\in\mathcal{V}} \sqrt{h_q^\tau[v] \cdot h_x^\tau[v]}. \tag{12}$$

The PPK inner product in (12) is equivalent to the Bhattacharyya coefficient [42] between h_q^τ and h_x^τ. Consequently, distance in PPK space induces the same rankings as Hellinger distance between histograms.

Typically in kernel methods, data is represented implicitly in a (typically high-dimensional) Hilbert space via the $n \times n$ matrix of inner products between training points, i.e., the *kernel matrix* [43]. This representation enables efficient learning, even when the dimensionality of the kernel space is much larger than the number of points (e.g., for histogram-intersection kernels [44]) or infinite (e.g., radial basis functions). The MLR algorithm has been extended to support optimization of distances in such spaces by reformulating the optimization in terms of the kernel matrix, and optimizing an $n \times n$ matrix $W \succeq 0$ [45]. While kernel MLR supports optimization in arbitrary inner product spaces, it can be difficult to scale up to large training sets (i.e., large n), and may require making some simplifying approximations to scale up, e.g., restricting W to be diagonal.

However, for the present application, we can exploit the specific structure of the probability product kernel (on histograms) and optimize distances in PPK space with complexity that depends on $|\mathcal{V}|$ rather than n, thereby supporting larger training sets. Note that PPK enables an *explicit* representation of the data according to a simple, coordinate-wise transformation:

$$h_x^\tau[v] \mapsto \sqrt{h_x^\tau[v]} \tag{13}$$

which, since $k(h_x^\tau, h_x^\tau) = 1$ for all h_x^τ, can be interpreted as mapping the $|\mathcal{V}|$-dimensional simplex to the $|\mathcal{V}|$-dimensional unit sphere. Training data may therefore be represented as a $|\mathcal{V}| \times n$ data matrix, rather than the $n \times n$ kernel matrix. As a result, we can equivalently apply (13) to the data, and learn a $|\mathcal{V}| \times |\mathcal{V}|$ matrix W with Algorithm 1, which is more efficient than using kernel MLR when $|\mathcal{V}| < n$, as is often the case in our experiments. Moreover, the probability product kernel does not require setting hyper-parameters (e.g., the bandwidth of a radial basis function kernel), and thus simplifies the training procedure.

IV. Experiments

Our experiments are designed to simulate query-by-example content-based retrieval of songs from a fixed database. Fig. 4 illustrates the high-level experimental setup: training and evaluation are conducted with respect to collaborative filter similarity (as described in Section II-A). In this section, we describe the sources of collaborative filter and audio data, experimental procedure, and competing methods against which we compare.

A. Data

1) Collaborative Filter: Last.FM: Our collaborative filter data is provided by Last.fm,[4] and was collected by Celma [4, Ch. 3]. The data consists of a users-by-artists matrix F of 359 347 unique users and 186 642 unique, identifiable artists; the entry $F[ij]$ contains the number of times user i listened to artist j. We binarize the matrix by thresholding at 10, i.e., a user must listen to an artist at least ten times before we consider the association meaningful.

2) Audio: CAL10K: For our audio data, we use the CAL10K data set [46]. Starting from 10 832 songs by 4661 unique artists, we first partition the set of artists into those with at least 100 listeners in the binarized CF matrix (2015, the *experiment set*), and those with fewer than 100 listeners (2646, the *codebook set*). We then restrict the CF matrix to just those 2015 artists in the experiment set, with sufficiently many listeners. From this restricted CF matrix, we compute the artist-by-artist similarity matrix according to (2).

Artists in the codebook set, with insufficiently many listeners, are held out from the experiments in Section IV-B, but their songs are used to construct four codebooks as described in Section III-A. From each held out artist, we randomly select

[4]http://www.last.fm/

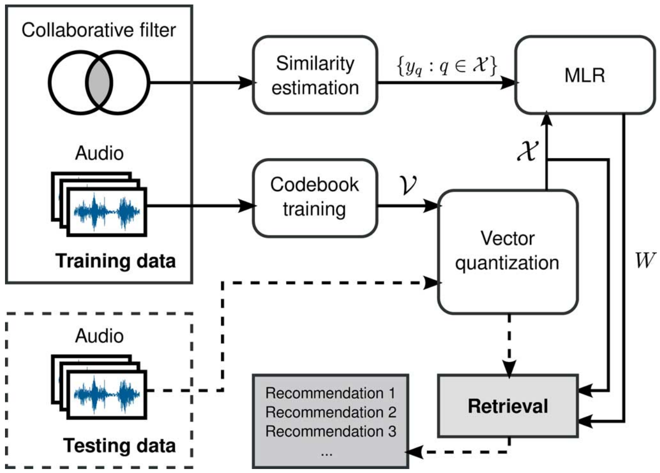

Fig. 4. Schematic diagram of training and retrieval. Here, "training data" encompasses both the subset of $\mathcal{X}$ used to train the metric W, and the codebook set $\mathcal{X}_C$ used to build the codebook $\mathcal{V}$. While, in our experiments, both sets are disjoint, in general, data used to build the codebook may also be used to train the metric.

TABLE I
STATISTICS OF CAL10K DATA, AVERAGED ACROSS TEN RANDOM TRAINING/VALIDATION/TEST SPLITS. *# RELEVANT* IS THE AVERAGE NUMBER OF RELEVANT SONGS FOR EACH TRAINING/VALIDATION/TEST SONG

	Training	Validation	Test
# Artists	806	604	605
# Songs	2122.3 ± 36.3	1589.3 ± 38.6	1607.5 ± 64.3
# Relevant	36.9 ± 16.4	36.4 ± 15.4	37.1 ± 16.0

one song, and extract a 5-s sequence of ΔMFCC vectors (431 half-overlapping 23-ms frames at 22 050 Hz). These samples are collected into a bag-of-features of approximately 1.1 million samples, which is randomly permuted, and clustered via online k-means in a single pass to build four codebooks of sizes $|\mathcal{V}| \in \{256, 512, 1024, 2048\}$, respectively. Cluster centers are initialized to the first (randomly selected) k points. Note that *only* the artists from the codebook set (and thus no artists from the experiment set) are used to construct the codebooks. As a result, the previous four codebooks are fixed throughout the experiments in the following section.

B. Procedure

For our experiments, we generate ten random splits of the experiment set of 2015 artists into 40% training, 30% validation, and 30% test *artists*.[5] For each split, the set of all training artist songs forms the *training set*, which serves as the database of "known" songs, $\mathcal{X}$. For each split, and for each (training/test/validation) artist, we then define the *relevant artist set* as the top 10 most similar *training*[6] artists. Finally, for any song q by artist i, we define q's *relevant song set*, $\mathcal{X}_q^+$, as all songs by all artists in i's relevant artist set. The songs by all other training artists, not in i's relevant artist set, are collected into $\mathcal{X}_q^-$, the set of irrelevant songs for q. The statistics of the training, validation, and test splits are collected in Table I.

[5]Due to recording effects and our definition of similarity, it is crucial to split at the level of artists rather than songs [47].

[6]Also for test and validation artists, we restrict the relevant artist set to the training artists to mimic the realistic setting of retrieving "known" songs from $\mathcal{X}$, given an "unknown" (test/validation) query.

For each of the four codebooks, constructed in the previous section, each song was represented by a histogram over codewords using (11), with $\tau \in \{1, 2, 4, 8\}$. Codeword histograms were then mapped into PPK space by (13). For comparison purposes, we also experiment with Euclidean distance and MLR on the raw codeword histograms.

To train the distance metric with Algorithm 1, we vary $C \in \{10^{-2}, 10^{-1}, \ldots, 10^9\}$. We experiment with three ranking losses Δ for training: area under the ROC curve (AUC), which captures global qualities of the ranking, but penalizes mistakes equally regardless of their position in the ranking; normalized discounted cumulative gain (NDCG), which applies larger penalties to mistakes at the beginning of the ranking than at the end, and is therefore more localized than AUC; and mean reciprocal rank (MRR), which is determined by the position of the first relevant result, and is therefore the most localized ranking loss under consideration here. After learning W on the training set, retrieval is evaluated on the validation set, and the parameter setting (C, Δ) which achieves highest AUC on the validation set is then evaluated on the test set.

To evaluate a metric W, the training set $\mathcal{X}$ is ranked according to distance from each test (validation) song q under W, and we record the mean AUC of the rankings over all test (validation) songs.

Prior to training with MLR, codeword histograms are compressed via principal components analysis (PCA) to capture 95% of the variance as estimated on the training set. While primarily done for computational efficiency, this step is similar to the latent perceptual indexing method described by Sundaram and Narayanan [35], and may also be interpreted as de-noising the codeword histogram representations. In preliminary experiments, compression of codeword histograms was not observed to significantly affect retrieval accuracy in either the native or PPK spaces (without MLR optimization).

C. Comparisons

To evaluate the performance of the proposed system, we compare to several alternative methods for content-based query-by-example song retrieval: first, similarity derived from comparing Gaussian mixture models of ΔMFCCs; second, an alternative (unsupervised) weighting of VQ codewords; and third, a high-level, automatic semantic annotation method. We also include a comparison to a manual semantic annotation method (i.e., driven by human experts), which although not content-based, can provide an estimate of an upper bound on what can be achieved by content-based methods. For both manual and automatic semantic annotations, we will also compare to their MLR-optimized counterparts.

1) Gaussian Mixtures: From each song, a Gaussian mixture model (GMM) over its ΔMFCCs was estimated via expectation-maximization [48]. Following Turnbull, *et al.* [49], each song is represented by a GMM consisting of eight components with diagonal covariance matrices.[7] The training set $\mathcal{X}$ is therefore represented as a collection of GMM distributions

[7]In addition to yielding the best performance for the auto-tagger described in [49], eight-component diagonal covariance GMMs yields audio representations of comparable space complexity to the proposed VQ approach.

$\{p_x : x \in \mathcal{X}\}$. This approach is representative of many previously proposed systems in the music information retrieval literature [6], [9], [50], and is intended to serve as a baseline against which we can compare the proposed VQ approach.

At test time, given a query song q, we first estimate its GMM p_q. We would then like to rank each $x \in \mathcal{X}$ by increasing Kullback–Leibler (KL) divergence [51] from p_q:

$$D(p_q \| p_x) = \int p_q(z) \log \frac{p_q(z)}{p_x(z)} \mathrm{d}z. \tag{14}$$

However, we do not have a closed-form expression for KL divergence between GMMs, so we must resort to approximate methods. Several such approximation schemes have been devised in recent years, including variational methods and sampling approaches [52]. Here, we opt for the Monte Carlo approximation

$$D(p_q \| p_x) \approx \sum_{i=1}^{m} \frac{1}{m} \log \frac{p_q(z_i)}{p_x(z_i)} \tag{15}$$

where $\{z_i\}_{i=1}^m$ is a collection of m independent samples drawn from p_q. Although the Monte Carlo approximation is considerably slower than closed-form approximations (e.g., variational methods), with enough samples, it often exhibits higher accuracy [50], [52]. Note that because we are only interested in the rank-ordering of $\mathcal{X}$ given p_q, it is equivalent to order each $p_x \in \mathcal{X}$ by increasing (approximate) cross-entropy

$$H(p_q, p_x) = \int p_q(z) \log \frac{1}{p_x(z)} \mathrm{d}z \approx \sum_{i=1}^{m} \frac{1}{m} \log \frac{1}{p_x(z_i)}. \tag{16}$$

For efficiency purposes, for each query q we fix the sample $\{z_i\}_{i=1}^m \sim p_q$ across all $x \in \mathcal{X}$. We use $m = 2048$ samples for each query, which was found to yield stable cross-entropy estimates in an informal, preliminary experiment.

2) TF-IDF: The algorithm described in Section II-B is a supervised approach to learning an optimal transformation of feature descriptors (in this specific case, VQ histograms). Alternatively, it is common to use the natural statistics of the data in an unsupervised fashion to transform the feature descriptors. As a baseline, we compare to the standard method of combining *term frequency-inverse document frequency* (TF-IDF) [53] representations with cosine similarity, which is commonly used with both text [53] and codeword representations [54].

Given a codeword histogram h_q^τ, for each $v \in \mathcal{V}$, $h_q^\tau[v]$ is mapped to its TF-IDF value by[8]

$$h_q^\tau[v] \mapsto h_q^\tau[v] \cdot \mathrm{IDF}[v] \tag{17}$$

where $\mathrm{IDF}[v]$ is computed from the statistics of the training set by[9]

$$\mathrm{IDF}[v] = \log \frac{|\mathcal{X}|}{|\{x \in \mathcal{X} : x[v] > 0\}|}. \tag{18}$$

Intuitively, $\mathrm{IDF}[v]$ assigns more weight to codewords v which appear in fewer songs, and reduces the importance of codewords appearing in many songs. The training set $\mathcal{X}$ is accordingly represented by TF-IDF vectors. At test time, each $x \in \mathcal{X}$ is ranked according to decreasing cosine-similarity to the query q:

$$\cos(h_q^\tau, h_x^\tau) = \frac{{h_q^\tau}^\top h_x^\tau}{\|h_q^\tau\| \cdot \|h_x^\tau\|}. \tag{19}$$

3) Automatic Semantic Tags: The proposed method relies on low-level descriptors to assess similarity between songs. Alternatively, similarity may be assessed by comparing high-level content descriptors in the form of *semantic tags*. These tags may include words to describe genre, instrumentation, emotion, etc. Because semantic annotations may not be available for novel query songs, we restrict attention to algorithms which automatically predict tags given only audio content.

In our experiments, we adapt the auto-tagging method proposed by Turnbull *et al.* [49]. This method summarizes each song by a *semantic multinomial distribution* (SMD) over a vocabulary of 149 tag words. Each tag t is characterized by a GMM p_t over ΔMFCC vectors, each of which was trained previously on the CAL500 data set [55]. A song q is summarized by a multinomial distribution s_q, where the tth entry is computed by the geometric mean of the likelihood of q's ΔMFCC vectors q_i under p_t:

$$s_q[t] \propto \left(\prod_{q_i \in q} p_t(q_i) \right)^{1/|q|}. \tag{20}$$

(Each SMD s_q is normalized to sum to 1.) The training set $\mathcal{X}$ is thus described as a collection of SMDs $\{s_x : x \in \mathcal{X}\}$. At test time, $\mathcal{X}$ is ranked according to increasing distance from the test query under the probability product kernel[10] as described in Section III-C. This representation is also amenable to optimization by MLR, and we will compare to retrieval performance after optimizing PPK representations of SMDs with MLR.

4) Human Tags: Our final comparison uses semantic annotations manually produced by humans, and may be interpreted as an upper bound on the performance of automated content analysis. Each song in CAL10K includes a partially observed, binary annotation vector over a vocabulary of 1053 tags from the Music Genome Project[11]. The annotation vectors are *weak* in the sense that a 1 indicates that the tag applies, while a 0 indicates only that the tag *may not* apply.

In our experiments, we observed the best performance by using cosine similarity as the retrieval function, although we also tested TF-IDF and Euclidean distances. As in the auto-tag case, we will also compare to tag vectors after optimization by MLR. When training with MLR, annotation vectors were compressed via PCA to capture 95% of the training set variance.

[8]Since codeword histograms are pre-normalized, there is no need to recompute the term frequency in (17).

[9]To avoid division by 0, we define $\mathrm{IDF}[v] = 0$ for any codeword v which is not used in the training set.

[10]We also experimented with χ^2-distance, ℓ_1, Euclidean, and (symmetrized) KL divergence, but PPK distance was always statistically equivalent to the best-performing distance.

[11]http://www.pandora.com/mgp.shtml

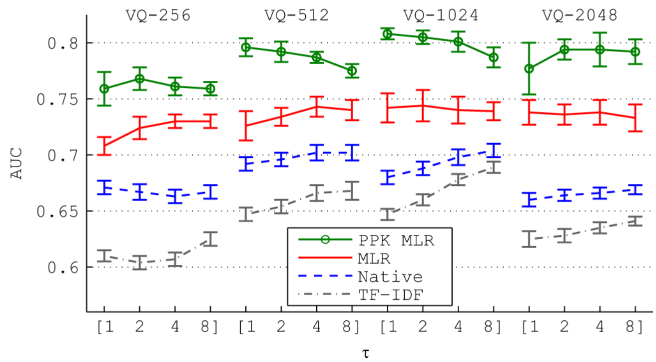

Fig. 5. Retrieval accuracy with vector quantized ΔMFCC representations. Each grouping corresponds to a different codebook size $|\mathcal{V}| \in \{256,512,1024,2048\}$. Each point within a group corresponds to a different quantization threshold $\tau \in \{1,2,4,8\}$. *TF-IDF* refers to cosine similarity applied to IDF-weighted VQ histograms; *Native* refers to Euclidean distance on unweighted VQ histograms; *MLR* refers to VQ histograms after optimization by MLR; *PPK MLR* refers to distances after mapping VQ histograms into probability product kernel space and subsequently optimizing with MLR. Error bars correspond to one standard deviation across trials.

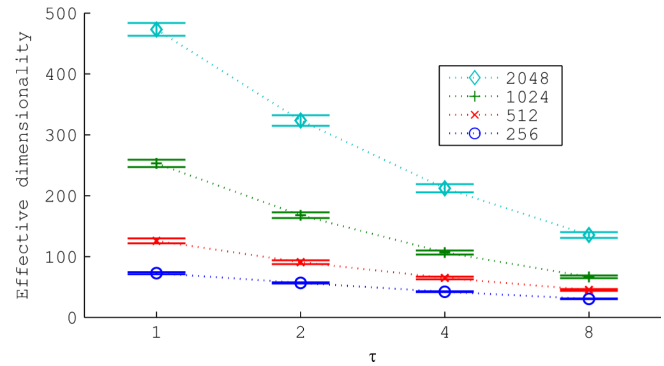

Fig. 6. *Effective dimensionality* of codeword histograms in PPK space, i.e., the number of principal components necessary to capture 95% of the training set's variance, as a function of the quantization threshold τ. (The results reported in the figure are the average effective dimension ± one standard deviation across trials.)

V. Results

A. Vector Quantization

In a first series of experiments, we evaluate various approaches and configurations based on VQ codeword histograms. Fig. 5 lists the AUC achieved by four different approaches (*Native*, *TF-IDF*, *MLR*, *PPK-MLR*), based on VQ codeword histograms, for each of four codebook sizes and each of four quantization thresholds. We observe that using Euclidean distance on raw codeword histograms[12] (*Native*) yields significantly higher performance for codebooks of intermediate size (512 or 1024) than for small (256) or large (2048) codebooks. For the 1024 codebook, increasing τ results in significant gains in performance, but it does not exceed the performance for the 512 codebook. The decrease in accuracy for $|\mathcal{V}| = 2048$ suggests that performance is indeed sensitive to overly large codebooks.

After learning an optimal distance metric with MLR on raw histograms (i.e., not PPK representations) (*MLR*), we observe two interesting effects. First, MLR optimization always yields significantly better performance than the native Euclidean distance. Second, performance is much less sensitive to the choice of codebook size and quantization threshold: all settings of τ for codebooks of size at least $|\mathcal{V}| \geq 512$ achieve statistically equivalent performance.

Finally, we observe the highest performance by combining the PPK representation with MLR optimization (*PPK-MLR*). For $|\mathcal{V}| = 1024$, $\tau = 1$, the mean AUC score improves from 0.680 ± 0.006 (Native) to 0.808 ± 0.005 (PPK-MLR). The effects of codebook size and quantization threshold are diminished by MLR optimization, although they are slightly more pronounced than in the previous case without PPK. We may then ask: does top-τ VQ provide any benefit?

Fig. 6 lists the effective dimensionality—the number of principal components necessary to capture 95% of the training set's variance—of codeword histograms in PPK space as a function of quantization threshold τ. Although for the best-performing codebook size $|\mathcal{V}| = 1024$, each of $\tau \in \{1,2,4\}$ achieves statistically equivalent performance, the effective dimensionality varies from 253.1 ± 6.0 $(\tau = 1)$ to 106.6 ± 3.3 $(\tau = 4)$. Thus, top-τ VQ can be applied to dramatically reduce the dimensionality of VQ representations, which in turn reduces the number of parameters learned by MLR, and therefore improves the efficiency of learning and retrieval, without significantly degrading performance.

B. Qualitative Results

Fig. 7 illustrates an example optimized similarity space produced by MLR on PPK histogram representations, as visualized in two dimensions by t-SNE [56]. Even though the algorithm is never exposed to any explicit semantic information, the optimized space does exhibit regions which seem to capture intuitive notions of genre, such as *hip-hop*, *metal*, and *classical*.

Table II illustrates a few example queries and their top-5 closest results under the Euclidean and MLR-optimized metric. The native space seems to capture similarities due to energy and instrumentation, but does not necessarily match CF similarity. The optimized space captures aspects of the audio data which correspond to CF similarity, and produces playlists with more relevant results.

C. Comparison

Fig. 5 lists the accuracy achieved by using TF-IDF weighting on codeword histograms. For all VQ configurations (i.e., for each codebook size and quantization threshold) TF-IDF significantly degrades performance compared to MLR-based methods, which indicates that inverse document frequency may not be as an accurate predictor of salience in codeword histograms as in natural language [53].

Fig. 8 shows the performance of all other methods against which we compare. First, we observe that *raw* SMD representations provide more accurate retrieval than both the GMM approach and *raw* VQ codeword histograms (i.e., prior to optimization by MLR). This may be expected, as previous studies have demonstrated superior query-by-example retrieval

[12]For clarity, we omit the performance curves for native Euclidean distance on PPK representations, as they do not differ significantly from the *Native* curves shown.

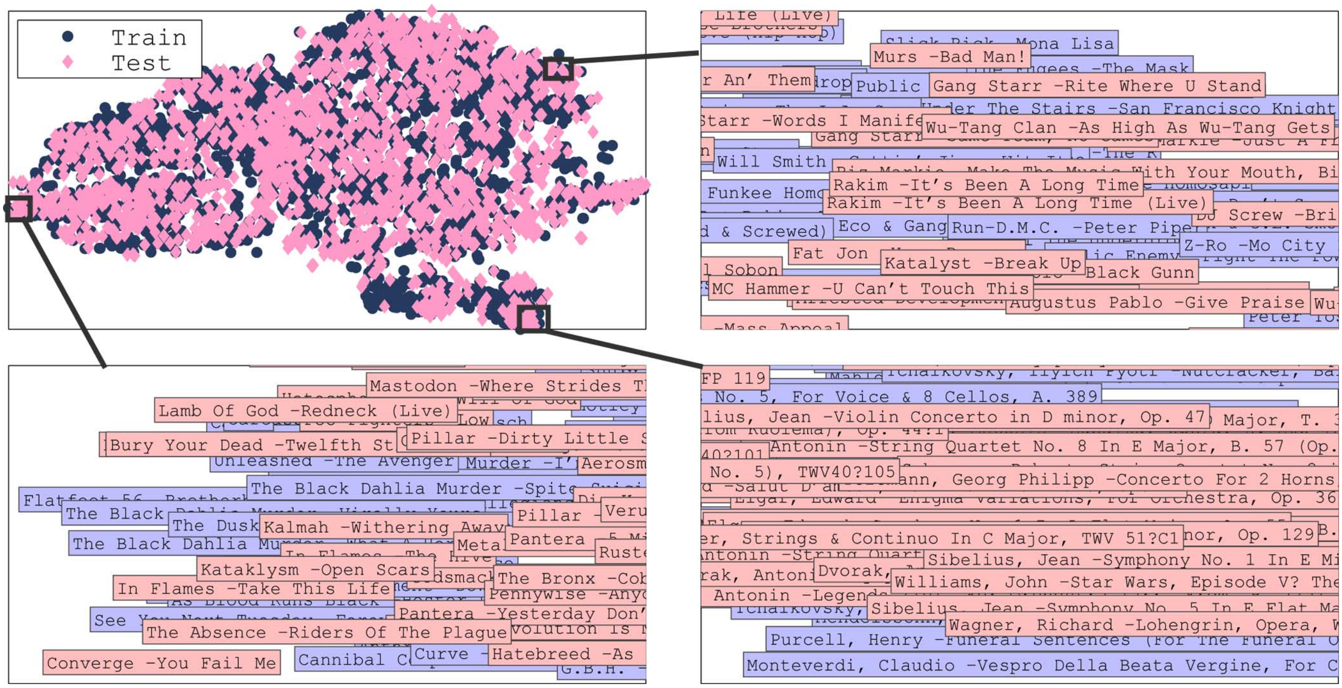

Fig. 7. A t-SNE visualization of the optimized similarity space produced by PPK+MLR on one training/test split of the data ($|\mathcal{V}| = 1024,\ \tau = 1$). Close-ups on three peripheral regions reveal *hip-hop* (upper-right), *metal* (lower-left), and *classical* (lower-right) genres.

TABLE II
EXAMPLE PLAYLISTS GENERATED BY 5-NEAREST (TRAINING) NEIGHBORS OF THREE DIFFERENT QUERY (TEST) SONGS (LEFT) USING EUCLIDEAN DISTANCE ON RAW CODEWORD HISTOGRAMS (CENTER) AND MLR-OPTIMIZED PPK DISTANCES (RIGHT). RELEVANT RESULTS ARE INDICATED BY ▶

Test query	VQ (Native)	VQ (PPK+MLR)
Ornette Coleman - Africa is the Mirror of All Colors	Judas Priest - You've Got Another Thing Comin' Def Leppard - Rock of Ages KC & The Sunshine Band - Give it Up Wynton Marsalis - Caravan Ringo Starr - It Don't Come Easy	Wynton Marsalis - Caravan ▶Dizzy Gillespie - Dizzy's Blues ▶Michael Brecker - Two Blocks from the Edge ▶Eric Dolphy - Miss Ann (live) Ramsey Lewis - Here Comes Santa Claus
Fats Waller - Winter Weather	▶Dizzy Gillespie - She's Funny that Way Enrique Morente - Solea Chet Atkins - In the Mood Rachmaninov - Piano Concerto #4 in Gmin Eluvium - Radio Ballet	Chet Atkins - In the Mood ▶Charlie Parker - What Is This Thing Called Love? ▶Bud Powell - Oblivion ▶Bob Wills & His Texas Playboys - Lyla Lou ▶Bob Wills & His Texas Playboys - Sittin' On Top Of The World
The Ramones - Go Mental	Def Leppard - Promises ▶The Buzzcocks - Harmony In My Head Los Lonely Boys - Roses Wolfmother - Colossal Judas Priest - Diamonds and Rust (live)	▶The Buzzcocks - Harmony In My Head Motley Crue - Same Ol' Situation ▶The Offspring - Gotta Get Away ▶The Misfits - Skulls ▶AC/DC - Who Made Who (live)

performance when using semantic representations of multimedia data [57], [58].

Moreover, SMD and VQ can be optimized by MLR to achieve significantly higher performance than raw SMD and VQ, respectively. The semantic representations in SMD compress the original audio content to a small set of descriptive terms, at a higher level of abstraction. In raw form, this representation provides a more robust set of features, which improves recommendation performance compared to matching low-level content features that are often noisier. On the other hand, semantic representations are inherently limited by the choice of vocabulary and may prematurely discard important discriminative information (e.g., subtle distinctions within sub-genres). This renders them less attractive as starting point for a metric learning algorithm like MLR, compared to less-compressed (but possibly noisier) representations, like VQ. Indeed, the latter may retain more information for MLR to learn an appropriate similarity function. This is confirmed by our experiments: MLR improves VQ significantly more than it does for SMD. As a result, MLR-VQ outperforms all other content-based methods in our experiments.

Finally, we provide an estimate of an upper bound on what can be achieved by automatic, content-based methods, by evaluating the retrieval performance when using manual annotations (*Tag* in Fig. 8): 0.834 ± 0.005 with cosine similarity, and 0.907 ± 0.008 with MLR-optimized similarity. The improvement in accuracy for human tags, when using MLR, indicates that even handcrafted annotations can be improved by learning an optimal distance over tag vectors. By contrast, TF-IDF on human tag vectors decreases performance to 0.771 ± 0.004, indicating that IDF does not accurately model (binary) tag salience. The gap in performance between content-based methods and

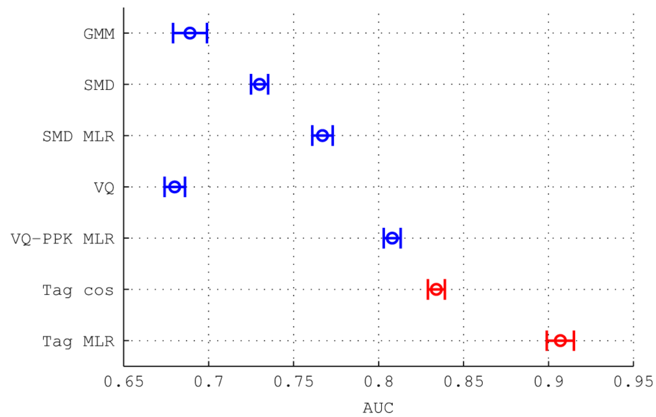

Fig. 8. Comparison of VQ-based retrieval accuracy to competing methods. *VQ* corresponds to a codebook of size $V = 1024$ with quantization threshold $\tau = 1$. *Tag*-based methods (red) use human annotations, and are not automatically derived from audio content. Error bars correspond to one standard deviation across trials.

manual annotations suggests that there is still room for improvement. Closing this gap may require incorporating more complex features to capture rhythmic and structural properties of music which are discarded by the simple timbral descriptors used here.

VI. Conclusion

In this paper, we have proposed a method for improving content-based audio similarity by learning from a sample of collaborative filter data. Collaborative filters form the basis of state-of-the-art recommendation systems, but cannot directly form recommendations or answer queries for items which have not yet been consumed or rated. By optimizing content-based similarity from a collaborative filter, we provide a simple mechanism for alleviating the cold-start problem and extending music recommendation to novel or less known songs.

By using implicit feedback in the form of user listening history, we can efficiently collect high-quality training data without active user participation, and as a result, train on larger collections of music than would be practical with explicit feedback or survey data. Our notion of similarity derives from user activity in a bottom-up fashion, and obviates the need for coarse simplifications such as genre or artist agreement.

Our proposed top-τ VQ audio representation enables efficient and compact description of the acoustic content of music data. Combining this audio representation with an optimized distance metric yields similarity calculations which are both efficient to compute and substantially more accurate than competing content-based methods. The proposed metric learning framework is robust with respect to the choice of codebook size and VQ threshold τ, and yields stable performance over a broad range of VQ configurations.

While in this work, our focus remains on music recommendation applications, the proposed methods are quite general, and may apply to a wide variety of applications involving content-based similarity, such as nearest-neighbor classification of audio signals.

References

[1] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Commun. ACM*, vol. 35, pp. 61–70, Dec. 1992.

[2] L. Barrington, R. Oda, and G. Lanckriet, "Smarter than genius? Human evaluation of music recommender systems," in *Proc. 10th Int. Conf. Music Inf. Retrieval*, 2009.

[3] J. Kim, B. Tomasik, and D. Turnbull, "Using artist similarity to propagate semantic information," in *Proc. 10th Int. Soc. Music Inf. Retrieval Conf., ser. ISMIR*, 2009.

[4] O. Celma, *Music Recommendation and Discovery in the Long Tail*. New York: Springer, 2010.

[5] B. Logan and A. Salomon, "A music similarity function based on signal analysis," in *Proc. IEEE Int. Conf. Multimedia and Expo*, 2001, pp. 745–748.

[6] J.-J. Aucouturier and F. Pachet, "Music similarity measures: What's the use?," in *Proc. 3rd Int. Conf. Music Inf. Retrieval, ser. ISMIR*, 2002, pp. 157–163.

[7] M. Hoffman, D. Blei, and P. Cook, "Content-based musical similarity computation using the hierarchical Dirichlet process," in *Proc. 9th Int. Conf. Music Inf. Retrieval, ser. ISMIR*, 2008, pp. 349–354.

[8] D. Ellis, B. Whitman, A. Berenzweig, and S. Lawrence, "The quest for ground truth in musical artist similarity," in *Proc. 2nd Int. Symp. Music Inf. Retrieval, ser. ISMIR*, Oct. 2002, pp. 170–177.

[9] A. Berenzweig, B. Logan, D. P. Ellis, and B. Whitman, "A large-scale evaluation of acoustic and subjective music-similarity measures," *Comput. Music J.*, vol. 28, no. 2, pp. 63–76, 2004.

[10] M. Slaney, K. Weinberger, and W. White, "Learning a metric for music similarity," in *Proc. 9th Int. Conf. Music Inf. Retrieval, ser. ISMIR*, Sep. 2008, pp. 313–318.

[11] B. McFee and G. Lanckriet, "Learning multi-modal similarity," *J. Mach. Learn. Res.*, vol. 12, pp. 491–523, Feb. 2011.

[12] M. Slaney and W. White, "Similarity based on rating data," in *Proc. 8th Int. Conf. Music Inf. Retrieval, ser. ISMIR*, 2007, pp. 479–484.

[13] K. Yoshii, M. Goto, K. Komatani, T. Ogata, and H. Okuno, "An efficient hybrid music recommender system using an incrementally trainable probabilistic generative model," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 16, no. 2, pp. 435–447, Feb. 2008.

[14] R. Stenzel and T. Kamps, "Improving content-based similarity measures by training a collaborative model," in *Proc. 6th Int. Conf. Music Inf. Retrieval, ser. ISMIR*, 2005, pp. 264–271.

[15] B. McFee, L. Barrington, and G. Lanckriet, "Learning similarity from collaborative filters," in *Proc. 11th Int. Soc. Music Inf. Retrieval Conf., ser. ISMIR*, 2010.

[16] B. McFee and G. Lanckriet, J. Fürnkranz and T. Joachims, Eds., "Metric learning to rank," in *Proc. 27th Int. Conf. Mach. Learn., ser. ICML*, Haifa, Israel, Jun. 2010, pp. 775–782.

[17] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web, ser. WWW '01.*, New York, 2001, pp. 285–295.

[18] M. Deshpande and G. Karypis, "Item-based top-n recommendation algorithms," *ACM Trans. Inf. Syst.*, vol. 22, pp. 143–177, Jan. 2004.

[19] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proc. 8th IEEE Int. Conf. Data Mining, ser. ICDM*, 2008, pp. 263–272.

[20] P. Jaccard, "Étude comparative de la distribution florale dans une portion des alpes et des jura," *Bull. del la Soc. Vaudoise des Sciences Naturelles*, vol. 37, pp. 547–579, 1901.

[21] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell, "Distance metric learning, with application to clustering with side-information," in *Advances in Neural Information Processing Systems, Ser. NIPS*. Cambridge, MA: MIT Press, 2003, pp. 505–512.

[22] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in Neural Information Processing Systems, ser. NIPS*, Y. Weiss, B. Schölkopf, and J. Platt, Eds. Cambridge, MA: MIT Press, 2006, pp. 451–458.

[23] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *Proc. 24th Int. Conf. Machine learning, ser. ICML.*, New York, 2007, pp. 209–216.

[24] J. P. Egan, *Signal Detection Theory and ROC Analysis, Ser. Series in Cognition and Perception*. New York: Academic, 1975.

[25] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large margin methods for structured and interdependent output variables," *J. Mach. Learn. Res.*, vol. 6, pp. 1453–1484, 2005.

[26] T. Joachims, "A support vector method for multivariate performance measures," in *Proc. 22nd Int. Conf. Mach. Learn., ser. ICML.*, New York, 2005, pp. 377–384.

[27] Y. Yue, T. Finley, F. Radlinski, and T. Joachims, "A support vector method for optimizing average precision," in *Proc. 30th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, ser. SIGIR.*, New York, 2007, pp. 271–278.

[28] S. Chakrabarti, R. Khanna, U. Sawant, and C. Bhattacharyya, "Structured learning for non-smooth ranking losses," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discov. and Data Mining, ser. KDD.*, New York, 2008, pp. 88–96.

[29] E. M. Voorhees, "Overview of the trec 2001 question answering track," in *Proc. 10th Text Retrieval Conf. (TREC)*, 2001, pp. 42–51.

[30] K. J. Kekäläinen, "Ir evaluation methods for retrieving highly relevant documents," in *Proc. 23rd Annu. Int. ACM SIGIR Conf.n Res. Develop. in Inf. Retrieval, ser. SIGIR.*, New York, 2000, pp. 41–48.

[31] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *J. Mach. Learn. Res.*, vol. 2, pp. 265–292, 2002.

[32] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[33] T. Joachims, T. Finley, and C.-N. J. Yu, "Cutting-plane training of structural SVMs," *Mach. Learn.*, vol. 77, no. 1, pp. 27–59, 2009.

[34] L. Fei-Fei and P. Perona, "A Bayesian hierarchical model for learning natural scene categories," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recogn., ser. CVPR*, 2005, pp. 524–531.

[35] S. Sundaram and S. Narayanan, "Audio retrieval by latent perceptual indexing," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., ser. ICASSP*, 2008, pp. 49–52.

[36] K. Seyerlehner, G. Widmer, and P. Knees, "Frame level audio similarity—A codebook approach," in *Proc. 11th Int. Conf. Digital Audio Effects, ser. DAFx*, Espoo, Finland, 2008.

[37] M. Hoffman, D. Blei, and P. Cook, "Easy as CBA: A simple probabilistic model for tagging music," in *Proc. 10th Int. Conf. Music Inf. Retrieval, ser. ISMIR*, 2009.

[38] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*. Upper Saddle River, NJ: Prentice-Hall, 1993.

[39] C. Buchanan, "Semantic-based audio recognition and retrieval," M.S. thesis, School of Informatics, Univ. of Edinburgh, Edinburgh, U.K., 1995.

[40] J. A. Hartigan, *Clustering Algorithms*. New York: Wiley, 1975.

[41] T. Jebara, R. Kondor, and A. Howard, "Probability product kernels," *J. Mach. Learn. Res.*, vol. 5, pp. 819–844, Dec. 2004.

[42] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by probability distributions," *Bull. Calcutta Math. Soc.*, vol. 35, pp. 99–109, 1943.

[43] B. Schölkopf and A. Smola, *Learning With Kernels*. Cambridge, MA: MIT Press, 2002.

[44] A. Barla, F. Odone, and A. Verri, "Histogram intersection kernel for image classification," in *Proc. Int. Conf. Image Process., ser. ICIP*, 2003, vol. 3, pp. III-513–III-16.

[45] C. Galleguillos, B. McFee, S. Belongie, and G. Lanckriet, "From region similarity to category discovery," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recogn., ser. CVPR*, 2011, pp. 2665–2672.

[46] D. Tingle, Y. Kim, and D. Turnbull, "Exploring automatic music annotation with acoustically-objective tags," in *Proc. IEEE Int. Conf. Multimedia Inf. Retrieval*, 2010.

[47] B. Whitman, G. Flake, and S. Lawrence, "Artist detection in music with minnowmatch," in *Proc. IEEE Signal Process. Soc. Workshop, Neural Netw. for Signal Process. XI*, 2001, pp. 559–568.

[48] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. R. Statist. Soc. Ser. B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[49] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet, "Semantic annotation and retrieval of music and sound effects," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 16, no. 2, pp. 467–476, Feb. 2008.

[50] J. Jensen, D. Ellis, M. Christensen, and S. Jensen, "Evaluation of distance measures between gaussian mixture models of MFCCs," in *Proc. 8th Int. Conf. Music Inf. Retrieval, ser. ISMIR*, 2007, pp. 107–108.

[51] S. Kullback, *Information Theory and Statistics*, 2nd ed. New York: Dover, 1968.

[52] J. Hershey and P. Olsen, "Approximating the Kullback Leibler divergence between gaussian mixture models," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., ser. ICASSP*, 2007, vol. 4, pp. IV-317–IV-320.

[53] G. Salton and C. Buckley, "Term weighting approaches in automatic text retrieval," Ithaca, NY, 1987.

[54] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Proc. 9th IEEE Int. Conf. Comput. Vis.—Vol. 2, ser. ICCV '03.*, Washington, DC, 2003, p. 1470 [Online]. Available: http://portal.acm.org/citation.cfm?id=946247.946751

[55] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet, "Towards musical query-by-semantic-description using the cal500 data set," in *Proc. 30th Annu. Int. ACM SIGIR Conf. Res. Develop. in Inf. Retrieval, ser. SIGIR.*, New York, 2007, pp. 439–446.

[56] L. van der Maaten and G. Hinton, "Visualizing high-dimensional data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, 2008.

[57] N. Rasiwasia, P. Moreno, and N. Vasconcelos, "Bridging the gap: Query by semantic example," *IEEE Trans. Multimedia*, vol. 9, no. 5, pp. 923–938, Aug. 2007.

[58] L. Barrington, A. Chan, D. Turnbull, and G. Lanckriet, "Audio information retrieval using semantic similarity," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., ser. ICASSP*, Apr. 2007, vol. 2, pp. II-725–II-728.

Brian McFee (S'10) received the B.S. degree in computer science from the University of California, Santa Cruz, in 2003 and the M.S. degree in computer science and engineering from the University of California at San Diego (UCSD), La Jolla, in 2008. He is currently pursuing a Ph.D. degree in computer science and engineering at UCSD.

His research focuses on machine learning, with applications in music information retrieval and recommendation.

Mr. McFee was a recipient of the Qualcomm Innovation Fellowship in 2010.

Luke Barrington received the Ph.D. degree in electrical and computer engineering from the University of California at San Diego (UCSD), La Jolla, with a thesis titled "Machines that Understand Music."

He developed frequency-temporal models of music that learned to automatically describe audio content with relevant words. He pioneered research in human computation and crowd sourcing as a means to train machine learning models. During the Ph.D. degree, he co-founded the UCSD Computer Audition Laboratory, published over 20 articles in journals and conferences, and was awarded two patents. He currently serves as founder and CTO of Tomnod, inc., where he leads the development of online applications that collect analyses of satellite and aerial imagery data from tens of thousands of users.

Dr. Barrington received the UCD Young Engineer of the year in 2001. In 2005, he was a National Science Foundation (NSF) EAPSI Fellow in Japan. From 2006 to 2008, he was the recipient of a U.S. NSF IGERT Fellowship and in 2010 he received the Qualcomm Innovation Fellowship. He is an avid musician and wails on the guitar.

Gert Lanckriet (M'10) received the M.S. degree in electrical engineering from the Katholieke Universiteit Leuven, Leuven, Belgium, in 2000 and the M.S. and Ph.D. degrees in electrical engineering and computer science from the University of California, Berkeley, in 2001 and 2005, respectively.

In 2005, he joined the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, where he heads the Computer Audition Laboratory. His research focuses on the interplay of convex optimization, machine learning, and signal processing, with applications in computer audition and music information retrieval.

Prof. Lanckriet was awarded the SIAM Optimization Prize in 2008 and is the recipient of a Hellman Fellowship, an IBM Faculty Award, an NSF CAREER Award, and an Alfred P. Sloan Foundation Research Fellowship. In 2011, MIT Technology Review named him one of the 35 top young technology innovators in the world (TR35).