Combining Audio Content and Social Context for Semantic Music Discovery

Douglas Turnbull[*]
Computer Science Department
Swarthmore College
Swarthmore, PA, USA
turnbull@cs.swarthmore.edu

Luke Barrington[*], Mehrdad Yazdani &
Gert Lanckriet
Electrical & Computer Engineering
University of California, San Diego
La Jolla, CA, USA
lukeinusa@gmail.com,myazdani@ucsd.edu,
gert@ece.ucsd.edu

ABSTRACT

When attempting to annotate music, it is important to consider both acoustic content and social context. This paper explores techniques for collecting and combining multiple sources of such information for the purpose of building a *query-by-text* music retrieval system. We consider two representations of the acoustic content (related to timbre and harmony) and two social sources (social tags and web documents). We then compare three algorithms that combine these information sources: calibrated score averaging (CSA), RankBoost, and kernel combination support vector machines (KC-SVM). We demonstrate empirically that each of these algorithms is superior to algorithms that use individual information sources.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; I.2.m [**Computing Methodologies**]: Artificial Intelligence; J.5 [**Computer Applications**]: Arts and Humanities—*Music*

General Terms

Algorithms, Design, Experimentation

Keywords

combining data sources, music IR, calibrated score averaging, RankBoost, kernel combination SVM

1. INTRODUCTION

Most academic and commercial music information retrieval (IR) systems focus on *either* content-based analysis of audio signals *or* context-based analysis of webpages, user preference information (i.e., collaborative filtering) or

---

[*]Both authors contributed equally to this work.

social tagging data. However, it seems natural that we can improve music IR by combining information related to *both* the audio content and social context of music. In this paper, we compare three techniques that combine multiple source of audio and social information about music.

We are explicitly interested in improving text-based semantic music annotation and retrieval. For example, one might say that "Wild Horses" by the Rolling Stones is "a sad folk-rock tune that features somber strumming on an acoustic guitar, clean electric slide guitar and plaintive vocals." Such descriptions are full of semantic information that is useful for music information retrieval. That is, we can index (i.e., annotate) music with *tags*, which are short text-based tokens, such as "sad", "folk-rock", and "electric slide guitar". More generally, for a large music corpus and a large vocabulary of tags, we are interested in representing each song-tag pair with a (probabilistic) score that reflects the strength of the semantic association between each song and each tag. Then, when a user enters a text-based query, we can extract tags from the query, rank-order the songs using the relevance scores for those tags, and return a list of the top scoring (i.e., most relevant) songs (e.g., see Table 1).

Semantic information for music can be obtained from a variety of sources [32]. For example, tags for music can be collected from humans using surveys, social tagging websites or annotation games [35, 21]. In addition, the relevance of tags to songs can be calculated automatically using content-based audio analysis [22, 7, 34] or by text-mining associated web documents [38, 15]. Taken together, these complementary sources of semantic information provide a description of the acoustic content and place the music in a social context.

In this paper, we consider three sources of music information that may be used to annotate songs for the purpose of semantic music retrieval: audio content, social tags and web documents. We analyze the audio signal by using two acoustic feature representations, one related to timbre and one related to harmony. We also use two more socially-situated sources, one based on social tags and one based on web documents. For each of these four representations, we describe algorithms that evaluate the relevance of a song to all tags from a given vocabulary. Using such algorithms, we can retrieve songs from a test corpus, ordered by their relevance to a given text query. We evaluate the performance of these algorithms using the CAL-500 human-annotated corpus of 500 songs labeled with 72 tags [33].

We then describe and compare three algorithms that *combine* the information produced by each music representation: calibrated score averaging (CSA), RankBoost [10], and the kernel combination support vector machine (KC-SVM) [18]. CSA and RankBoost are similar in that they combine sets of rank-orderings, where each rank-ordering comes from a separate information representation. They differ in how they deal with missing data and how they combine the rankings. For the KC-SVM algorithm, we first design a set of kernel matrices, where each is derived from one of the music representations. We then use convex optimization to learn the optimal linear combination of these kernel matrices, producing a single "combined" kernel which can be used by a support vector machine (SVM) to rank order test set songs.

The following section describes some of the related work on annotating music with tags, as well as existing approaches to combining multiple representations of music information. Section 3 describes how we annotate music by analyzing audio signals, collecting tags from social networks, and processing text documents that are downloaded from the Internet. Section 4 describes three algorithms that combine these three sources of music information. Section 5 provides a comparative analysis of these algorithms and contrasts them to approaches that use only one source of information. We conclude in Section 6.

2. RELATED WORK

Early work on content-based audio analysis for text-based music information retrieval focused (and continues to focus) on music classification by genre, emotion, and instrumentation (e.g., [36, 20, 8]). These classification systems effectively "tag" music with class labels (e.g., 'blues', 'sad', 'guitar'). Recently, *autotagging* systems have been developed to annotate music with a larger, more diverse vocabulary of (non-mutually exclusive) tags [34, 22, 7, 31].

Recently, eleven autotagging systems were compared head-to-head in the "Audio Tag Classification" task of the 2008 Music Information Retrieval Evaluation eXchange (MIREX) [5]. Due to multiple evaluation metrics and lack of statistical significance, there was no clear "best" system, but our system was one of the top performing systems for a number of the evaluation metrics [34]. Our system uses a generative approach that learns a Gaussian mixture model (GMM) distribution over an audio feature space for each tag in the vocabulary. We use this approach for content-based music annotation (see Section 3.1).

Mandel and Ellis proposed another top performing approach in which they learn a binary SVM for each tag in the vocabulary [22]. They use Platt scaling [27] to convert SVM decision function scores to probabilities so that tag relevance can be compared across multiple SVMs. We follow a similar procedure in Section 4.3. Eck et. al. [7] also use a discriminative approach by learning a boosted decision stump classifier for each tag. Finally, Sordo et. al. [31] present a non-parametric approach that uses a content-based measure of music similarity to propagate tags from annotated songs to similar songs that have not been annotated.

Socially-oriented music information can be mined from corpora of text documents. Whitman and Ellis [38] represent album reviews as (TF-IDF) document vectors over tag vocabularies of n-grams, adjectives and noun phrases. They use the TF-IDF weights for each tag to train and evaluate a content-based autotagging system. Knees et al. [15] propose

an alternative approach called *rank-based relevance scoring* in which they collect a mapping from songs to a large corpus of webpages by querying a search engine (e.g., Google) with song, album and artist names. When a user enters a free-text query string, the corpus of webpages is ranked using an IR approach and then the mapping from webpages back to songs is used to retrieve relevant songs. This approach improves on the standard vector space model (e.g., TF-IDF) [15]. In Section 3.2.2, we use a variant of this approach to extract information from web documents.

A second source of social music information comes directly from users who "tag" music with free-text tokens. These annotations are collected, summarized, and made available by musically-oriented social networking sites like Last.fm and MyStrands. While they have become a popular source of semantic music information [16, 19], Lamere and Pampalk note that there is a substantial sparse data problem due to a large numbers of unique songs (150M) and tags (1.2M) [17]. This problem is compounded by both popularity bias (e.g., only popular songs are tagged) and the cold-start problem (e.g., it takes time to annotate new songs).

The goal of our work is to combine multiple representations of music information from acoustic and social sources to improve query-by-text music retrieval. While there is relatively little work on this exact problem, there has been significant research on the task combining multiple complementary audio representations for music classification [36, 9, 22]. Tzanetakis and Cook [36] present a number of content-based feature sets that relate to timbre, harmony and melody. They concatenate and summarize these feature sets into a single vector to represent each song. They use these music feature vectors with standard classifiers (e.g., nearest neighbor, GMM) for the task of genre classification.

Flexer et al.[9] combine information from two audio representations related to tempo and timbre (MFCCs). They use a nearest-neighbor classifier on each feature space to determine probabilities of dance-music genres. Using the naïve Bayes assumption of class-conditional independence given each feature set, they find that the product of these two probabilities improves 8-way genre classification of dance music. This approach is akin to how multiple calibrated scores are combined by our CSA algorithm (Section 4.1).

Beyond the music domain, there has been a good deal of research on combining multiple sources (or representations) of information [4, 26, 30]. Approaches can be roughly divided into *early fusion* where feature representations are merged (e.g., stacking feature vectors [36]) before classification or ranking and *late fusion* where outputs of multiple classifiers or ranking algorithms are combined (e.g., [9]). Late fusion approaches, often used in "meta-search" engines, can be further divided into *score-based* approaches that combine the (probabilistic) scores output by the classifiers, and *rank-based* approaches that combine multiple rank-orderings of (text, image, or audio) documents. Rank-based approaches (also referred to as rank aggregation) are more general since they can be implemented when scores are not available.

Manmantha et al. [23] provide an example score-based approach where they estimate a mixed Gaussian-exponential distribution over scores for each search engine. Using these distributions, they map scores to posterior probabilities (e.g., calibration) and then combine these probabilities by averaging them. This approach is related to our CSA algo-

rithm, but differs in that we use a non-parametric technique called isotonic regression [39] to calibrate our scores.

Freund et al. [10] describe the RankBoost algorithm. This rank-based approach is relatively easy to implement, has a strong theoretical foundation, and has been shown to perform well on many learning tasks [3]. We describe the RankBoost algorithm in more detail in Section 4.2.

Lastly, Lanckriet et al. [18] propose an approach that is neither early fusion nor late fusion since the multiple representations are combined and a decision boundary is learned simultaneously. They use a kernel matrix to represent each information source and learn a linear combination of these kernels that optimizes performance on a discriminative classification task. It has been shown that, for protein classification [18] and music retrieval [2] tasks, an optimal combination of heterogenous feature kernels performs better than any individual feature kernel.

3. SOURCES OF MUSIC INFORMATION

In this section, we describe three sources of music information and show how we extract four meaningful feature-based representations from them. For each representation, we derive a relevance score function, $r(s;t)$, that evaluates the relevance of song s to tag t. The song-tag relevance scores derived from representations based on the audio content are *dense*, while those resulting from social representations are considered *sparse* since the strength of association between some songs and some tags is unknown (i.e., missing). For example, less-popular songs tend to be more sparsely annotated since fewer humans have listened to these songs (i.e., the "cold start" problem). (See [32] for more detail.)

3.1 Representing Audio Content

We use the supervised multiclass labeling (SML) model, recently proposed by Turnbull et al. [34], to automatically annotate songs with tags based on audio content analysis. The SML model is parameterized by one Gaussian mixture model (GMM) distribution over an audio feature space for each tag in the vocabulary. First, the audio track of each song, s, is represented as a bag of feature vectors, $\mathcal{X} = \{\mathbf{x}_1, ..., \mathbf{x}_T\}$, where each $\mathbf{x}_i$ is a feature vector that represents a short-time segment of audio, and T depends on the length of the song. We first use the expectation maximization algorithm to learn a GMM distribution of the set of audio feature vectors $\mathcal{X}$ that describes each song. Next, we identify a set of example songs that humans have associated with a given tag. Finally, the GMMs that model these example songs are used by the mixture-hiearchies expectation maximization algorithm [37] to learn the parameters of GMM distribution that represents the tag.

Given a novel song s, the set of audio features $\mathcal{X}$ is extracted and the likelihood is evaluated using each of the tag GMMs. The result is a vector of probabilities that, when normalized, can be interpreted as the parameters of a multinomial distribution over the vocabulary of tags. Under this probabilistic setup, the relevance of song s to tag t may be written as:

$$r_{audio}(s;t) \propto p(t|\mathcal{X}), \qquad (1)$$

given a number of simplifying assumptions (see Section III.b of [34] for details.) We next describe two different audio feature representations used to produce two different bags of feature vectors for each song, $\mathcal{X}_{MFCC}$ and $\mathcal{X}_{Chroma}$.

3.1.1 MFCCs

Mel frequency cepstral coefficients (MFCCs) are a popular audio feature representation for a number of music information retrieval tasks and were incorporated into all of the top performing autotagging systems in the most recent MIREX competition [34, 22, 7]. MFCCs are loosely associated with the musical notion of timbre (e.g., "color of the music") since they are a low-dimensional representation of the spectrum[1] of a a short-time audio sample.

For each 22050 Hz-sampled monaural song in the data set, we compute the first 13 MFCCs for each half-overlapping short-time ($\sim$23 msec) segment. Over the time series of audio segments, we calculate the first and second instantaneous derivatives (referred to as *deltas*) for each MFCC. This results in about 5,000 39-dimensional MFCC+delta feature vectors per 30 seconds of audio content. We summarize an entire song by modeling the distribution of its MFCC+delta features with an 8-component GMM. We model each tag with a 16-component GMM.

3.1.2 Chroma

Chroma features [11] attempt to represent the harmonic content (e.g, keys, chords) of a short-time window of audio by computing the spectral energy present at frequencies that correspond to each of the 12 notes in a standard chromatic scale (e.g., black and white keys within one octave on a piano). We extract a 12-dimensional chroma feature every $\frac{1}{4}$ second and, as with MFCCs, model the song and tag distributions of chroma features with 8- and 16-component GMMs, respectively.

3.2 Representing Social Context

We can also summarize each song in our dataset with an *annotation vector* over a vocabulary of tags. Each real-valued element of this vector indicates the relative strength of association between the song and a tag. We propose two methods for collecting this semantic information: social tags and web-mined tags. The annotation vectors are, in general, *sparse* as most songs are annotated with only a few tags. A missing song-tag pair can arise for two reasons: either the tag is not relevant or the tag is relevant but nobody has annotated the song with it. It is also important to note that annotation vectors can be considered *noisy* observations since they do not always accurately reflect the semantic relationships between songs and tags due to the unstructured and inconsistent nature of the data collection process.

3.2.1 Social Tags

Last.fm[2] is a music discovery website that allows users to contribute *social* tags through a text box in their various audio player interfaces. By September of 2008, their 20 million monthly users had annotated 3.8 million items (songs, artists, albums or record labels) over 50 million times (a rate of 2.5 million annotations per month) using a vocabulary of 1.2 million unique free-text tags [17]. While this may seem substantial, given that Last.fm's database contains over 150 million songs by 16 million artists, these annotations account for only a small fraction of available music.

---

[1]The spectrum of an audio signal represents the various harmonic and inharmonic elements that combine to produce a particular acoustic experience.
[2]www.last.fm

For each song s in our dataset, we attempt to collect two lists of social tags from Last.fm using their public data sharing AudioScrobbler[3] website. The first list relates the *song* to a set of tags where each tag has a *tag score* that ranges from 0 (low) to 100 (high). This score is a function of both the number and diversity of users who have annotated that song with the tag and is a trade secret of Last.fm. The second list associates the *artist* with tags and aggregates the tag scores for all the songs by that artist.

The relevance score $r_{social}(s;t)$ for song s and tag t is the sum of the tag scores on the artist list and song list plus the tag score for any synonyms or wildcard matches of t on either list. For example, a song is considered to be annotated with 'down tempo' if it has instead been annotated with 'slow beat'. In addition, 'blues' matches with the tags 'delta electric blues', 'blues blues blues', and 'rhythm & blues'.

3.2.2 Web-Mined Tags

In order to extract tags from a corpus of web documents, we adapt the relevance scoring (RS) algorithm that has recently been proposed by Knees et. al. [15]. They have shown this method to be superior to algorithms based on vector space representations. To generate relevance scores, RS works as follows:

1. **Collect Document Corpus:** For each song in the music corpus, query a search engine with the song title, artist name, and album title. Collect the web documents returned by the search engine. Retain the (many-to-many) mapping M such that $M_{s,d} = 1$ if document d was found when using song s in the query, and 0 otherwise.

2. **Tag Songs:** For each tag, t;
   (a) Use t as a query string to find the set of relevant documents $\mathcal{D}_t$ from the text-corpus retrieved in Step 1. Each document $d \in \mathcal{D}_t$ will be associated with a *relevance weight*, $w_{d,t}$ (defined below).
   (b) For each song s, sum the relevance weights for all the documents $d \in \mathcal{D}_t$:
   $$r_{web}(s;t) = \sum_{d \in \mathcal{D}_t} M_{s,d} \cdot w_{d,t}.$$

We modify this algorithm in two ways. First, in [15], the relevance weight $w_{d,t}$ is inversely proportional to the rank of the relevant document. In our implementation, the relevance weight is a function of the number of times the tag appears in the document (tag-frequency), the number of documents with the tag (document frequency), the number of total words in the document, the number of words or documents in the corpus, etc. Specifically, the relevance weights are determined by the MySQL match function.[4]

The second modification is that we use *site-specific* queries when creating our corpus of web documents (Step 1). That is, Knees et. al. [15] collect the top 100 documents returned by Google when given queries of the form:

- "<artist name>" music
- "<artist name>" "<album name>" music review
- "<artist name> " "<song name>" music review

for each song in the data set. We use site-specific queries by appending the substring 'site:<music site url>' to the three query templates, where <music site url> is the url

for a music website that is known to have high quality information about songs, albums or artists. These sites include allmusic.com, amazon.com, bbc.co.uk, billboard.com, epinions.com, musicomh.com, pandora.com, pitchforkmedia.com, rollingstone.com, and wikipedia.org. For these 10 music sites and one non-site-specific query, we collect and store the top 10 pages returned by the Google search engine. This results in a maximum of 33 queries and a maximum of 330 pages per song. On average, we are only able to collect 150 webpages per song since some of the less popular songs are not well represented by these music sites.

4. COMBINING MULTIPLE SOURCES OF MUSIC INFORMATION

Given a query tag t, our goal is to find a single rank ordering of songs based on their relevance to tag t. We present three algorithms that combine the multiple sources of music information to produce such a ranking. The first two algorithms, calibrated score averaging (CSA) and RankBoost, directly combine the individual rank orderings provided by each of our four music information representations. For the two audio content representations, we use the SML algorithm presented in Section 3.1 to produce these rank orderings. For the two social context sources, the rank orderings are constructed by using social tag scores or web relevance scores as described in Section 3.2.1. The third algorithm, Kernel Combination SVM (KC-SVM), uses convex optimization to combine a set of kernel matrices derived from each of the four music representations.

All three algorithms are considered *supervised* since they use labeled training data to learn how best to combine music representations. That is, we have a ground truth of *binary judgement labels* for each song-tag pair (e.g., 1 if relevant, 0 otherwise) which we denote as $l(s;t)$ for tag t and song s.

4.1 Calibrated Score Averaging (CSA)

Each representation of a data source produces a relevance score function, $r(s;t)$, indicating how relevant tag t is for describing song s. Using training data, we can learn a function $g(\cdot)$ that *calibrates* scores such that $g(r(s;t)) \approx P(t|r(s;t))$. This allows us to compare data sources in terms of calibrated posterior probabilities rather than incomparable scores.

As described by Zadrozny and Elkan [39], we use isotonic regression [28] to estimate a function g for each representation. More specifically, we use the pair-adjacent violators (PAV) algorithm [1, 6] to learn the stepwise-constant isotonic (i.e., non-decreasing) function that produces the best fit in terms of minimum mean-squared error. To learn this function g for tag t, we start with a (low to high score) rank-ordered training set $s^{(1)}, s^{(2)}, ... s^{(N)}$ of N songs where $r(s^{(i-1)};t) < r(s^{(i)};t)$. We initialize g to be equal to the sequence of binary training labels (e.g., $g(r(s^{(i)};t)) = l(s^{(i)};t)$. If the training data is perfectly ordered, then g is isotonic and we are done. Otherwise, there exists an i where there is a *pair-adjacent violation* such that $g(r(s^{(i-1)};t)) > g(r(s^{(i)};t))$. To remedy this violation, we update $g(r(s^{(i-1)};t))$ and $g(r(s^{(i)};t))$ so that they both become $[g(r(s^{(i-1)};t)) + g(r(s^{(i)};t))]/2$. We repeat this process until we have eliminated every pair-adjacent violation. At this point, g is isotonic and we combine it with the corresponding scores $r(s^{(i)};t)$ to produce a stepwise function that maps scores to approximate probabilities.

For example, if we have 7 songs with relevance scores equal to $(1, 2, 4, 5, 6, 7, 9)$ and ground truth labels equal to $(0, 1, 0, 1, 1, 0, 1)$, then $g(r) = 0$ for $r < 2$, $g(r) = 1/2$ for $2 \le r < 6$, $g(r) = 2/3$ for $6 \le r < 9$, and $g(r) = 1$ for $9 \le r$. We use Dümbgen's [6] linear-time $O(N)$ implementation of the PAV algorithm for our experiments in Section 5.

Recall that the social context data sources are sparse: many song-tag scores are *missing*. This may mean that the tag is actually relevant to the song but that no data is found to connect them (e.g., no humans bothered to annotate the song with the tag). The most straightforward approach to dealing with this problem is to estimate $P(t|r(s;t) = \emptyset)$ with the prior probability $P(t)$. However, we find empirically that a missing song-tag score often suggests that the tag is truly not relevant. Instead, we use the training data to estimate:

$$P(t|r(s;t) = \emptyset) = \frac{\#(\text{relevant songs with } r(s;t) = \emptyset)}{\#(\text{songs with } r(s;t) = \emptyset)}. \quad (2)$$

Once we have learned a calibration function for each representation, we convert the vector of scores for a test set song to a vector of approximate posterior probabilities. We can combine these posterior probabilities by using the arithmetic average, geometric average, harmonic average, median, minimum, maximum, and other variants [14]. We also consider variants that ignore missing scores such that we combine only the non-missing calibrated scores. Of all these variants, we find that the arithmetic average produces the best empirical tag-based retrieval results.

4.2 RankBoost

In a framework that is conceptually similar to the Adaboost algorithm, the RankBoost algorithm produces a *strong* ranking function H that is a weighted combination of *weak* ranking functions h_t [10]. Each weak ranking function is defined by the representation, a threshold, and a default value for missing data. For a given song, the weak ranking function is an indicator function that outputs 1 if the score for the associated representation is greater than the threshold or if the score is missing and the default value is set to 1. Otherwise, it outputs 0. During training, RankBoost iteratively builds an ensemble of weak ranking functions and associated weights. At each iteration, the algorithm selects the weak learner (and associated weight) that maximally reduces the *rank loss* of a training data set given the current ensemble. We use the implementation of RankBoost shown in Figures 2 and 3 of [10].[5]

4.3 Kernel Combination SVM (KC-SVM)

In contrast to the two previous methods of directly combining the outputs of each individual system, we could combine sources at the feature level and produce a single ranking. Lanckriet et al. [18] propose a linear combination of M different kernels that each encode different data features:

$$\mathbf{K} = \sum_{m=1}^{M} \mu_m \mathbf{K}_m, \quad \text{where } \mu_m > 0 \text{ and } \mathbf{K}_m \succeq 0 \; \forall m. \quad (3)$$

Since each individual kernel matrix, $\mathbf{K}_m$, is positive semi-definite, their positively-weighted sum, $\mathbf{K}$, is also a valid, positive semi-definite kernel [29].

---

[5] We also enforce the positive cumulative weight constraint for the RankBoost algorithm as suggested at the end of Section 4 in [10].

The individual kernel matrices $\mathbf{K}_m$ represent similarities between all songs in the data set. One kernel matrix is derived from each of the data representations described in Section 3. The kernels are normalized by projection onto the unit sphere [29].

For the two audio content features (MFCC and Chroma), we model the set of audio features that represent a song, $\mathcal{X} = \{\mathbf{x}_1, ..., \mathbf{x}_T\}$, with a GMM distribution, $p(\mathbf{x}; \theta)$, where the GMM is specified by parameters θ. We then compute the entries of a probability product kernel (PPK) [12] by comparing the parameters of the GMM distributions that model each song. We have found experimentally that the PPK performs better than other kernels that attempt to capture similarities between distributions (such as the Kullback-Leibler divergence kernel space [25]) and that the PPK is also easier and more elegant to compute. The PPK between songs i and j is computed as:

$$\mathbf{K}_{\text{audio}}(i, j) = \int p(\mathbf{x}; \theta_i)^\rho p(\mathbf{x}; \theta_j)^\rho d\mathbf{x}, \quad (4)$$

where $\rho > 0$. When $\rho = 1/2$, the PPK corresponds to the Bhattacharyya distance between two distributions. This case has a geometric interpretation similar to the inner-product between two vectors. That is, the PPK measures the cosine of the "angle" between the two distributions. Another advantage of the PPK is that closed-form solutions for GMMs and other parametric distributions are available [12], whereas this is not the case for Kullback-Leibler divergence.

For each of the social context features, we compute a radial basis function (RBF) kernel [29] with entries:

$$\mathbf{K}_{\text{social}}(i, j) = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}), \quad (5)$$

where $\mathbf{K}(i, j)$ represents the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$, the annotation vectors for songs i and j, described in Section 3.2. The hyper-parameter σ is estimated using cross validation. If any element of an annotation vector (i.e., song-tag pair) is missing, we set that element to zero. If a song has not been annotated with any tags, we assign that song the average annotation vector (i.e., the estimated vector of prior probabilities of each tag in the vocabulary).

For each tag t and corresponding class-label vector, $\mathbf{y}$, ($y_i = +1$ if $l(i; t) = 1$ and $y_i = -1$ otherwise), the primal problem for the single-kernel SVM is to find the decision boundary with maximum margin separating the two classes. The optimum value of the dual problem is inversely proportional to the margin separating the classes and is convex in the kernel, $\mathbf{K}$. The kernel combination SVM problem requires learning the set of weights, μ, that combine the feature kernels, $\mathbf{K}_m$, into the "optimum" kernel, while also solving the standard SVM optimization. The optimum $\mathbf{K}$ can be learned by minimizing the function that optimizes the dual (thereby maximizing the margin) with respect to the kernel weights, μ:

$$\min_{\mu} \left\{ \max_{0 \le \alpha \le C, \alpha^T \mathbf{y} = 0} 2\alpha^T \mathbf{e} - \alpha^T \text{diag}(\mathbf{y}) \mathbf{K} \text{diag}(\mathbf{y}) \alpha \right\}$$
$$\text{subject to: } \mu^T \mathbf{e} = 1 \quad (6)$$
$$\mu_m \ge 0 \quad \forall m = 1, ..., M,$$

where $\mathbf{K} = \sum_{m=1}^{M} \mu_m \mathbf{K}_m$ and $\mathbf{e}$ is an n-vector of ones such that $\mu^T \mathbf{e} = 1$ constrains the weights μ to sum to one. C is a hyper-parameter that limits violations of the margin (in

Table 1: Tag-based music search examples for Calibrated Score Averaging (CSA). The top ranked songs for each of the first 5 folds (during 10-fold cross validation) for 12 representative tags. In each box, the tag is listed first (in bold). The second row is the the area under the ROC curve (AUC) and the mean average precision (MAP) for the tag (averaged over 10-fold cross validation). Each artist-song pair is the top ranked song for the tag and is follow by "(m)" if it is considered misclassified, according to the ground truth. Note that some of the misclassified songs may actually be representative of the tag.

Synthesized Song Texture	**Acoustic Song Texture**	**Electric Song Texture**
0.80 / 0.71	0.73 / 0.76	0.76 / 0.73
Tricky - *Christiansands* (**m**)	Robert Johnson - *Sweet Home Chicago*	Portishead - *All Mine*
Propellerheads - *Take California*	Neil Young - *Western Hero*	Tom Paul - *A little part of me* (**m**)
Aphex Twin - *Come to Daddy*	Cat Power - *He War* (**m**)	Spiritualized - *Stop Your Crying* (**m**)
New Order - *Blue Monday*	John Lennon - *Imagine*	Muddy Waters - *Mannish Boy*
Massive Attack - *Risingson*	Ani DiFranco - *Crime for Crime*	Massive Attack - *Risingson* (**m**)
Female Vocals	**Male Vocals**	**Distorted Electric Guitar**
0.95 / 0.90	0.71 / 0.82	0.78 / 0.42
Billie Holiday - *God Bless The Child*	The Who - *Bargain*	The Who - *Bargain* (**m**)
Andrews Sisters - *Boogie Woogie Bugle Boy*	Bush - *Comedown*	Bush - *Comedown*
Alanis Morissette - *Thank U*	AC/DC - *Dirty Deeds Done Dirt Cheap*	The Smithereens - *Behind the Wall of Sleep*
Shakira - *The One*	Bobby Brown - *My Prerogative*	Adverts - *Gary Gilmore's Eys* (**m**)
Alicia Keys - *Fallin'*	Nine Inch Nails - *Head Like a Hole*	Sonic Youth - *Teen Age Riot*
Jazz	**Blues**	**Soft Rock**
0.96 / 0.82	0.84 / 0.45	0.77 / 0.37
Billie Holiday - *God Bless The Child*	B.B. King - *Sweet Little Angel*	Steely Dan - *Rikki Don't Lose That #* (**m**)
Thelonious Monk - *Epistrophy*	Canned Heat - *On the Road Again* (**m**)	Carpenters - *Rainy Days and Mondays* (**m**)
Lambert, Hendricks & Ross-*Gimme That Wine*	Cream - *Tales of Brave Ulysses* (**m**)	Cat Power - *He War* (**m**)
Stan Getz - *Corcovado*	Muddy Waters - *Mannish Boy*	Carly Simon - *You're So Vain*
Norah Jones - *Don't Know Why*	Chuck Berry - *Roll Over Beethoven* (**m**)	Bread - *If*
Calming	**Aggressive**	**Happy**
0.81 / 0.66	0.84 / 0.51	0.67 / 0.49
Crosby, Stills & Nash - *Guinnevere*	Pizzle - *What's Wrong with my foot?*	The Turtles - *Elenore*
Carpenters - *Rainy Days and Mondays*	Rage Against the Machine-*Maggie's Farm*	Jane's Addiction - *Been Caught Stealing*
Cowboy Junkies - *Postcard Blues*	Aphex Twin - *Come to Daddy*	Stevie Wonder - *For Once in My Life*
Tim Hardin - *Don't Make Promises*	Black Flag - *Six Pack*	New Order - *Blue Monday* (**m**)
Norah Jones - *Don't Know Why*	Nine Inch Nails - *Head Like a Hole*	Altered Images-*Don't Talk to Me About Love*

practice, we find it necessary to set C independently for both classes). The non-zero entries of the solution vector α indicate the support vectors that define the decision boundary.

The problem in Equation 6 can be formalized as a quadratically-constrained quadratic program (for full details, see [18]). The solution returns a linear decision function that defines the distance of a new song, s_z, from the hyperplane boundary between the positive and negative classes (i.e., the relevance of s_z to tag t):

$$r_{SVM}(s_z; t) = \sum_{i=1}^{n} \alpha_i \mathbf{K}(i, z) + b, \qquad (7)$$

where b is the offset of the decision boundary from the origin. SVM classifiers use the sign of the decision function to indicate on which side of the decision boundary a given data point lies and thus classify it as belonging to the class or not (e.g., decides whether tag t applies to the song or not). More generally, the distance from the decision boundary may be used to rank data points by their relevance to the class (e.g., rank songs by their relevance to tag t) [22].

5. SEMANTIC MUSIC RETRIEVAL EXPERIMENTS

In this section, we apply the three algorithms presented in the previous section to the task of semantic (i.e., tag-based) music retrieval. We experiment on the CAL-500 data set [33]: 500 songs by 500 unique artists each annotated by a minimum of 3 individuals using a 174-tag vocabulary[6]. A song is considered to be annotated with a tag if 80% of the

human annotators agree that the tag is relevant. For the experiments reported here, we consider a subset of 72 tags by requiring that each tag be associated with at least 20 songs and removing some tags that we deemed to be redundant or overly subjective. These tags represent genres, instruments, vocal characteristics, song usages, and other musical characteristics. The CAL-500 data is a reasonable ground truth since it is complete and redundant (i.e., multiple individuals explicitly evaluated the relevance of every tag for each song). However, any such ground truth will be subjective due to the nature of the music annotation task [17].

Given a tag (e.g., "jazz"), the goal is to rank all songs by their relevance to the query tag (e.g. jazz songs at the top). In most cases, we can *directly* rank songs using the scores associated with the song-tag pairs for a tag. However, when using the SVM framework, we learn a decision boundary for each tag (e.g., a boundary between jazz / not jazz). We then rank all test songs by their distance (positive or negative) from the decision boundary, using Equation 7. The songs which most strongly embody the query tag should have a large positive distance from the boundary. Reformulations of the single-kernel SVM exist which optimize for ranking rather than classification [13] but the distance from the boundary provides a monotonic ranking of the entire test set which is suitable for this semantic retrieval task.

We compare the direct and SVM ranking results to the human-annotated labels provided in the CAL-500 dataset and evaluate the rankings using two metrics: the area under the receiver operating characteristic (ROC) curve (denoted AUC) and the mean average precision (MAP) [24]. The ROC curve compares the rate of correct detections to false alarms at each point in the ranking. A perfect ranking (i.e.,

Table 2: The number of tags for which each musical feature representation was the best at predicting tag relevance.

Representation	Direct Ranking	SVM Ranking
MFCC	51	42
Chroma	0	0
Social Tags	9	21
Web-Mined Tags	12	9

Table 3: Evaluation of semantic music retrieval. All reported AUC and MAP values are averages over a vocabulary of 72 tags, each of which has been averaged over 10-fold cross validation. The top four rows represent individual data source performance. *Single Source Oracle* (SSO) picks the best single source for retrieval given a tag, based on test set performance. The final three approaches combine information from all four data sources using the algorithms described in Section 4. Note the performance differences between single source and multiple source algorithms are significant (one-tailed, paired t-test over the vocabulary with $\alpha = 0.05$). However, the differences between between SSO, CSA, RB and KC are not statistically significant.

Representation	Direct Ranking		SVM Ranking	
	AUC	MAP	AUC	MAP
MFCC	0.731	0.473	0.722	0.467
Chroma	0.527	0.299	0.604	0.359
Social Tags	0.623	0.431	0.708	0.477
Web-Mined Tags	0.625	0.413	0.699	0.477
Single Source Oracle (SSO)	0.756	0.530	0.753	0.534
Calib. Score Avg. (CSA)	**0.763**	**0.538**	·	·
RankBoost (RB)	0.760	0.531	·	·
Kernel Combo (KC)	·	·	0.756	0.529

all the relevant songs at the top) results in an AUC equal to one. Ranking songs randomly, we expect the AUC to be 0.5. Mean average precision (MAP) is found by moving down the ranked list of test songs and averaging the precisions (the ratio of corlrectly-labeled songs to the length of the list) at every point where we correctly identify a new song.

5.1 Single Data Source Results

For each representation (MFCC, Chroma, Social Tags, Web-Mined Tags), we evaluate the direct ranking and the single-kernel SVM ranking. For SVM ranking, we construct a kernel and use it to train a one-vs-all SVM classifier for each tag where the negative examples are all songs not labeled with that tag. We train SVMs using 400 songs, find the optimum regularization parameter, C, using a validation set of 50 songs and use this final model to report results on a test set of 50 songs. The performance of each kernel, averaged using 10-fold cross validation for each tag (such that each song appears in the test set exactly once), and then averaged over the set of 72 tags, is shown on the rightmost columns of Table 3. To be consistent with SVM ranking, we use 10-fold cross validation for direct ranking and average evaluation metrics over each fold, and then over each tag. These results appear center columns of Table 3.

Table 3 also shows the results that could be achieved if the single best data source for each tag were known in ad-

vance and used to rank the songs. This *single source oracle* (SSO) can be considered an empirical upper bound since it selects the best data source for each tag based on *test set* performance and should be the minimum target for our combination algorithms. For example, the best representation for the tag "jazz" is web-mined tags while the best representation for "hip hop" is MFCC. The number of tags for which each data source was most useful is shown in Table 2.

Table 3 demonstrates that all four data sources produce rankings that are significantly better than random, and that MFCC produces the significantly best individual rankings[7]. Table 2 indicates that MFCC is the single best data source for about 60% of the tags while the social context-based features are best for the other 40%. While Chroma produces the the worst overall performance, the AUC for SVM ranking is 0.6 and is significantly better than random.

5.2 Multiple Data Source Results

Using the three algorithms described in Section 4, we can combine information from the four data sources to significantly enhance our tag-based music retrieval system. These results are shown in the bottom three rows of Table 3. We also produce qualitative search results in Table 1 to provide context for our evaluation metrics. The best performance is achieved using CSA, though the performance is neither significantly better than RankBoost nor KC-SVM. In addition, CSA is not significantly better than the SSO.

As previously noted, the Chroma representation produces the worst single source results (AUC of 0.527 for direct ranking is not much better than random). If we omit the Chroma representation, the AUC for CSA increases to 0.767, a significant difference over SSO. RankBoost and KC-SVM also show slight improvements, but not so much that they are significantly better than SSO. While this may seem like positive observation for CSA, it suggests that CSA is less robust to additional noisy representations.

If we examine the 8 single and 3 multiple data source algorithms when considering each tag individually, 47 of the 72 tags are improved by one of the multiple data source algorithms. Specifically, CSA performs best for 16 tags, Rank-Boost performs best for 11, and Kernel Combination performs best for 20 tags. This suggests that each algorithm is individually useful and that combination of their outputs could further enhance semantic music retrieval.

6. DISCUSSION

In this paper, we have described three sources of music information and show that they are each individually useful for semantic music retrieval. Furthermore, we have explored three algorithms that further improve retrieval performance by effectively combining these information sources. While the CSA algorithm produced the best overall results, our experiments suggest that it is more negatively affected by noisy information than KC-SVM or RankBoost. CSA and RankBoost are generally easier to code and take less time to train than KC-SVM since they do not involve convex optimization. CSA has the added benefit of being easy to "tune" if we are interested in designing a user interface in which the user decides how much relative weight to place on each data source.

---

[7]Unless otherwise noted, all statistical hypothesis tests between two algorithms are one-tailed, paired t-test over the vocabulary (sample size = 72) with $\alpha = 0.05$

Future work will involve incorporating additional audio representations such as those that relate to melody and rhythm [36]. In addition, we are currently collecting a larger music corpus ($+10,000$ songs) accompanied by a larger vocabulary containing thousands of tags. While the CAL-500 data set used in the paper may seem small by comparison, it will continue to be useful for training and evaluation since it difficult to collect ground truth annotations for even a modest number of song-tag pairs.

7. REFERENCES

[1] M. Ayer, H.D. Brunk, G.M. Ewing, W.T. Reid, and E. Silverman. An empirical distribution function for sampling with incomplete information. *Annals of Mathematical Statistics*, 1955.

[2] L. Barrington, M. Yazdani, D. Turnbull, and G. Lanckriet. Combining feature kernels for semantic music retrieval. *ISMIR*, 2008.

[3] C. Cortes and M. Mohri. AUC optimization vs. error rate minimization. In *NIPS*. MIT Press, 2004.

[4] W.B. Croft. Combining approaches to information retrieval. *Advances in Information Retrieval*, 2000.

[5] S. J. Downie. The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 2008.

[6] Lutz Dümbgen. Isotonic regression software (Matlab), http://staff.unibe.ch/duembgen/software/#Isotone.

[7] D. Eck, P. Lamere, T. Bertin-Mahieux, and S. Green. Automatic generation of social tags for music recommendation. In *NIPS*, 2007.

[8] S. Essid, G. Richard, and B. David. Inferring efficient hierarchical taxonomies for music information retrieval tasks: Application to music instruments. *ISMIR*, 2005.

[9] A. Flexer, F. Gouyon, S. Dixon, and G. Widmer. Probabilistic combination of features for music classification. *ISMIR*, 2006.

[10] Y. Freund, R. Iyer, R. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *JMLR*, 4:933–969, 2003.

[11] M. Goto. A chorus selection detection method for musical audio singals and its application to a music listening station. *IEEE TASLP*, 14-5, 2006.

[12] T. Jebara, R. Kondor, and A. Howard. Probability product kernels. *Journal of Machine Learning Research*, 5:819–844, 2004.

[13] T. Joachims. Optimizing search engines using clickthrough data. *ACM Conference on Knowledge Discovery and Data Mining*, 2002.

[14] J. Kittler, M. Hatef, R.P.W. Duin, and J. Matas. On combining classifiers. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[15] P. Knees, T. Pohle, M. Schedl, D. Schnitzer, and K. Seyerlehner. A Document-centered Approach to a Natural Language Music Search Engine. *ECIR*, 2008.

[16] P. Knees, T. Pohle, M. Schedl, and G. Widmer. A music search engine built upon audio-based and web-based similarity measures. In *ACM SIGIR*, 2007.

[17] P. Lamere and E. Pampalk. Social tags and music information retrieval. ISMIR Tutorial, 2008.

[18] G.R.G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M.I. Jordan. Learning the kernel matrix with semi-definite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[19] M. Levy and M. Sandler. A semantic space for music derived from social tags. In *ISMIR*, 2007.

[20] T. Li and G. Tzanetakis. Factors in automatic musical genre classification of audio signals. *IEEE WASPAA*, 2003.

[21] M. Mandel and D. Ellis. A web-based game for collecting music metadata. In *ISMIR*, 2007.

[22] M. Mandel and D. Ellis. Multiple-instance learning for music information retrieval. In *ISMIR*, 2008.

[23] R. Manmatha, T. Rath, and F. Feng. Modeling score distributions for combining the outputs of search engines. In *ACM SIGIR*, New York, NY, USA, 2001. ACM.

[24] C.D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[25] P.J. Moreno, P.P. Ho, and N. Vasconcelos. A Kullback-Leibler divergence based kernel for SVM classification in multimedia applications. *NIPS*, 2004.

[26] E. S. Parris, M. J. Carey, and H. Lloyd-Thomas. Feature fusion for music detection. In *EUROSPEECH*, pages 2191–2194, 1999.

[27] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 1999.

[28] T. Robertson, F. Wright, and R. Dykstra. *Order Restricted Statistical Inference*. Wiley and Sons, 1988.

[29] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, USA, 2004.

[30] C. Snoek, M. Worring, and A. Smeulders. Early versus late fusion in semantic video analysis. In *ACM Multimedia*, 2005.

[31] M. Sordo, C. Lauier, and O. Celma. Annotating music collections: How content-based similarity helps to propagate labels. In *ISMIR*, 2007.

[32] D. Turnbull, L. Barrington, and G. Lanckriet. Five approaches to collecting tags for music. In *ISMIR*, 2008.

[33] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Towards musical query- by- semantic-description using the CAL500 data set. In *ACM SIGIR*, 2007.

[34] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE TASLP*, 2008.

[35] D. Turnbull, R. Liu, L. Barrington, D. Torres, and G Lanckriet. Using games to collect semantic information about music. In *ISMIR*, 2007.

[36] G. Tzanetakis and P. R. Cook. Musical genre classification of audio signals. *IEEE Transaction on Speech and Audio Processing*, 10(5):293–302, 7 2002.

[37] N. Vasconcelos. Image indexing with mixture hierarchies. *IEEE CVPR*, pages 3–10, 2001.

[38] B. Whitman and D. Ellis. Automatic record reviews. In *ISMIR*, 2004.

[39] B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *KDD*. ACM, 2002.