Modeling music and words using a multi-class naïve Bayes approach

Douglas Turnbull
UC San Diego
La Jolla, CA 92093
dturnbul@cs.ucsd.edu

Luke Barrington
UC San Diego
La Jolla, CA 92093
lbarring@ucsd.edu

Gert Lanckriet
UC San Diego
La Jolla, CA 92093
gert@ece.ucsd.edu

Abstract

We propose a *query-by-text* system for modeling a heterogeneous data set of music and words. We quantitatively show that our system can both *annotate* a novel song with semantically meaningful words and *retrieve* relevant unlabeled songs from a database given a text-based query. We explain two feature extraction methods useful for summarizing the audio content of a song. We describe a supervised multi-class naïve Bayes model and compare two parameter estimation techniques. Our approach is influenced by recent computer vision research on the related tasks of image annotation and retrieval.

Keywords: music annotation, music retrieval, query-by-text, heterogeneous data

1. Music Annotation and Retrieval

Music is a form of communication that can represent human emotions, personal style, geographic origins, spiritual foundations, social conditions, and other aspects of humanity. Listeners naturally use words in an attempt describe what they hear, though two listeners may use drastically different words when describing the same piece of music. However, words related to some aspects of the audio content, such as instrumentation and genre, may be agreed upon by a majority of listeners. This agreement suggests that it is possible to create a computer audition system that can learn the relationship between audio content and words. By jointly modeling these two representations, we create a model that can be used to both *retrieve* sounds given a text-based query and to *annotate* a sound with text given the audio content.

A central goal of the music information retrieval community is to create systems that efficiently store and retrieve songs from large databases of musical content [1]. The most common way to store and retrieve music uses metadata such as the name of the composer or artist, the name of the song or the release date of the album. We consider a more general definition of musical metadata as any non-acoustic representation of a song. This includes genre and instrument labels, song reviews, ratings according bipolar adjectives (e.g., happy/sad), and purchase sales records. These representations can be used as input to collaborative filtering systems that help users search for music. The drawback of all these systems is that they require a novel song to be *manually* annotated before it can be retrieved.

Another approach, called *query-by-similarity*, takes an audio-based query and measures the similarity between the query and all of the songs in a database [1]. One drawback of query-by-similarity is that it requires a user to have a useful audio exemplar in order to specify a query. For cases in which no such exemplar is available, researchers have developed *query-by-humming* [2], *-beatboxing* [3], and *-tapping* [4]. However, it can be hard, especially for an untrained user, to emulate the tempo, pitch, melody, and timbre well enough to make these systems viable [2]. A natural alternative is to describe music using words. A good deal of research has focused on content-based classification of music by genre [5], emotion [6], and instrumentation [7]. These classification systems effectively 'annotate' music with class labels (e.g., 'blues', 'sad', 'guitar'). The assumption of a predefined taxonomy and the explicit labeling of songs into classes can give rise to a number of problems [8] due to the fact that music is inherently subjective.

We propose a content-based *query-by-text* music retrieval system that learns a relationship between acoustic features and words using a heterogeneous data set of songs and song reviews. Our goal is to create a more general system that directly models the relationship between audio content and a vocabulary that is less constrained than existing content-based classification systems. The query-by-text paradigm has been largely influenced by work on the similar task of image annotation. We specifically adapt a *supervised multi-class naïve Bayes* [9] model since it has performed well on the task of image annotation. This approach views semantic annotation as one M-class problem rather than M binary one-vs-all problems where M is the number of words in a predefined vocabulary. A comparative summary of alternative supervised one-vs-all [10] and unsupervised [11, 12] models for image annotation is presented in [9].

Despite interest within the computer vision community, there has been relatively little work on developing 'query-by-text' for audio (and specifically music) data. One exception is the work of Whitman [13, 14, 15]. Our approach differs from his in number of ways. First, he uses a set of web-documents associated with an artist whereas we use song reviews created by experts. Second, he takes a one-vs-all approach and learns a discriminative classifier (a support vector machine or a regularized least-squares classifier) for each term in the vocabulary. We use a generative multi-class approach by estimating a probability distribution over a feature space for each term in our vocabulary. An advantage of our approach is that, for annotation, our model outputs a natural ranking of words [9]. Other query-by-text audition systems [16, 17] have been developed for annotation and retrieval of sound effects.

2. Feature Extraction

In order to model the relationship between songs and words, each song is represented by both the words extracted from the text review and the audio features extracted from the acoustic waveform.

2.1. Text Feature Extraction

A song review often contains semantic information about the audio content (e.g., genre, instrumentation, emotion, style, rhythm) of a song. While much of this information can only be extracted though sentence- or document-level comprehension, modeling these high-level aspects of prose requires a complex language model. Instead, we focus on word-level comprehension using the *bag-of-words* representation. We reduce a song review to the set of words $\mathcal{W}$ that are found in both the review and our musical vocabulary $\mathcal{V}$. An example of a song review and the associated bag-of-words representation can be found in the first two columns of Table 1.

Our musical vocabulary consists of 317 musically informative words that the authors have hand picked from the list of the 1200 most common words found in a corpus of song reviews. “Musically informative” means that the word may describe something about the audio content, as opposed to words whose meaning is historical, cultural, syntactical etc. We do not include common stop words (‘the’, ‘into’, ‘a’), vague words (‘meaningful’, ‘across’), or general words (‘song’, ‘genre’). In addition, we preprocess the text with a custom stemming algorithm that alters suffixes so that some words, such as ‘guitar’ and ‘guitars’, are considered identical, while others, such as ‘blue’ and ‘blues’, remain distinct.

2.2. Audio Feature Extraction

Our musical data set consists of MP3 audio files which we convert to single channel audio data with a sampling rate of 22,050Hz. We examine two feature extraction techniques that have been useful for classifying music by genre [5].

2.2.1. Dynamic Mel-Frequency Cepstral Coefficients

Mel-Frequency Cepstral Coefficients (MFCCs) describe the spectral shape of a short-time audio frame in a concise and perceptually meaningful way and are popular features for speech recognition and music classification (e.g., [18, 19, 16]). We calculate 13 MFCC coefficients for each short-time frame of 512 samples (23ms) of audio.

In an attempt to capture musically-relevant details about changes between frames (e.g., beat onsets, rhythmic pulse, pitch transitions), we collect a series of MFCC vectors and use them to calculate *dynamic* MFCC (dMFCC) features

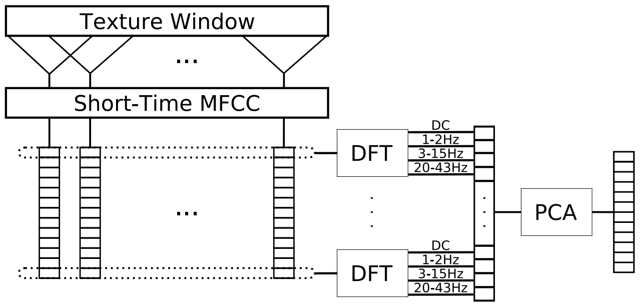

Figure 1. dMFCC feature extraction schematic

(see Figure 1). We consider a *texture window* of 16640 samples (755ms) comprised of 64 half-overlapping short-time frames. For each of the 13 MFCCs, we take a discrete Fourier transform (DFT) over the texture window of 64 points, normalize by the DC value (to remove the effect of volume) and summarize the resulting spectrum by integrating across 4 bins: (unnormalized) DC, 1-2Hz, 3-15Hz and 20-43Hz. The resulting 52 features (4 features for each of the 13 MFCCs) describe a texture window. We further reduce the dimensionality by performing Principal Components Analysis (PCA) [20] in the 52-dimensional vector space to find a projection into a 12-dimensional subspace. These 12 principal components account for 99% of the training data variance.

2.2.2. Auditory Filter-bank Temporal Envelope

The auditory filter-bank temporal envelope (AFTE) features extract information about the temporal and spectral characteristics of music (see Figure 2) [5]. In our implementation, 743ms analysis windows of a sound waveform are passed through a biologically-inspired 18 channel gammatone filter-bank [21]. We examine the positive temporal envelope of these gammatone filter responses by rectifying (squaring) each time series. We retain only the low-frequency, slowly-modulating envelope by taking the absolute value of the DFT of the rectified, Hamming-windowed signal and ignoring all of spectral components above 1kHz. To summarize the spectrum of this temporal envelope in a concise form that still retains much of its analytical capacity, we look at the data in 4 chunks of the spectrum; DC, 3-15Hz, 20-150Hz and 150-100Hz. With 18 gammatone filters, this results in a total of 72 features describing each 743ms analysis window. Again we use PCA for dimensionality reduction and represent each feature vector with 12 coefficients. This projection accounts for 95% of the variance in the training set.

3. Modeling Music and Words

Consider a vocabulary $\mathcal{V}$ consisting of M unique words. Each word $w_i \in \mathcal{V}$ may be a unigram, such as ‘happy’ or ‘blues’, or a bigram, such as ‘electric;guitar’ or ‘bob;dylan’. The goal in annotation is to find a set $\mathcal{W} = \{w_1, ..., w_A\}$ of A semantically meaningful words that describe a query song s_q. Retrieval involves rank ordering a set of R songs

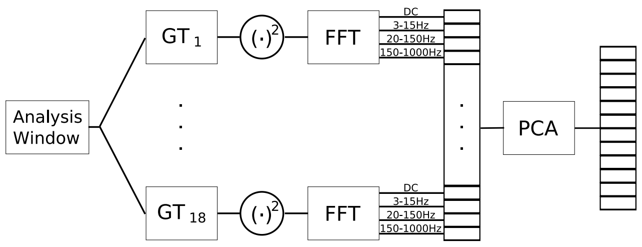

Figure 2. AFTE feature extraction schematic

$\mathcal{S} = \{s_1, ..., s_R\}$ given a query $\mathcal{W}_q$. It will be convenient to represent each annotation $\mathcal{W}$ as a binary vector $\mathbf{y} = \{y_1, ..., y_M\}$ where $y_i = 1$ if $w_i \in \mathcal{W}$, and 0 otherwise. We represent a song s as a set $\mathcal{X} = \{\mathbf{x_1}, ..., \mathbf{x_T}\}$ of T real-valued feature vectors where each vector, $\mathbf{x_t}$, is extracted from a short segment (e.g., 3/4 seconds) of the audio content and T depends on the length of the song. Our data set $\mathcal{D}$ will then be represented as a set $\{(\mathcal{X}_1, \mathbf{y}_1), ..., (\mathcal{X}_D, \mathbf{y}_D)\}$.

Annotation can be thought of as a multi-class classification problem in which each word $w_i \in \mathcal{V}$ represents a class. Our approach involves modeling a class-conditional distribution $P(\mathbf{x}|i), i \in \{1, ..., M\}$ for each word $w_i \in \mathcal{V}$. Given a query song represented by $\mathcal{X} = \{\mathbf{x_1}, ..., \mathbf{x_T}\}$, the Bayes decision rule for selecting the individual word with the minimum probability of error is given by:

$$i^* = \arg\max_i P(i|\mathcal{X}_q) = \arg\max_i \frac{P(\mathcal{X}_q|i)P(i)}{P(\mathcal{X}_q)},$$

where $P(i)$ is the prior probability that word w_i will appear in an annotation. If we assume that $\mathbf{x}_a$ and $\mathbf{x}_b$ ($\forall a, b \leq T, a \neq b$) are conditionally independent given word w_i, then

$$i^* = \arg\max_i [\prod_{t=1}^{T} P(\mathbf{x_t}|i)] \cdot P(i). \qquad (1)$$

We assume a uniform prior ($P(i) = 1/M$ for $i = 1, .., M$) since the T factors in the product dominate the word prior. Taking logarithms results in our final *annotation* equation:

$$i^* = \arg\max_i \sum_{t=1}^{T} \log P(\mathbf{x_t}|i), \qquad (2)$$

While the naïve Bayes assumption introduced in (1) is unrealistic, modeling the interaction between feature vectors may be infeasible due to computational complexity and data sparsity. Computing (2) for each word creates an ordering for all words in the vocabulary. To annotate a song, we select the A words that individually maximize this equation.

For retrieval, we want to rank all songs in a test set based on their conditional probability given a single-word query w_q. We find empirically that using the posterior $P(\mathcal{X}|q)$ always returns the same ranking under every trained word model since some songs are much more likely than others. The first reason for this is that longer songs (with more feature vectors) have lower log likelihoods resulting from the sum of additional log probability terms. It has been argued that the underestimation of the log likelihood is due to the poor conditional independence assumption in (1) between the audio feature vectors [22]. The standard solution is to calculate the *average* log posterior for each track (where T is proportional to the length of the song):

$$\mathcal{X}^* = \arg\max_{\mathcal{X}} \frac{1}{T} \sum_{t=1}^{T} \log P(\mathbf{x_t}|q). \qquad (3)$$

The second, more subtle source of bias is that the class conditional density functions $P(\mathbf{x}|q)$ for most feature vectors take on values very similar to the song prior density function $P(\mathbf{x})$. This creates a *song bias* where songs that have high likelihood under the prior distribution will have high likelihood under most of the class conditional distributions. We normalize for this song bias, $P(\mathcal{X})$, and use the *likelihood* $P(q|\mathcal{X})$ instead of the posterior for *retrieval*:

$$\begin{aligned} \mathcal{X}^* &= \arg\max_{\mathcal{X}} P(q|\mathcal{X}) \\ &= \arg\max_{\mathcal{X}} \frac{P(\mathcal{X}|q)P(q)}{P(\mathcal{X})} \\ &= \arg\max_{\mathcal{X}} \frac{[\prod_{t=1}^{T} P(\mathbf{x_t}|q)] \cdot P(q)}{\sum_{i=1}^{M} [\prod_{t=1}^{T} P(\mathbf{x_t}|i)] \cdot P(i)} \\ &= \arg\max_{\mathcal{X}} \frac{\sum_{t=1}^{T} \log P(\mathbf{x_t}|q)}{\sum_{i=1}^{M} \sum_{t=1}^{T} \log P(\mathbf{x_t}|i)}. \qquad (4) \end{aligned}$$

Again, we assume a uniform word prior and take logarithms for computational simplicity. Normalizing with the song bias effectively allows each song to place more weight on the words that have highest *relative* posterior. We rank songs by the weight that each song in the database places on the query word. Note that the factor $1/T$ introduced in (3) to account for the song length cancels out in (4).

4. Parameter Estimation

For each word w_i, we learn the parameters of the class conditional density, $P(\mathbf{x}|i)$ using audio features from all songs which have w_i in their associated annotations. The training set $\mathcal{T}_i$ for word w_i consists of only the *positive* examples:

$$\mathcal{T}_i = \{\mathcal{X}_d : [\mathbf{y}_d]_i = 1\} \qquad (5)$$

Note that the alternative supervised one-vs-all framework learns a classifier for each word in the vocabulary using both the positive and *negative* examples, explicitly creating a negative-class model [9]. This approach is problematic when using a data set that is *weakly labeled*: the absence of a word from the annotation does not necessarily mean that the song could not be correctly labeled with that word. In this case, the negative-class model can not be learned from data points that could have been positively labeled. Our multi-class framework focuses on learning the positive-class model using only data that is known to be positively labeled.

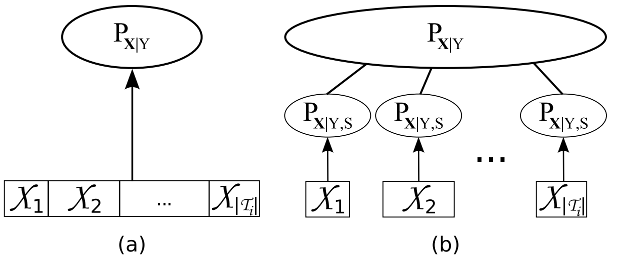

Figure 3. (a) Direct and (b) Naive Averaging parameter estimation. Arrows indicate that parameters are learned using EM.

We learn a set of M *word-level* conditional distributions $P(\mathbf{x}|i)$ for $i = 1, ..., M$, where each distribution is a C-component mixture of Gaussians distribution parameterized by $\{\pi_c, \mu_c, \Sigma_c\}$ for $c = 1, ..., C$. The word-level distribution for word w_i is given by:

$$P(\mathbf{x}|i) = \sum_{c=1}^{C} \pi_c \mathcal{N}(\mathbf{x}|\mu_c, \Sigma_c)$$

where $\mathcal{N}(\cdot, \mu, \Sigma)$ is a multivariate Gaussian distribution with mean μ and covariance matrix Σ. We consider only diagonal covariance matrices since using full covariance matrices can cause models to overfit the training data while scalar covariances do not provide adequate generalization.

We consider two parameter estimation techniques: direct estimation and naive averaging [9]. Both techniques are similar in that, for each word $w_i \in \mathcal{V}$, they use the Expectation-Maximization (EM) algorithm for fitting a mixture of Gaussians [20] distribution to the training data set $\mathcal{T}_i$ described in (5). They differ in how they break down the parameter estimation problem into subproblems and merge these results to produce a final density estimate.

4.1. Direct Estimation

Direct estimation trains a model for each word w_i using the superset of feature vectors for all the songs that have word w_i in the associated human annotation: $\bigcup \mathcal{X}_d \; \forall d$ such that $\mathcal{X}_d \in \mathcal{T}_i$. Using this training set, we directly learn the word-level mixture of Gaussian distribution using the EM algorithm (Figure 3a).

The drawback of using this method is that computational complexity increases with training set size. For example, if $\mathcal{T}_i$ contains 200 songs and there are on average T = 600 feature vectors per song, we must train each word-level model using 120,000 feature vectors. To learn a mixture of Gaussians distribution (with $C = 32$ Gaussian components), it can take many hours to train a single word-level model. A more serious deficiency of this estimation method is that the EM algorithm can converge to a bad local optimum since the set of 120,000 feature vectors can contain any and all of the outliers that exist for that word class.

4.2. Naive Averaging

Instead of directly estimating a word-level distribution for w_i, we can first learn *song-level* distributions: $P(\mathbf{x}|i, j)$, $j \in 1, ..., |\mathcal{T}_i|$ where the variable j indicates a song. We use EM to train a song-level distribution from the feature vectors extracted from that song. We then create a word-level distribution by averaging the song-level distributions of each song reviewed with w_i. *Naive averaging* gives equal weight to each song-level distribution, which results in the following distribution:

$$P(\mathbf{x}|i) = \frac{1}{|\mathcal{T}_i|} \sum_{j=1}^{|\mathcal{T}_i|} \sum_{k=1}^{K} \pi_k^{(j)} \mathcal{N}(\mathbf{x}|\mu_k^{(j)}, \Sigma_k^{(j)})$$

where K is the number of song-level mixture components (Figure 3b).

Training a model for each song in the training set and summing them is relatively efficient but the drawback of this estimation technique is that the size of word-level models grows with the size of the training database since there are $|\mathcal{T}_i| \cdot K$ components. Using the example above, if $\mathcal{T}_i$ contains 200 songs and we model each song-level distribution with $K = 8$ components then to evaluate the word-level model for a feature vector $\mathbf{x}$, we need to evaluate the probability of $\mathbf{x}$ under 1,600 multivariate Gaussian distributions.

5. Experimental Setup and Results

In this section, we quantitatively demonstrate that our system can both annotate songs with a number of relevant words and retrieve songs from database given a text query. We adopt similar evaluation methods to those used for image annotation [9, 12]. It should be noted that it is difficult for us to directly compare our results with Whitman's related work [13] since much of his research focuses on evaluation of vocabulary selection rather than retrieval performance.

We collect a set of 2,131 songs in MP3 format from our personal collections and their associated song reviews. Reviews are natural language documents describing individual songs created by human experts at AMG Allmusic [23] (see Table 1). Reviews are parsed, stemmed and converted to binary document vectors. Each review contains, on average, 19 of the 317 words in our vocabulary. For each 2 to 12 minute song, we extract one dMFCC and AFTE feature vector from half-overlapping 3/4 second windows. After applying PCA, the resulting representation is a bag of between 320 and 1920 12-dimensional feature vectors.

We randomly partition our data of song-review pairs into a training set (80%) and a test set (20%). The training set is used for learning the PCA projection matrix and the parameters for our each of our M word-level distributions. The test set is used for model evaluation. We consider two parameter estimation methods (direct with $C = 32$, naive averaging with $K = 8$) and two audio feature extraction techniques (dMFCC, AFTE) for a total of four models. We compare these models against three random baselines: random sample, prior stochastic, and prior deterministic. For each song, *random sample* picks words at random (without replacement) from our vocabulary to annotate a song.

Table 1. Original review plus the bag of words, direct model and random baseline annotations for the Monkees' "I'm a Believer".

Human Review	Bag of Words	Model Annotation	Random	Stochastic Prior	Deterministic Prior
The best of the '60s *good*-time *pop songs* and one of the most *infectious* singles ever recorded, "I'm a Believer" by the *Monkees* grooves along with a ragged accompaniment featuring handclaps and tambourine, fab *electric piano solos*, and a few well-timed, rushing-up-to-the-brink pauses just before the contagious, fervent *chorus*. For a song so incredibly *catchy*, it's hardly surprising that *songwriter* Neil Diamond (before his transformation into a *sex* bomb for middle-aged females across the nation) and producer Jeff Barry were responsible; they were two of the most talented hires on the assembly line at the *pop-song* factory known as the Brill Building, responsible for dozens of *hits* during the '60s. "I'm a Believer" itself ranks as the third most *popular rock* song of the '60s, behind only the *Beatles'* "Hey Jude" and "I Want to Hold Your Hand." It spent seven weeks at number one in *America*, *hit* the *top* of the charts in Britain as well, and charted in over a dozen countries...	american beatles catchy chorus complex electric good high hit infectious love monkees motown piano ...	monkees bouncy beatles witty call;response descending john;lennon pop;song beat british	lyric soundtrack perfect subtle electronic led;zeppelin tone contemporary roots emotion	debut;album bass catchy vocal instrumental late studio love blues chorus	lyric guitar band vocal rock pop hit love melody chorus

Prior-stochastic samples words (without replacement) from a multinomial distribution parameterized by the word prior distribution, $P(i)$ for $i = 1, ..., 317$, that are estimated using the word counts observed in the training set. *Prior-deterministic* ranks words according to the word priors $P(i)$ thus always selecting the same words for every annotation.

5.1. Annotation

Using each model, we annotate all test set songs with the 10 most likely words using (2). Annotation performance is measured using mean *per-word* precision and recall. For each word w, $|w_H|$ is the number of songs that have w in the "human" song review. $|w_A|$ is the number of songs the model "automatically" annotates with w. $|w_C|$ is the number of "correct" words used in both the song reviews and by the model. Per-word recall is $|w_C|/|w_H|$ and per-word precision is $|w_C|/|w_A|$. Mean per-word recall/precision is the average of these ratios over all 317 words in our vocabulary.

Since precision is undefined for words that the model never uses, we actually compute *smoothed* precision by placing a small non-negative weight $\epsilon/307$ on each word that the model did not use to annotate a test song ($\epsilon = 10^{-4}$). The weight of a word that is used by the model is corrected to $1-(\epsilon/10)$ so that the total weight distributed across any one test song is 10. The smoothed estimate for words that are not used by a model is approximately the word prior, $P(i)$. Without smoothing and defining precision $\equiv 0$ for words where $|w_A| = 0$, the precision of the deterministic prior (which always chooses the same 10 words) is reduced from 0.060 to 0.010 while mean precisions for all other models remain roughly unaffected.

Quantitative annotation results for the four models and three random baselines are in Table 3. Models using dMFCC features perform best and significantly beat the random baselines by 3 times in mean recall and 2 times in mean precision. Table 1 shows example annotations created by one model (dMFCCs, direct estimation) and random baselines.

5.2. Retrieval

For each word w_q, we rank the test songs in $\mathcal{S}$ according to (4) and calculate the mean average precision (mAP) [12] and the mean area under the receiver operating characteristic (ROC) curve (mAROC). Average precision is found by moving down our ranked list of test songs and averaging the precisions at every point where we correctly identify a new song. The ROC curve plots true positive rate as a function of the false positive rate as we move down our ranked list of songs. The area under the ROC is found by integrating the ROC curve. (Random guessing produces an area of 0.5 as shown empirically in Table 3). Columns 4 and 5 of Table 3 show mAP and mAROC found by averaging each metric over all the words in our vocabulary.

Similar to the annotation results, we see that our best models (direct and naive averaging with dMFCC) perform significantly better than random in both mAP and mAROC. Also, we see that models trained using dMFCC features outperform those that use AFTE features. Qualitative retrieval results for one song are shown in Table 2.

6. Discussion

While our models significantly outperform the random baselines, our best annotation results (recall = 0.09, precision = 0.12) leave much room for improvement. State-of-the-art content-based image annotation systems report mean per-word recall and precision scores of about 0.25 [9]. However, the relative objectivity of the tasks in the two domains as well as the vocabulary, the quality of annotations, the features, and the amount of data differ greatly between our music annotation system and existing image annotation systems making any direct comparisons somewhat misleading.

It should be noted that our "ground truth" human reviews represent *noisy* versions of ideal annotations. A music reviewer creating a document to describe a song does not make explicit decisions about whether specific words that we include in our vocabulary are relevant or not. Thus, relevant words are often omitted (weak labeling) and erroneous words can be included by our representation of the reviews (e.g., "this song does not rock"). We expect to improve performance in future work by replacing natural language song reviews with *clean* annotations from a manually labeled a data set. We also expect performance to improve with better, automatic vocabulary selection as in [14].

Our system has explicitly been designed to be modular so that we can incorporate new training data, test different feature extraction techniques and use alternative heterogeneous data models. Numerous short and medium-time feature extraction techniques have been proposed my the MIR

Table 2. Test set songs retrieved by our model using the query word "punk;rock" and songs in which "punk;rock" appears in the associated song review.

Automatically Retrieved	Manually Reviewed
smashing pumpkins-cherub rock	clash-the guns of brixton
ramones-pinhead	r.e.m.-radio free europe
guns n roses-you could be mine	ramones-cretin hop
neutral milk hotel-holland 1945	replacements-answering machine
built to spill-you were right	stooges-t.v. eye
replacements-answering machine	television-see no evil
cheap trick-dream police	
oasis-supersonic	
germs-manimal	
weezer-buddy holly	

Table 3. Annotation and retrieval results. All models perform significantly better than the 'random sample' baseline in a one-sided, paired t-test ($\alpha = 0.01$). Recall = mean per-word recall, Prec = mean per-word smoothed precision, mAP = mean average precision, mAROC = mean area under the ROC curve.

Model	Annotation		Retrieval	
	Recall	Prec	mAP	mAROC
Random Baselines				
Random Sample	0.030	0.060	0.071	0.49
Prior (Stochastic)	0.032	0.060	0.072	0.50
Prior (Deterministic)	0.032	0.060	0.068	0.50
dMFCC Features				
Direct	**0.087**	**0.108**	**0.105**	**0.60**
Naive Averaging	**0.072**	**0.119**	**0.109**	**0.61**
AFTE Features				
Direct	**0.067**	**0.089**	**0.092**	**0.58**
Naive Averaging	**0.055**	**0.110**	**0.097**	**0.59**

community, such as those based on psychoacoustic models [5] and autoregression [24]. We plan to explore these existing techniques as well as to design novel methods specific to the "query-by-text" annotation/retrieval tasks. We are also interested in a music model that takes account of the temporal relationships between features (e.g., a hidden Markov model [25]) as an alternative to our "bag-of-feature-vectors" representation. We plan to implement alternative parameter estimation techniques, such as Mixture Hierarchies EM [9], and experiment with unsupervised models [11, 12].

One topic not addressed by this paper is retrieval with multi-word queries. We can imagine one approach that combines rankings output from individual word models or another approach that merges word-level distributions (e.g., using naive averaging) to create a "query-level" distribution.

Acknowledgments

We would like to thank N. Vasconcelos, A. Chan, and R. Manmatha and the anonymous reviewers for their helpful comments. We used M. Slaney's Auditory Toolbox for Gammatone filtering and MFCC computation [26], A. McCallum's Rainbow [27] for text parsing and I. Nabney's NETLAB [28] for mixture model training.

References

[1] M. Goto and K. Hirata. Recent studies on music information processing. *Acoustical Science and Technology*, 25(4):419–425, 2004.

[2] R. B. Dannenberg and N. Hu. Understanding search performance in query-by-humming systems. *ISMIR*, 2004.

[3] A. Kapur, M. Benning, and G. Tzanetakis. Query by beat-boxing: Music information retrieval for the dj. *ISMIR*, 2004.

[4] Gunnar Eisenberg, Jan-Mark Batke, and Thomas Sikora. Beatbank - an mpeg-7 compliant query by tapping system. *Audio Engineering Society Convention*, 2004.

[5] M. F. McKinney and J. Breebaart. Features for audio and music classification. *ISMIR*, 2003.

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

[7] S. Essid, G. Richard, and B. David. Inferring efficient hierarchical taxonomies for mir tasks: Application to musical instruments. *ISMIR*, 2005.

[8] F. Pachet and D. Cazaly. A taxonomy of musical genres. *RIAO*, 2000.

[9] G. Carneiro and N. Vasconcelos. Formulating semantic image annotation as a supervised learning problem. *IEEE CVPR*, 2005.

[10] D. Forsyth and M. Fleck. Body plans. *IEEE CVPR*, 1997.

[11] D. M. Blei and M. I. Jordan. Modeling annotated data. *ACM SIGIR*, 2003.

[12] S. L. Feng, R. Manmatha, and V. Lavrenko. Multiple bernoulli relevance models for image and video annotation. *IEEE CVPR*, 2004.

[13] B. Whitman. *Learning the meaning of music*. PhD thesis, Massachusetts Institute of Technology, 2005.

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

[15] B. Whitman and R. Rifkin. Musical query-by-description as a multiclass learning problem. *IEEE MMSP*.

[16] M. Slaney. Semantic-audio retrieval. *IEEE ICASSP*, 2002.

[17] P. Cano and M. Koppenberger. Automatic sound annotation. In *IEEE workshop on Machine Learning for Signal Processing*, 2004.

[18] L. Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

[19] B. Logan. Mel frequency cepstral coefficients for music modeling. *ISMIR*, 2000.

[20] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: Data mining, inference, and prediction*. Springer, 2001.

[21] W. M. Hartmann. *Signals, Sound, and Sensation*. Springer-Verlag, 1998.

[22] D. Reynolds, T.F. Quatieri, and R.B. Dunn. Speaker verification using adapted gaussian mixture models. *Digital Signal Processing*, 10:19–41, 2000.

[23] AMG. Allmusic guide. http://www.allmusic.com.

[24] A. Meng, P. Ahrendt, and J. Larsen. Improving music genre classification by short-time feature integration. *IEEE ICASSP*, 2005.

[25] A. Flexer, E. Pampalk, and G. Widmer. Novelty detection for spectral similarity of songs. *ISMIR*, 2005.

[26] M. Slaney. Auditory toolbox. *Interval Research Corporation Technical Report 1998-010*. http://rvl4.ecn.purdue.edu/ malcolm/interval/1998-010/.

[27] A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow, 1996.

[28] I. Nabney and C. Bishop. Netlab toolbox. http://www.ncrg.aston.ac.uk/netlab/.