Modeling the Semantics of Sound

Douglas Turnbull
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093
dturnbul@cs.ucsd.edu

Luke Barrington
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, CA 92093
lbarring@ucsd.edu

David Torres
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093
dtorres@cs.ucsd.edu

Gert Lanckriet
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, CA 92093
gert@ece.ucsd.edu

Abstract

Despite the recent focus on semantic image/video annotation and retrieval, relatively little work has been done on semantic audio annotation and retrieval. We show that the supervised mutli-class naïve Bayes model, which has successfully been used for image annotation, can be used to model the semantics of audio data. This model, as opposed a host of recently proposed unsupervised model, has shown superior performance and requires relatively little computation time for both parameter estimation and inference. Fast parameter estimation is achieved using the *mixture hierarchies* algorithm. We consider two heterogeneous audio and text data sets; sound efffects with captions, and music with associated reviews. We show that, with the supervised multi-class model, we can both *annotate* a novel audio track with semantically meaningful words and *retrieve* relevant audio track given a text-based query.

1 Introduction

Sound carries rich information from which we derive semantic understanding and we naturally use language to describe what we hear and what we might want to hear. This paper presents an initial *computer audition* system that derives similar notions of semantics from sound. Specifically, our goal is to create a system that can both *annotate* audio content with semantically meaningful language and *retrieve* relevant sounds requested by human users. We view the related problems of

sound annotation and retrieval as a supervised learning problem in which each heterogeneous piece of data is represented by both audio and text. We learn a joint probabilistic model of audio music features and semantic tokens (referred to as 'words') from the data. Using the model, we can infer likely words given a novel sound, and we can rank a set of unannotated sounds given a text-based query.

Our proposed solution uses a supervised multi-class naïve Bayes approach. This model was recently used by [1] for image annotation and was proposed as an alternative to the numerous unsupervised models that have been developed over the last five years [2, 3, 4, 5]. The supervised approach directly represents each word as a class rather than introducing latent variables to encode a set of states defining a joint distribution over the audio content and all the words in the vocabulary. The supervised approach is appealing since it has been shown to achieve good annotation and retrieval results[1], it is conceptually straight-forward, and can be computationally efficient with respect to both parameter estimation and inference.

In our implementation, efficient estimation is achieved using hierarchical approach that involves first modeling the audio content of each track with a Gaussian mixture model (GMM) and then combining sets of these track-level distributions using the *mixture hierarchies* algorithm [6]. This algorithm is an extension of the standard expectation maximization (EM) algorithm for learning a GMM. The mixture hierarchies GMM parameter estimation technique and two alternatives are described in Section 4.

2 Related work

While the idea of developing a system to model the semantics of images and videos has received recent attention [1, 3, 4, 5, 7], literature on modeling the semantics of audio data is relatively limited. Slaney's Semantic Audio Retrieval system [8, 9], and extensions proposed by Buchanan [10], represent one tread of research on sound effects retrieval. Their approach involves creating separate hierarchical models in the acoustic and text space, and then creating links between the two spaces for either retrieval or annotation. Cano and Koppenberger propose a similar approach based on nearest neighbor classification [11]. These drawback of these non-parametric approaches is that inference requires calculating the similarity between a query and every training example. We propose a parametric approach that requires one calculation per semantic concept. In practice, the number of semantic concepts is orders of magnitude smaller than the number of potential training data points, leading to a more scalable solution.

Previous work on modeling the semantics of audio data has focused almost exclusively on sound effects. One exception is the work of Whitman and Ellis who present research on modeling a heterogeneous data set of music and words [12]. Their work focuses on finding terms that can be predicted from audio content using binary classifiers learned for each term. By identifying these "grounded" annotations they prune sentences in order to create unbiased album reviews.

More generally, there has been substantial work on acoustic signal processing and content-based information retrieval (specifically involving the retrieval of text, music, images, and videos). The music information retrieval community, has produced many useful audio feature extraction techniques for modeling music (for example, see [13, 14, 15]). This community has focused on music classification (by genre, instrumentation, emotion) and *query-by-example* retrieval systems. Query-by-example retrieves sounds from a database based on similarity between audio-based queries such as songs [16] or audible noises (e.g. query-by-humming [17]). We propose a system that can both annotate and retrieve music based on semantic keywords; *query-by-text*.

We take inspiration from research on modeling the semantics of images and video (see [1] for a recent review), employ Mel-frequency cepstral coefficients (MFCC) features shown to be useful for speech and audio modeling [18] and apply these tools to the relatively unexplored domain of semantic modeling of sound.

3 Semantic audio annotation and retrieval

This section by formalizes the related problems of semantic audio annotation and retrieval as a supervised multi-class naïve Bayes approach where each word in our vocabulary represents a class.

We learn class-conditional distributions for each class using only the training songs that have been positively label with the associated word. The alternative one-versus-all approach is infeasible since our data sets are *weakly labeled*: the absence of a word from an annotation does not necessarily mean that the track could not be correctly labeled with that word. For example, a song reviewer might fail to mention the word 'drum' even though drums are featured in the song. Using a multi-class framework, we focus on learning just the positive-class model from only the positively labeled data.

3.1 Problem formulation

Consider a vocabulary $\mathcal{V}$ consisting of $|\mathcal{V}|$ 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 'creaky;door'. 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 audio tracks (i.e. songs) $\mathcal{S} = \{s_1, ..., s_R\}$ 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 an audio track s as a set $\mathcal{X} = \{\mathbf{x_1}, ..., \mathbf{x_T}\}$ of T real-valued feature vectors, where each vector $\mathbf{x_t}$ represents features extracted from a short segment of the audio content and T depends on the length of the song. Our data set $\mathcal{D}$ is a collection of track-annotation pairs $\mathcal{D} = \{(\mathcal{X}_1, \mathbf{y}_1), ..., (\mathcal{X}_D, \mathbf{y}_D)\}$.

3.2 Multi-class naïve Bayes model

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 $P(\mathbf{x}|i), i \in \{1, ..., |\mathcal{V}|\}$ 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:

$$\begin{aligned} i^* &= \arg\max_i P(i|\mathcal{X}_q) \\ &= \arg\max_i \frac{P(\mathcal{X}_q|i)P(i)}{P(\mathcal{X}_q)} \end{aligned}$$

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

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

We assume a uniform prior (i.e. $P(i) = 1/M$ for all $i = 1, .., M$) because the T factors in the product will dominate the word prior. Taking the logarithm, we arrive at our final *annotation* equation:

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

While the naïve Bayes assumption introduced in Equation 1 is unrealistic, attempting to model the interaction between feature vectors may be infeasible due to computational complexity and data sparsity. Computing Equation 2 for each word creates an ordering for all words in the vocabulary. During annotation, we select the 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 features) 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 (see Equation 1) between the audio feature vectors [19]. The standard solution is to calculate the *average* log posterior for each track

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

where T is proportional to the length of the song.

The second, more subtle, problem with using the posterior emerges as a result of the multiple instance learning method our weakly labeled data set requires. The class conditional density functions $P(\mathbf{x}|q)$ for most features take on values very similar to the track prior density function $P(\mathbf{x})$. This creates a *track bias* in which tracks that have high likelihood under the prior distribution will have high likelihood under most of the class conditional distributions. We normalize for this track bias, $P(\mathcal{X})$, and use the *likelihood* $P(q|\mathcal{X})$ instead of the posterior for *retrieval*:

$$\begin{aligned}
\mathcal{X}^* &= \arg\max_{\mathcal{X}} P(q|\mathcal{X}) && (4)\\
&= \arg\max_{\mathcal{X}} \frac{P(\mathcal{X}|q)P(q)}{P(\mathcal{X})} && (5)\\
&= \arg\max_{\mathcal{X}} \frac{[\prod_{t=1}^{T} P(\mathbf{x_t}|q)] \cdot P(q)}{\sum_{i=1}^{|\mathcal{V}|}[\prod_{t=1}^{T} P(\mathbf{x_t}|i)] \cdot P(i)} && (6)\\
&= \arg\max_{\mathcal{X}} \frac{\sum_{t=1}^{T} \log P(\mathbf{x_t}|q)}{\sum_{i=1}^{|\mathcal{V}|}\sum_{t=1}^{T} \log P(\mathbf{x_t}|i)} && (7)
\end{aligned}$$

where M is the size of the vocabulary. Again, we assume a uniform word prior and take logarithmic transform for computational simplicity. By normalizing with the song bias, we effectively allow each song to place more weight on the words that have highest *relative* posterior. We then rank songs by the weight that each song in the database places on the query word. Note that the factor $1/T$ introduced in Equation 3 to account for the song length cancels out in our Equation 7.

4 Parameter Estimation

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

$$\mathcal{T}_i = \{\mathcal{X}_d \,:\, [\mathbf{y}_d]_i = 1\} \quad (8)$$

The output of parameter estimation is a set of M *word-level* distributions $P(\mathbf{x}|i)$ for $i = 1, ..., M$, where we represent each distribution as a R-component mixture of Gaussian distribution parameterized by $\{\pi_r, \mu_r, \Sigma_r\}$ for $r = 1, ..., R$. The word-level distribution for word w_i is given by:

$$P(\mathbf{x}|i) = \sum_{r=1}^{R} \pi_r \mathcal{N}(\mathbf{x}|\mu_r, \Sigma_r)$$

where $\mathcal{N}(\cdot|\mu, \Sigma)$ is a multivariate Gaussian distribution with mean μ and covariance matrix Σ. In this work, 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. The resulting set of M models each have $\mathcal{O}(R \cdot D)$ parameters, where D is the dimension of feature vector $\mathbf{x}$.

We consider three parameter estimation techniques for learning a supervised multi-class naïve Bayes model: direct estimation, naïve averaging, and mixture hiearchies [1]. The 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 to training data. They differ in how they break down the problem of parameter estimation into subproblems and then 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 (see Figure 1a). The drawback of using this method is that computational complexity

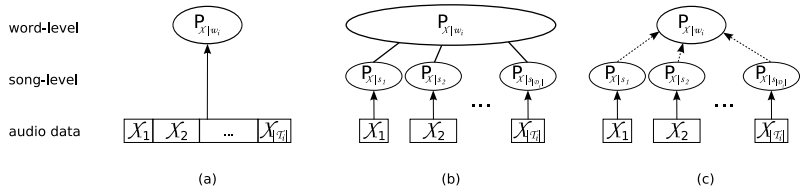

Figure 1: (a) Direct, (b) naive averaging, and (c) mixture hierarchies parameter estimation. Solid arrows indicate that the distribution parameters are learned using standard EM. Dashed arrows indicate that the distribution is learned using mixture hierarchies EM.

increases with training set size. We find that, in practice, we are unable to estimate parameters using this method in a reasonable amount of time since there are on the order of 100,000s of training vectors for each word-level distribution. $\bigcup \mathcal{X}_d$. We can subsample our training data but this is not optimal since we not utilizing all of the available training data. E

4.2 Naive averaging

Instead of directly estimating a word-level distribution for w_i, we can first learn *track-level* distributions: $P(\mathbf{x}|i,j)$, $j \in 1,...,|\mathcal{T}_i|$ where the variable j indicates an audio track. 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 track-level distributions of each track annotated with w_i. *Naive averaging* gives equal weight to each track-level distribution, resulting in the following distribution:

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

where K is the number of mixture components in each track-level distribution (see Figure 1b).

Training a model for each track in the training set and summing them is relatively efficient, however, the drawback of this estimation technique is that the size of word-level models grows with the size of the training database since there will be $|\mathcal{T}_i| \cdot K$ components for word w_i. In practice, we may have to evaluate thousands of multivariate Gaussian distributions each of the feature vectors $\mathbf{x}_t \in \mathcal{X}_q$, where $\mathcal{X}_q$ represents a novel query track. Note that $\mathcal{X}_q$ may contain on the order of 10,000 feature vectors depending on the audio representation.

4.3 Mixtures hierarchies estimation

The benefit of direct estimation is that it produces a parametric distribution with a fixed number of parameter. However, in practice, parameter estimation is infeasible without subsampling the training data. Naïve averaging estimation can efficiently produce a parametric distribution, but it is computationally expensive to evaluate this distribution since the number of parameters increases with the size of the training data set. Mixture hierarchy estimation is an alternative is efficient and produces a parametric distribution with a fixed number of parameters [6].

Consider the set of $|\mathcal{T}_i|$ track-level distribution (each with K mixture components) that are learned during naïve estimation. We can estimate a word-level distribution with R component using an extension of the EM algorithm where the track-level distributions are consider the children and the word-level distribution is the parent. (See Figure 1c.) This EM algorithm iterates between the E-step and the M-step:

E-step: Compute the responsibilities of each of the parent components to a child component

$$h_{(j),k}^{r} = \frac{\left[\mathcal{N}(\mu_k^{(j)}|\mu_r, \mathbf{\Sigma}_r) e^{-\frac{1}{2} trace\{(\mathbf{\Sigma}_r)^{-1}\mathbf{\Sigma}_k^{(j)}\}}\right]^{\pi_k^{(j)} N} \pi_r}{\sum_l \left[\mathcal{N}(\mu_k^{(j)}|\mu_l, \mathbf{\Sigma}_l) e^{-\frac{1}{2} trace\{(\mathbf{\Sigma}_l)^{-1}\mathbf{\Sigma}_k^{(j)}\}}\right]^{\pi_k^{(j)} N} \pi_l}$$

where N is a user defined parameter. (In practice, we set $N = K$ so that $E[\pi_k^{(j)} N] = 1$.)

M-step: Update the parameters of the parent distribution

$$\pi_r^{new} = \frac{\sum_{(j),k} h^r_{(j),k}}{|\mathcal{T}_i| \cdot K} \tag{9}$$

$$\mu_r^{new} = \sum_{(j),k} w^r_{(j),k} \mu_k^{(j)}, \text{where } w^r_{(j),k} = \frac{h^r_{(j),k} \pi_k^{(j)}}{\sum_{(j),k} h^r_{(j),k} \pi_k^{(j)}} \tag{10}$$

$$\mathbf{\Sigma}_r^{new} = \sum_{(j),k} w^r_{(j),k} \left[\mathbf{\Sigma}_k^{(j)} + (\mu_k^{(j)} - \mu_t)(\mu_k^{(j)} - \mu_t)^T \right] \tag{11}$$

From a generative perspective, the track-level distribution is generated by sampling *mixture components* from the word-level distribution. The observed audio features are then samples from the track-level distribution. Note that the number of parameters for the word-level distribution is the same as the number of parameter resulting from direct estimation yet we learn this model using all of the training data without subsampling. We have essentially replaced one computationally expensive (and often impossible) run of the standard EM algorithm with $|\mathcal{T}_i|$ computationally inexpensive runs and one run of the mixture hierarchies EM. In practice, mixture hierarchies EM requires about the same computation time of one run of standard EM.

5 Model evaluation

In this section, we quantitatively evaluate our supervised multi-class naïve Bayes model for audio annotation and retrieval. Our music data set consist of 2,131 song-review pair where the reviews have been composed by expert music critics at AMG Allmusic [20]. The song taken from the our private collections of music. The sound effects data set consists 1364 track-caption pairs from the BBC Sound Effects. We find it hard to compare our results to existing work [8, 11] since the existing results are mainly qualitiative and relate to individual tracks, or focus on a small subsets of sound effects (e.g. animial vocalizations or isolated musical instrument). To our knowledge, there has been very little work done on semantic music annotation [12] and virtually no work focused on semantic music retrieval.

We evaluate our system against three baselines models: 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. *Prior-stochastic* samples words (without replacement) from a multinomial distribution parameterized by the word prior distribution, $P(i)$ for $i = 1...|\mathcal{V}|$, estimated using the observed word counts of the training set. *Prior-deterministic* ranks words according to the word prior, $P(i)$, and always select the same words for every annotation.

Evaluation is performed using 10-fold validation. However, both models learning using direct estimation and inference using models learned using naive averaging can take long time (e.g. days). For these estimation techniques, we present results for just one fold.

5.1 Representing text and audio data

We represent each text document (i.e. song review or sound effects caption) as a *bag of words*: a set of words $\mathcal{W}$ that are found in both the review and our vocabulary $\mathcal{V}$. In the context of music, our musical vocabulary consists of 317 musically informative words that we hand picked from a list of common words found in a corpus of song reviews. "Musically informative" means that the word may describe something about the audio content. 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', become the same word, while other words, such as 'blue' and 'blues', remain unaffected. Our sound effects corpus consists of the captions associated with BBC Sound Effects library. We select the 349 words each of which appear 5 or more times in this corpus.

Similarly, sounds are represented as a *bag-of-feature-vectors*: we extract an unordered set of feature vectors for each segments of audio data. The length of the segment and feature extraction technique depends on the class of audio. For music, we compute dynamic Mel-frequency cepstral coefficients

(dMFCCs) each half-overlapping, medium-time ($\sim$743 msec) segment of music audio[14]. This results in about 800 52-dimensional feature vectors for a five minute song. For sound effects, we compute delta cepstrum feature vectors for each half-overlapping short-tme ($\sim$12 msec) segment [10]. We extract about 5000 39-dimensional feature vectors each 30 second of audio content.

We have also explored using principal component analysis (PCA) to reduce the dimension of the audio feature vectors. We find that in practice PCA increases performance for music but decreases performance for sound effect. The improvement of performance is most likely due to the fact DMFCC features are correlated with one another.

5.2 Annotation

Using Equation 2, we annotate all test set songs with 10 words and all test set sound effect tracks with 4 words. Annotation performance is measured using mean *per-word* precision and recall. Per-word precision measures how often our system correctly guesses that a word will appears in a review. Per-word Recall measures the percentage of reviews with a word that the system identifies as containing that word. More formally, for each word w, $|w_H|$ is the number of tracks that have word w in the "human" document. $|w_A|$ is the number of tracks that a model "automatically" annotates with word w. $|w_C|$ is the number of "correct" words that have been used both in the document 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 and precision is the average of these ratios over all the words in our vocabulary. It should be noted that these metrics range between 0.0 and 1.0, but one may be upper bounded by a value less than 1.0 if either the number of words that appear corpus is greater or lesser than the number of words that are output by our system. For example, our system outputs 2150 words for the 215 test songs for a corpus that contains 4116 words, thus mean recall will be upper-bounded by a value less than one. The exact upper bound will depend on the relative word frequencies of each word in the vocabulary. For our music data set, mean per-word recall is bound by 0.84 if we perfectly predict the most infrequent words.

Precision is undefined for words that the model never uses. Therefore, we actually compute *smoothed* precision by placing a small non-negative weight $\epsilon/|\mathcal{V}|$ on each word that the model did not use to annotate a test song ($\epsilon = 0.0001$). 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_Y. If we do not smooth and define 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. It may seem more straightforward to use *per-song* precision and recall, rather than the per-word metrics describe above. However, per-song metrics can lead to artificially good results if a system is good at predicting the few common words and bad at predicting the many rare words in the vocabulary. Our goal is to find a system that is good at predicting all the words.

In Table 1, we see that our model trained using mixture hierarchy estimation significantly outperforms the three baselines for both the music and sound effects data sets. We see that for music, models trained using the three parameter estimation techniques are comparable, but for sound effects, naive averaging results in superior performance. For the music data set, each word is often present in 100s of reviews, where is for sound effects, each word is in 10s of captions. Thus for sound effects, the number of mixture components is manageable and produces a good density estimate.

5.3 Retrieval

For each word w_q, we rank the test songs in $\mathcal{S}$ according to Equation 6 and calculate the mean average precision (mAP) [3] 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. A receiver operating characteristic (ROC) curve is a plot of 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 (AROC) is found by integrating the ROC curve and is upper bound by 1.0. Random guessing produces an AROC of 0.5 (as shown

Table 1: Audio annotation and retrieval results: mAP = mean average precision, mAROC = mean area under the ROC curve, Cover = number of unique words used by a model. Values represent means and standard error using 10-fold cross validation. Values without an associated standard deviation represent the result of just one fold (due to computational constraints).

Model	Annotation			Retrieval	
	Recall	Precision	Cover	mAP	mAROC
Music - $\|\mathcal{V}\| = 317$, A=10					
Random Sample	0.029 (0.002)	0.059 (0.002)	316 (1)	0.082 (0.001)	0.496 (0.002)
Prior (Stochastic)	0.032 (0.001)	0.061 (0.002)	297 (1)	0.084 (0.001)	0.502 (0.002)
Prior (Deterministic)	0.032 (0.000)	0.061 (0.001)	10 (0)	0.081 (0.001)	0.496 (0.004)
DMFCC features, 12 PCA dimensions					
Direct (R=32)	**0.081**	0.106	290	**0.121**	0.600
NaiveAvg (K = 8)	0.072	**0.119**	290	0.109	**0.610**
MixHier (K = 8, R = 32)	0.077 (0.003)	0.095 (0.003)	263 (2)	0.117 (0.003)	0.598 (0.003)
Sound effects - $\|\mathcal{V}\| = 349$, A=4					
Random Sample	0.014 (0.002)	0.012 (0.001)	269 (3)	0.052 (0.002)	0.509 (0.003)
Prior (Stochastic)	0.013 (0.001)	0.011 (0.001)	229 (2)	0.049 (0.001)	0.505 (0.005)
Prior (Deterministic)	0.018 (0.002)	0.010 (0.000)	4 (0)	0.052 (0.002)	0.502 (0.005)
Delta cepstrum features					
Direct ($R = 12$)	0.166	0.150	207	0.152	0.761
NaiveAvg ($K = 8$)	**0.243**	**0.220**	208	**0.201**	**0.793**
MixHier ($K = 8, R = 12$)	0.189 (0.011)	0.125 (0.012)	194 (1)	0.187 (0.013)	0.759 (0.007)

empirically in Table 1) since . Columns 4 and 5 of Table 1 show mAP and mAROC found by averaging each metric over all the words in our vocabulary.

Similar to the annotation results, we see that (in Table 1) our model significantly outperforms the baseline models. Again, note that naive averaging estimation produces the best sound effects model and may be preferred over mixture hierarchies estimation when the number of track-level models is small.

5.4 Comments

While our models significantly outperform the random baselines, the best audio annotation results, especially our music results, leave room for improvement. State-of-the-art content-based image annotation systems [1] report mean per-word recall and precision scores of about 0.25 which is comparable to our best sound effects results. 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 audio annotation system and existing image annotation systems. This makes any direct comparisons somewhat misleading.

For music, 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 found in our feature representation of the reviews (e.g., "this song does not rock"). In an informal evaluation of our "ground truth", we asked six individuals to evaluate 20 songs each. For each song, we presented individual with five words from the ground truth annotation, and asked them to say whether each word was relevant or not. We found that on average about two of the five words were relevant (e.g. precision = 0.38). This suggest that we are learning word models based on with many irrelevant songs.

6 Discussion

The goal for this work is to provide a framework semantic audio annotation and retrieval. Our initial approach was based on a recently proposed image annotation model. We can imagine adapting other models such as the correspondence latent Dirichlet allocation model [2] or the mutliple Bernoulli relevance model [3] for modeling audio data. Future work will also involves incorporating alternative audio representations. We used existing audio feature extraction techniques and model these

features using GMMs. One drawback of using GMMs is that they ignore the temporal nature of extracted audio feature vectors. Exploring alternative audio representations and modeling the temporal aspects our our audio (with hidden Markov models) may lead to better performance.

Another goal was to use existing annotations rather than to manually annotate each audio track. This approach allows us to rapidly incorporate new training data without spending time and energy on creating labels. However, our initial approach for deriving semantic labels for music (e.g. representing a review as a bags-of-words) produced unreliable annotations. This may be remedied by collecting a data set that is specifically designed for the task of music annotation and retrieval.

References

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

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

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

[4] J. Li and J. Z. Wang. Automatic linguistic indexing of pictures by a statistical modeling approach. *IEEE PAMI*, 25(9):1075–1088, 2003.

[5] K. Barnard, P. Duygulu, D. Forsyth, N. de Freitas, D. Blei, and M. Jordan. Matching words and pictures. *JMLR*, 3:1107–1135, 2002.

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

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

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

[9] M. Slaney. Mixtures of probability experts for audio retrieval and indexing. *IEEE Multimedia and Expo*, 2002.

[10] C. R. Buchanan. Semantic-based audio recognition and retrieval. Master's thesis, School of Informatics, University of Edinburgh, 2005.

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

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

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

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

[15] 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.

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

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

[18] L. Rabiner and B. H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

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

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