SEMANTIC ANNOTATION AND RETRIEVAL OF MUSIC USING A BAG OF SYSTEMS REPRESENTATION

Katherine Ellis
University of California, San Diego
kellis@ucsd.edu

Emanuele Coviello
University of California, San Diego
ecoviell@ucsd.edu

Gert R.G. Lanckriet
University of California, San Diego
gert@ece.ucsd.edu

ABSTRACT

We present a content-based auto-tagger that leverages a rich dictionary of musical codewords, where each codeword is a generative model that captures timbral and temporal characteristics of music. This leads to a higher-level, concise "Bag of Systems" (BoS) representation of the characteristics of a musical piece. Once songs are represented as a BoS histogram over codewords, traditional algorithms for text document retrieval can be leveraged for music auto-tagging. Compared to estimating a single generative model to directly capture the musical characteristics of songs associated with a tag, the BoS approach offers the flexibility to combine different classes of generative models at various time resolutions through the selection of the BoS codewords. Experiments show that this enriches the audio representation and leads to superior auto-tagging performance.

1. INTRODUCTION

Given a vast and constantly growing collection of online songs, music search and recommendation systems increasingly rely on automated algorithms to analyze and index music content. In this work, we investigate a novel approach for automated content-based tagging of music with semantically meaningful tags (e.g., genres, emotions, instruments, usages, etc.). Most previously proposed auto-taggers rely either on discriminative algorithms [2, 7, 11–13], or on generative probabilistic models, including Gaussian mixture models (GMMs) [19, 20], hidden Markov models (HMMs) [13, 15], hierarchical Dirichlet processes (HDPs) [9], codeword Bernoulli average models (CBA) [10], and dynamic texture mixture models (DTMs) [5].

Most generative approaches first propose a general probabilistic model — the base model — that can adequately capture the typical characteristics of musical audio signals. Then, for each tag in a given vocabulary, an instance of this base model is fine-tuned to *directly* model the audio patterns that are specific and typical for songs associated with that tag. For example, Turnbull et. al. [19] propose Gaussian mixture models (GMMs) over a "bag of features" (BoF) representation, where each acoustic feature represents the timbre of a short snippet of audio. Coviello et. al. [5] use dynamic texture mixture models (DTMs) over a "bag of fragments" representation, where each fragment is a sequence of acoustic features extracted from a few seconds of audio. DTMs capture information about the temporal dynamics (e.g. rhythm, beat, tempo) of an audio fragment, as well as instantaneous timbral content.

Such *direct* generative approaches may suffer from two inherent limitations. First, their flexibility is determined by the choice of the base model. Since different base models may capture complementary characteristics of a musical signal, selecting a single base model may restrict the modeling power a priori. For example, Coviello et al. [5] reported that DTMs are particularly suitable to model tags with significant temporal characteristics, while GMMs are favorable for some tags for which "timbre says it all". Moreover, specifying a base model implies setting its time scale parameters. This limits direct generative approaches to detecting musical characteristics (timbre, temporal dynamics, etc.) at one fixed time resolution, for each tag in the vocabulary. This is suboptimal, since the acoustic patterns that characterize different tags may occur at different time resolutions. Second, estimating tag models may require tuning a large number of parameters, depending on the complexity of the base model. For tags with relatively few observations (i.e., songs associated with the tag), this may be prone to overfitting.

To address these limitations, we propose to use generative models to *indirectly* represent tag-specific musical characteristics, by leveraging them to extract a *high-level* song representation. In particular, we propose to model a song using a "bag of systems" (BoS) representation for music. The BoS representation is analogous to the "bag of words" (BoW) framework employed in text retrieval [1], which represents documents by a histogram of word counts from a given dictionary. In the BoS approach, each word is a generative model with fixed parameters. Given a rich dictionary of such "musical codewords", a song is represented by "counting" the of occurrences of each codeword in the song — by assigning song segments to the codeword with largest likelihood. Finally, BoS histograms can be modeled by appealing to standard text mining methods (e.g., logistic regression, topic models, etc.), to obtain tag-level models for automatic annotation and retrieval. A BoS approach has been used for the classification of videos [4, 14], and a similar idea has inspired the anchor modeling for speaker identification [16].

By leveraging the complementary modeling power of *various* classes of generative models, the BoS approach is more flexible than direct generative approaches. In this work, we demonstrate how combining Gaussian and dynamic texture codewords with different time resolutions enriches the representation of a song's acoustic content and improves performance. A second advantage of the BoS approach is that it decouples modeling *music* from modeling *tags*. This allows us to leverage sophisticated generative models for the former, while avoiding overfitting by resorting to relatively simpler BoW models for the latter. More precisely, in a first step, a dictionary of sophisticated codewords may be estimated from *any* large collection of representative audio data, which need not be annotated. This allows to learn a general, rich BoS representation of *music* robustly. Next, *tag* models are estimated to capture the typical codeword patterns in the BoS histograms of songs associated with each tag. As each tag model already leverages the descriptive power of a sophisticated codebook representation, relatively simple tag models (with fewer tunable parameters) may be estimated reliably, even from small sets of tag-specific training songs.

In summary, we present a new approach to auto-tagging that constructs a rich dictionary of musically meaningful words and represents each song as a histogram over these words. This simple, compact representation of the musical content of a song is computationally efficient once learned and expected to be more robust than a single low-level audio representation. It can benefit from the modeling capabilities of several classes of generative models, and exploit information at multiple time scales.

2. THE BAG OF SYSTEMS REPRESENTATION OF MUSIC

Analogous to the BoW representation of text documents, the BoS approach represents songs with respect to a codebook, in which generative models are used in lieu of words. These generative models compactly characterize typical audio features, musical dynamics or other acoustic patterns in songs.

We discuss codebook generation in Section 2.1, the generative models used as codewords in Section 2.2, and the representation of songs using the codebook in Section 2.3.

2.1 Codebook generation

To build a codebook, we first choose M classes of base models (each with a certain allocation of time scale parameters). From each model we derive a set of representative codewords, i.e., instances of that model class that capture meaningful musical patterns. We do this first by defining a representative collection of songs, i.e., a codebook set, $\mathcal{X}_c$, and then modeling each song in $\mathcal{X}_c$ as a mixture of K_s models from each model class. After parameter estimation, the mixture components provide us with characteristic instances of that model class and become codewords. Finally, we aggregate all codewords to form the BoS codebook, $\mathcal{V}$, which contains $|\mathcal{V}| = MK_s|\mathcal{X}_c|$ codewords.

Each codeword in the BoS codebook can be seen as characterizing a prototypical audio pattern or texture, and codewords from different classes of generative models capture different types of musical information. If the codebook set, $\mathcal{X}_c$, is sufficiently diverse, the estimated codebook will be rich enough to represent songs well.

2.2 The codewords

To obtain a diverse codebook, we consider Gaussian models (to characterize timbre) and dynamic texture (DT) models [6] (to capture temporal dynamics) at various time resolutions. First, a time resolution is chosen by representing songs as a sequence of feature vectors, $\mathcal{Y} = \{y_1, \dots, y_T\}$, extracted from half-overlapping time windows of length η. The sampling rate and the length η of the windows determines the time resolution of the generative models. Second, a generative model (Gaussian or DT) is chosen, and mixture models are estimated for all songs in the codebook set, $\mathcal{X}_c$.

2.2.1 Gaussian codewords

To learn Gaussian codewords, we fit a Gaussian mixture model (GMM) to each song in $\mathcal{X}_c$, to capture the most prominent audio textures it exhibits. More specifically, for each song in $\mathcal{X}_c$, we treat the sequence of its feature vectors, $\mathcal{Y}$, as an unordered bag of features, and use the EM algorithm to estimate the parameters of a GMM from these features. Finally, each mixture component is considered as a codeword, characterized by parameters $\Theta_i = \{\mu_i, \Sigma_i\}$, where μ_i and Σ_i are the mean and covariance of the i^{th} mixture component of the GMM, respectively.

2.2.2 Dynamic Texture codewords

Dynamic texture (DT) codewords are learned by modeling each song in $\mathcal{X}_c$ as a mixture of DTs, and considering each individual DT as a codeword.

DTs explicitly model the temporal dynamics of audio by modeling ordered sequences of audio features rather than individual features. From the sequence of feature vectors extracted from a song, $\mathcal{Y}$, we sample subsequences, i.e., fragments, $y_{1:\tau}$, of length τ every ν seconds. We then represent the song by an unordered bag of these audio fragments, $\mathcal{Y} = \{y^1_{1:\tau}, \ldots, y^T_{1:\tau}\}$.

A DT treats an audio fragment $y_{1:\tau}$ as the output of a linear dynamical system (LDS):

$$x_t = Ax_{t-1} + v_t, \quad (1)$$

$$y_t = Cx_t + w_t + \bar{y}, \quad (2)$$

where the random variable $y_t \in \mathbb{R}^m$ encodes the timbral content (audio feature vector) at time t, and a lower dimensional hidden variable $x_t \in \mathbb{R}^n$ encodes the dynamics of the observations over time. The model is specified by parameters $\Theta = \{A, Q, C, R, \mu, S, \bar{y}\}$, where the state transition matrix $A \in \mathbb{R}^{n\times n}$ encodes the evolution of the hidden state x_t over time, $v_t \sim \mathcal{N}(0, Q)$ is the driving noise process, the observation matrix $C \in \mathbb{R}^{m\times n}$ encodes the basis functions for representing the observations y_n, $\bar{y}$ is the mean of the observation vectors, and $w_t \sim \mathcal{N}(0, R)$ is the observation noise. The initial condition is distributed as $x_1 \sim \mathcal{N}(\mu, S)$.

Because songs often consist of several heterogeneous sections, such as chorus, verse, etc., one dynamic texture model is generally not rich enough to describe an entire song [5]. Therefore, we model a song by a dynamic texture mixture (DTM) where an assignment variable $z \in \{1, 2, ..., K_s\}$ selects which of K_s DTs is generating an audio fragment. A DTM model can be interpreted as summarizing the dominant temporal dynamics that occur in a song. For a given a song, the DTM parameters are estimated via the EM algorithm [3] and, once again each mixture component Θ_i is a codeword, capturing a particular musical dynamic.

2.3 Representing songs with the codebook

Once a codebook is available, a song is represented by a codebook multinomial (CBM) $\mathbf{b} \in \mathbb{R}^{|\mathcal{V}|}$ that reports how often each codeword appears in that song, where $b[i]$ is the weight of codeword i in the song.

To build the CBM for a given song, we count the number of occurrences of each codeword in the song by computing its likelihood at various points in the song (e.g., every ν seconds) and comparing it to the likelihood of other codewords derived from the same base model class (since likelihoods are only comparable between similar models with the same time resolution). To compute the likelihood of a given codeword at a certain point in the song, we extract a fragment of audio information y^t depending on the time scale and model class of the codeword in question. I.e., for GMM codewords, y^t is a single audio feature vector, extracted from a window of width η, while for DTM codewords, y^t is a sequence of τ such feature vectors. We count an occurrence of the codeword under attention if it has the highest likelihood of all the codewords in that class.

We construct the histogram $\mathbf{b}$ for song $\mathcal{Y}$ by counting the frequency with which each codeword $\Theta_i \in \mathcal{V}$ is chosen to represent a fragment:

$$b[i] = \frac{1}{M|\mathcal{Y}_m|} \sum_{y^t \in \mathcal{Y}_m} \mathbb{1}[\Theta_i = \operatorname*{argmax}_{\Theta \in \mathcal{V}_m} P(y^t|\Theta)] \quad (3)$$

where $\mathcal{V}_m \subseteq \mathcal{V}$ is the subset of codewords derived from the model class m which codeword Θ_i is derived. Normalizing by the number of fragments $|\mathcal{Y}_m|$ (according to class m) in the song and the number of model classes M leads to a valid multinomial distribution.

We find that the codeword assignment procedure outlined above tends to assign only a few different codewords to each song. In order to diversify the CBMs, we generalize equation 3 to support the assignment of multiple codewords at each point in the song. Hence, for a threshold $k \in \{1, 2, \ldots, |\mathcal{V}_m|\}$, we assign the k most likely codewords (again comparing only within a model class) to each fragment. The softened histogram is then constructed as:

$$b[i] = \frac{1}{M|\mathcal{Y}_m|} \sum_{y^t \in \mathcal{Y}_m} \frac{1}{k}\mathbb{1}[\Theta_i = \operatorname*{argmax}_{\Theta \in \mathcal{V}_m}{}^k P(y^t|\Theta)] \quad (4)$$

where the additional normalization factor of $1/k$ ensures that b is still a valid multinomial for $k > 1$.

3. MUSIC ANNOTATION AND RETRIEVAL USING THE BAG-OF-SYSTEMS REPRESENTATION

Once a BoS codebook $\mathcal{V}$ has been generated and songs are represented by codebook histograms (i.e., CBMs), a content-based auto-tagger may be obtained based on this representation — by modeling the characteristic codeword patterns in the CBMs of songs associated with each tag in a given vocabulary. In this section, we formulate annotation and retrieval as a multiclass multi-label classification of CBMs and discuss the algorithms used to learn tag models.

3.1 Annotation and retrieval with BoS histograms

Formally, assume we are given a training dataset $\mathcal{X}_t$, i.e., a collection of songs annotated with semantic tags from a vocabulary $\mathcal{T}$. Each song s in $\mathcal{X}_t$ is associated with a CBM $\mathbf{b}_s$ which describes the song's acoustic content with respect to the BoS codebook $\mathcal{V}$. The song s is also associated with an annotation vector $\mathbf{c}_s = (c_1, \ldots, c_{|\mathcal{T}|})$ which express the song's semantic content with respect to $\mathcal{T}$, where $c_i = 1$ if s has been annotated with tag $w_i \in \mathcal{T}$, and $c_i = 0$ otherwise. A dataset is a collection of CBM-annotation pairs $\mathcal{X}_t = \{(\mathbf{b}_s, \mathbf{c}_s)\}_{s=1}^{|\mathcal{X}_t|}$.

Given a training set $\mathcal{X}_t$, standard-text mining algorithms are used to learn tag-level models to capture which patterns in the CBMs are predictive for each tag in $\mathcal{T}$. Given the

CBM representation of a novel song, $\mathbf{b}$, we can then resort to the previously trained tag-models to compute how relevant each tag in $\mathcal{T}$ is to the song. In this work, we consider algorithms that have a probabilistic interpretation, for which it is natural to define probabilities $p(w_i|\mathbf{b})$, for $i = 1, \ldots, |T|$, which we rescale and aggregate to form a semantic multinomial (SMN) $\mathbf{p} = (p_1, \ldots, p_{|\mathcal{T}|})$, where $p_i \propto p(w_i|\mathbf{b})$ and $\sum_{i=1}^{|\mathcal{T}|} p_i = 1$. Hence we define the relevance of a tag to the song as the corresponding entry in the SMN.

Annotation involves selecting the most representative tags for a new song, and hence reduces to selecting the tags with highest entries in $\mathbf{p}$. Retrieval consists of rank ordering a set of songs $\mathcal{S} = \{s_1, s_2 \ldots s_R\}$ according to their relevance to a query. When the query is a single tag w_i from $\mathcal{T}$, we define the relevance of a song to the tag by $p(w_i|\mathbf{b})$, and therefore we rank the songs in the database based on the i^{th} entry in their SMN.

3.2 Learning tag-models from CBMs

The CBM representation of songs is amenable to a variety of annotation and retrieval algorithms. In this work, we investigate one generative algorithm, Codeword Bernoulli Average modeling (CBA), and one discriminative algorithm, multiclass kernel logistic regression (LR).

3.2.1 Codeword Bernoulli Average

The CBA model proposed by Hoffman et. al. [10] is a generative process that models the conditional probability of a tag word appearing in a song. Hoffman et al. define CBA based on a vector quantized codebook representation of songs. For our work, we adapt the CBA model to use a BoS codebook.

For each song, CBA defines a collection of binary random variables $y_w \in \{0, 1\}$, which determine whether or not tag w applies to the song. These variables are generated in two steps. First, given the song's CBM $\mathbf{b}$, a codeword z_w is chosen according to the CBM, i.e., $z_w \sim \text{Multinomial}(b_1, \ldots, b_{|\mathcal{V}|})$. Then a value for y_w is chosen from a Bernoulli distribution with parameter β_{kw},

$$p(y_w = 1|z_w, \beta) = \beta_{z_w w} \tag{5}$$

$$p(y_w = 0|z_w, \beta) = 1 - \beta_{z_w w}. \tag{6}$$

We use the author's code [10] to fit the CBA model. To build the SMN of a novel song we compute the posterior probabilities $p(y_{w_i} = 1|\mathbf{b}, \beta) = p_i$ under the estimated CBA model, and normalize $\mathbf{p} = (p_1, \ldots, p_{|\mathcal{V}|})$.

3.2.2 Multiclass Logistic Regression

Logistic regression defines a linear classifier with a probabilistic interpretation by fitting a logistic function to all CBMs associated to each tag:

$$P(w_i|\mathbf{b}, \beta_i) \propto \exp \beta_i^T \mathbf{b} \tag{7}$$

Kernel logistic regression finds a linear classifier after applying a non-linear transformation to the data, $\varphi : \mathbb{R}^d \to \mathbb{R}^{d_\varphi}$. The feature mapping φ is indirectly defined via a kernel function

$$K(\mathbf{a}, \mathbf{b}) = \langle \varphi(a), \varphi(b) \rangle, \tag{8}$$

where $\mathbf{a}$ and $\mathbf{b}$ are CBMs.

In our experiments we use the histogram intersection kernel [17], which is defined by the kernel function:

$$K(\mathbf{a}, \mathbf{b}) = \sum_j min(a_j, b_j). \tag{9}$$

In our implementation we use the software package Liblinear [8] and learn an L_2-regularized logistic regression model for each tag using the "one-vs-the rest" approach. As with CBA, we collect the posterior probabilities $p(w_i|\mathbf{b})$ and normalize to build the SMN.

4. EXPERIMENTAL SETUP

4.1 Music Datasets

The **CAL500** [19] dataset consists of 502 Western popular songs from 502 different artists. Each song-tag association has been evaluated by at least 3 humans, using a vocabulary of 149 tags. CAL500 provides binary annotations that can be safely considered hard-labels, i.e., $c_i = 1$ when a tag i applies to the song and 0 when the tag does not apply. We restrict our experiments to the 97 tags with at least 30 example songs. CAL500 experiments use 5-fold cross-validation where each song appears in the test set exactly once.

The **Swat10k** dataset [18] is a collection of over ten thousand songs from 4,597 different artists, weakly labeled from a vocabulary of over 500 tags. The song-tag associations are mined from Pandora's website. We restrict our experiments to the 55 tags in common with CAL500.

4.2 Codebook parameters

For our experiments, we build codebooks using three classes of generative models: one class of GMMs and two classes of DTMs at different time resolutions. To learn DTM codewords, we use feature vectors consisting of 34 Mel-frequency bins. The feature vectors used to learn GMM codewords are Mel-frequency cepstral coefficients appended with first and second derivatives (MFCC-delta). Window and fragment length for each class of codewords are specified in Table 1.

Model Class	Window length (η)	Fragment length	Fragment step (ν)
BoS-DTM$_1$	12 ms	726 ms	145 ms
BoS-DTM$_2$	93 ms	5.8 s	1.16 s
BoS-GMM$_1$	46 ms	46 ms	23 ms

Table 1. Time resolutions of model classes

4.3 Experiments

Our first experiment is cross-validation on CAL500, using the training set $\mathcal{X}_t$ as the codebook set $\mathcal{X}_c$ and re-training the codebook for each split. We learn $K_s = 4$ codewords of each model class per song. We build 5 codebooks: one for each of the 3 classes of codewords, one combining the two classes of DTM codewords (BoS-DTM$_{1,2}$) and one combining all three classes of codewords (BoS-DTM$_{1,2}$-GMM$_1$). These results are discussed in Section 5.1.

A second experiment investigates using a codebook set $\mathcal{X}_c$ that is disjoint from any of the training sets $\mathcal{X}_t$. By sampling $\mathcal{X}_c$ as a subset of the Swat10k dataset, we illustrate how a codebook may be learned from any collection of songs (whether annotated or not). Training and testing of tag models is still performed as five-fold cross-validation on CAL500. We perform one experiment with $|\mathcal{X}_c| = 400$, $K_s = 4$, to obtain a codebook of the same size as those learned on the CAL500 training set. Another experiment uses $|\mathcal{X}_c| = 4,597$, for which one song was chosen from each artist in Swat10k, and $K_s = 2$. The results are discussed in Section 5.2.

Finally, we conduct an experiment learning codebooks and training tag models on the Swat10k dataset and testing these models on CAL500, in order to determine how well the BoS approach adapts to training on a separate, weakly labeled dataset. We use the same codebook learned from one song from each artist in Swat10k as above, with $|\mathcal{X}_c| = 4,597$, and $K_s = 2$ codewords per song for each model class. Now our training set $\mathcal{X}_t$ is the entire Swat10k dataset. We train tag models with the settings (regularization of LR, etc.) found through cross-validation on CAL500, in order to avoid overfitting, and test these models on the CAL500 songs. These results are discussed in Section 5.3.

4.4 Annotation and retrieval

We annotate each test song CBM with 10 tags, as described in Section 3. Annotation performance is measured using mean per-tag precision, recall and F-score. Retrieval performance is measured using area under the receiver operating characteristic curve (AROC), mean average precision (MAP), and precision at 10 (P10) [19].

5. EXPERIMENTAL RESULTS

5.1 Results on CAL500

Results on the CAL500 dataset are shown in Table 2. In general, we achieve the best results with the softened histogram CBM representation (see Section 2.3), using a threshold of $k = 10$ for CBA and $k = 5$ for LR. For comparison we also show results using the hierarchical EM algorithm (HEM) to directly build GMM tag models (HEM-GMM) [19] and to directly build DTM tag models (HEM-DTM) [5]. These approaches are state of the art auto-tagging algorithms that use the same generative models we use to build BoS codebooks, in a more traditional framework. The HEM-GMM experiments use GMM tag models consisting of 4 mixture components, with the same audio features as the BoS-GMM$_1$ experiments. The HEM-DTM experiments use DTM tag models consisting of 16 mixture components with the same features and time scale parameters as the BoS-DTM$_2$ experiments. The BoS approach outperforms the direct tag modeling approach for all metrics except precision, where HEM-DTM is still best. Additionally, the greatest improvements are seen with codebooks that combine the richest variety of codewords. These codebooks capture the most information from the audio features, which leads to more descriptive tag models and increases the quality of the tag estimation.

		Annotation			**Retrieval**		
		Precision	Recall	F-Score	AROC	MAP	P10
HEM-GMM		0.374	0.205	0.213	0.686	0.417	0.425
HEM-DTM		**0.446**	0.217	0.264	0.708	0.446	0.460
BoS-DTM$_1$	CBA	0.369	0.251	0.237	0.722	0.465	0.482
	LR	0.416	0.257	0.270	0.730	0.471	0.483
BoS-DTM$_2$	CBA	0.382	0.241	0.233	0.717	0.457	0.471
	LR	0.404	0.251	0.260	0.725	0.466	0.480
BoS-GMM$_1$	CBA	0.359	0.243	0.227	0.714	0.450	0.463
	LR	0.396	0.251	0.257	0.724	0.464	0.479
BoS-DTM$_{1,2}$	CBA	0.375	0.254	0.240	0.729	0.473	0.495
	LR	0.413	0.264	0.274	0.738	0.480	0.496
BoS–DTM$_{1,2}$-GMM$_1$	CBA	0.378	0.262	0.248	0.738	0.482	0.505
	LR	0.434	**0.272**	**0.281**	**0.748**	**0.493**	**0.508**

Table 2. BoS codebook performance on CAL500, compared to Gaussian tag modeling (HEM-GMM) and DTM tag modeling (HEM-DTM).

Since the classification algorithms we use to model tags have fewer parameters than direct tag modeling approaches, the BoS approach is more robust for tags with fewer example songs. We demonstrate this in Figure 1, which plots the improvement in MAP over HEM-DTM as a function of the tag's training set cardinality. The BoS approach shows the greatest improvement for tags with few training examples.

5.2 Results learning codebook from unlabeled songs

Table 3 shows results using BoS codebooks learned from unlabeled songs. These results are roughly equivalent to using codebooks learned from CAL500, and in fact outperform the CAL500 codebooks with a larger codebook set. This shows that a dictionary of musically meaningful codewords may be estimated from *any* large collection of songs, which need not be labeled, and that a performance gain can be achieved by adding unlabeled songs to the codebook set.

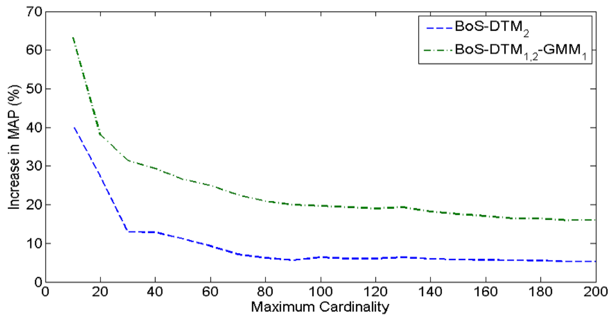

Figure 1. Retrieval performance of the BoS approach with LR, relative to HEM-DTM, as a function of the maximum cardinality of tag subsets. For each point in the graph, the set of all CAL500 tags is restricted to those associated with a number of songs that is at most the abscissa value.

	$\lvert\mathcal{X}_c\rvert$		**Annotation** Precision	Recall	F-score	**Retrieval** AROC	MAP	P10
CAL500	400	CBA	0.378	0.262	0.248	0.738	0.482	0.505
		LR	**0.434**	0.272	0.281	0.748	0.493	0.508
Swat10K	400	CBA	0.355	0.263	0.244	0.741	0.484	0.505
		LR	0.429	0.269	0.277	0.749	0.492	0.498
	4,597	CBA	0.377	0.263	0.249	0.744	0.489	0.505
		LR	**0.434**	**0.273**	**0.282**	**0.751**	**0.497**	**0.517**

Table 3. Results using codebooks learned from unlabeled data (Swat10k), compared with codebooks from CAL500, with codewords from model classes BoS-DTM$_{1,2}$-GMM$_1$, where $\lvert\mathcal{X}_c\rvert$ is the cardinality of the codebook training set.

5.3 Results training on Swat10k

Results training codebooks and tag models on the Swat10k dataset, in Table 4, show that the BoS approach still outperforms the direct tag modeling approaches when trained on a separate dataset. We also see that the generative CBA model catches up to the discriminative LR model in some performance metrics, which is expected, since generative models tend to be more robust on weakly labeled datasets.

6. CONCLUSION

We have presented a semantic auto-tagger that leverages a rich "bag of systems" representation of music. The latter can be learned from any representative set of songs, which need not be annotated, and allows to integrate the descriptive quality of various generative models of musical content, with different time resolutions. This approach improves performance over directly modeling tags with a single type of generative model. It also proves significantly more robust for tags with few training examples.

7. ACKNOWLEDGMENTS

The authors thank L. Barrington and M. Hoffman for providing the code of [19] and [10] respectively, and acknowledge support from Qualcomm, Inc., Yahoo! Inc., the Hellman Fellowship Program, and NSF Grants CCF-0830535 and IIS-1054960.

		Annotation Precision	Recall	F-Score	**Retrieval** AROC	MAP	P10
HEM-GMM		0.297	0.404	0.264	0.714	0.350	0.315
HEM-DTM		0.289	0.391	0.259	0.702	0.354	0.314
BoS-DTM$_{1,2}$-GMM$_1$	CBA	0.310	**0.495**	0.295	0.756	**0.414**	**0.361**
	LR	**0.336**	0.493	**0.319**	**0.757**	**0.414**	0.353

Table 4. Summary of results training on Swat10k.

8. REFERENCES

[1] D. Aldous. Exchangeability and related topics. 1985.

[2] Michael Casey, Christophe Rhodes, and Malcolm Slaney. Analysis of minimum distances in high-dimensional musical spaces. *IEEE Transactions on Audio, Speech and Language Processing*, 16(5):1015–1028, 2008.

[3] A. B. Chan and N. Vasconcelos. Modeling, clustering, and segmenting video with mixtures of dynamic textures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):909–926, 2008.

[4] A.B. Chan, E. Coviello, and G. Lanckriet. Clustering dynamic textures with the hierarchical EM algorithm. In *Proc. IEEE CVPR*, 2010.

[5] E. Coviello, A. Chan, and G. Lanckriet. Time Series Models for Semantic Music Annotation. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(5):1343–1359, July 2011.

[6] G. Doretto, A. Chiuso, Y. N. Wu, and S. Soatto. Dynamic textures. *Intl. J. Computer Vision*, 51(2):91–109, 2003.

[7] D. Eck, P. Lamere, T. Bertin-Mahieux, and S. Green. Automatic generation of social tags for music recommendation. In *Advances in Neural Information Processing Systems*, 2007.

[8] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[9] M. Hoffman, D. Blei, and P. Cook. Content-based musical similarity computation using the hierarchical Dirichlet process. In *Proc. ISMIR*, pages 349–354, 2008.

[10] M. Hoffman, D. Blei, and P. Cook. Easy as CBA: A simple probabilistic model for tagging music. In *Proc. ISMIR*, pages 369–374, 2009.

[11] M.I. Mandel and D.P.W. Ellis. Multiple-instance learning for music information retrieval. In *Proc. ISMIR*, pages 577–582, 2008.

[12] S.R. Ness, A. Theocharis, G. Tzanetakis, and L.G. Martins. Improving automatic music tag annotation using stacked generalization of probabilistic svm outputs. In *Proc. ACM MULTIMEDIA*, pages 705–708, 2009.

[13] E. Pampalk, A. Flexer, and G. Widmer. Improvements of audio-based music similarity and genre classification. In *Proc. ISMIR*, pages 628–633, 2005.

[14] A. Ravichandran, R. Chaudhry, and R. Vidal. View-invariant dynamic texture recognition using a bag of dynamical systems. In *CVPR*, 2009.

[15] J. Reed and C.H. Lee. A study on music genre classification based on universal acoustic models. In *Proc. ISMIR*, pages 89–94, 2006.

[16] D.E. Sturim, DA Reynolds, E. Singer, and JP Campbell. Speaker indexing in large audio databases using anchor models. In *icassp*, pages 429–432. IEEE, 2001.

[17] M.J. Swain and D.H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[18] Derek Tingle, Youngmoo E. Kim, and Douglas Turnbull. Exploring automatic music annotation with "acoustically-objective" tags. In *Proc. MIR*, pages 55–62, New York, NY, USA, 2010. ACM.

[19] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech and Language Processing*, 16(2):467–476, February 2008.

[20] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002.