

EASY AS CBA: A SIMPLE PROBABILISTIC MODEL FOR TAGGING MUSIC

Matthew D. Hoffman

Dept. of Computer Science
Princeton University

mdhoffma at cs.princeton.edu

David M. Blei

Dept. of Computer Science
Princeton University

blei at cs.princeton.edu

Perry R. Cook

Dept. of Computer Science
Dept. of Music
Princeton University

prc at cs.princeton.edu

ABSTRACT

Many songs in large music databases are not labeled with semantic tags that could help users sort out the songs they want to listen to from those they do not. If the words that apply to a song can be predicted from audio, then those predictions can be used both to automatically annotate a song with tags, allowing users to get a sense of what qualities characterize a song at a glance. Automatic tag prediction can also drive retrieval by allowing users to search for the songs most strongly characterized by a particular word. We present a probabilistic model that learns to predict the probability that a word applies to a song from audio. Our model is simple to implement, fast to train, predicts tags for new songs quickly, and achieves state-of-the-art performance on annotation and retrieval tasks.

1. INTRODUCTION

It has been said that talking about music is like dancing about architecture, but people nonetheless use words to describe music. In this paper we will present a simple system that addresses tag prediction from audio—the problem of predicting what words people would be likely to use to describe a song.

Two direct applications of tag prediction are semantic annotation and retrieval. If we have an estimate of the probability that a tag applies to a song, then we can say what words in our vocabulary of tags best describe a given song (automatically annotating it) and what songs in our database a given word best describes (allowing us to retrieve songs from a text query).

We present the Codeword Bernoulli Average (CBA) model, a probabilistic model that attempts to predict the probability that a tag applies to a song based on a vector-quantized (VQ) representation of that song’s audio. Our CBA-based approach to tag prediction

- Is easy to implement using a simple EM algorithm.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page.

© 2009 International Society for Music Information Retrieval.

- Is fast to train.
- Makes predictions efficiently on unseen data.
- Performs as well as or better than state-of-the-art approaches.

2. DATA REPRESENTATION

2.1 The CAL500 data set

We train and test our method on the CAL500 dataset [1, 2]. CAL500 is a corpus of 500 tracks of Western popular music, each of which has been manually annotated by at least three human labelers. We used the “hard” annotations provided with CAL500, which give a binary value $y_{jw} \in \{0, 1\}$ for all songs j and tags w indicating whether tag w applies to song j .

CAL500 is distributed with a set of 10,000 39-dimensional Mel-Frequency Cepstral Coefficient Delta (MFCC-Delta) feature vectors for each song. Each Delta-MFCC vector summarizes the timbral evolution of three successive 23ms windows of a song. CAL500 provides these feature vectors in a random order, so no temporal information beyond a 69ms timescale is available.

Our goals are to use these features to predict which tags apply to a given song and which songs are characterized by a given tag. The first task yields an automatic annotation system, the second yields a semantic retrieval system.

2.2 A vector-quantized representation

Rather than work directly with the MFCC-Delta feature representation, we first vector quantize all of the feature vectors in the corpus, ignoring for the moment what feature vectors came from what songs. We:

1. Normalize the feature vectors so that they have mean 0 and standard deviation 1 in each dimension.
2. Run the k-means algorithm [3] on a subset of randomly selected feature vectors to find a set of K cluster centroids.
3. For each normalized feature vector f_{ji} in song j , assign that feature vector to the cluster k_{ji} with the smallest squared Euclidean distance to f_{ji} .

This vector quantization procedure allows us to represent each song j as a vector \mathbf{n}_j of counts of a discrete set of codewords:

$$n_{jk} = \sum_{i=1}^{N_j} 1(k_{ji} = k) \quad (1)$$

where n_{jk} is the number of feature vectors assigned to codeword k , N_j is the total number of feature vectors in song j , and $1(a = b)$ is a function returning 1 if $a = b$ and 0 if $a \neq b$.

This discrete “bag-of-codewords” representation is less rich than the original continuous feature vector representation. However, it is effective. Such VQ codebook representations have produced state-of-the-art performance in image annotation and retrieval systems [4], as well as in systems for estimating timbral similarity between songs [5,6].

3. THE CODEWORD BERNOULLI AVERAGE MODEL

In order to predict what tags will apply to a song and what songs are characterized by a tag, we developed the Codeword Bernoulli Average model (CBA). CBA models the conditional probability of a tag w appearing in a song j conditioned on the empirical distribution \mathbf{n}_j of codewords extracted from that song. Once we have estimated CBA’s hidden parameters from our training data, we will be able to quickly estimate this conditional probability for new songs.

3.1 Related work

One class of approaches treats audio tag prediction as a set of binary classification problems to which variants of standard classifiers such as the Support Vector Machine (SVM) [7,8] or AdaBoost [9] can be applied. Once a set of classifiers has been trained, the classifiers attempt to predict whether or not each tag applies to previously unseen songs. These predictions come with confidence scores that can be used to rank songs by relevance to a given tag (for retrieval), or tags by relevance to a given song (for annotation). Classifiers like SVMs or AdaBoost focus on binary classification accuracy rather than directly optimizing the continuous confidence scores that are used for retrieval tasks, which might lead to suboptimal results for those tasks.

Another approach is to fit a generative probabilistic model such as a Gaussian Mixture Model (GMM) for each tag to the audio feature data for all of the songs manifesting that tag [2]. The posterior likelihood $p(\text{tag}|\text{audio})$ of the feature data for a new song being generated from the model for a particular tag is then used to estimate the relevance of that tag to that song (and vice versa). Although this model tells us how to generate the audio feature data for a song conditioned on a *single* tag, it does not define a generative process for songs with *multiple* tags, and so heuristics are necessary to estimate the posterior likelihood of a set of tags.

Rather than assuming that the audio for a song depends on the tags associated with that song, we will assume that

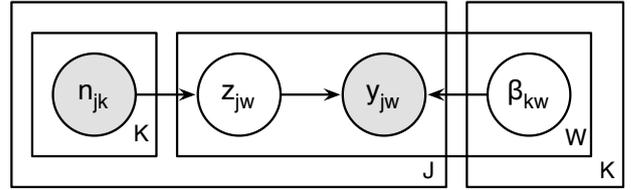


Figure 1. Graphical model representation of CBA. Shaded nodes represent observed variables, unshaded nodes represent hidden variables. A directed edge from node a to node b denotes that the variable b depends on the value of variable a . Plates (boxes) denote replication by the value in the lower right of the plate. J is the number of songs, K is the number of codewords, and W is the number of unique tags.

the tags depend on the audio data. This will yield a probabilistic model with a discriminative flavor, and a more coherent generative process than that in [2].

3.2 Generative process

CBA assumes a collection of binary random variables \mathbf{y} , with $y_{jw} \in \{0, 1\}$ determining whether or not tag w applies to song j . These variables are generated in two steps. First, a codeword $z_{jw} \in \{1, \dots, K\}$ is selected with probability proportional to the number of times n_{jk} that that codeword appears in song j ’s feature data:

$$p(z_{jw} = k | \mathbf{n}_j, N_j) = \frac{n_{jk}}{N_j} \quad (2)$$

Then a value for y_{jw} is chosen from a Bernoulli distribution with parameter β_{kw} :

$$\begin{aligned} p(y_{jw} = 1 | z_{jw}, \beta) &= \beta_{z_{jw}w} \\ p(y_{jw} = 0 | z_{jw}, \beta) &= 1 - \beta_{z_{jw}w} \end{aligned} \quad (3)$$

The full joint distribution over \mathbf{z} and \mathbf{y} conditioned on the observed counts of codewords \mathbf{n} is:

$$p(\mathbf{z}, \mathbf{y} | \mathbf{n}) = \prod_w \prod_j \frac{n_{jz_{jw}}}{N_j} \beta_{z_{jw}w} \quad (4)$$

The random variables in CBA and their dependencies are summarized in figure 1.

3.3 Inference using expectation-maximization

We fit CBA with maximum-likelihood (ML) estimation. Our goal is to estimate a set of values for our Bernoulli parameters β that will maximize the likelihood $p(\mathbf{y} | \mathbf{n}, \beta)$ of the observed tags \mathbf{y} conditioned on the VQ codeword counts \mathbf{n} and the parameters β . Analytic ML estimates for β are not available because of the latent variables \mathbf{z} . We use the Expectation-Maximization (EM) algorithm, a widely used coordinate ascent algorithm for maximum-likelihood estimation in the presence of latent variables [10].

Each iteration of EM operates in two steps. In the expectation (“E”) step, we compute the posterior of the latent

variables \mathbf{z} given our current estimates for the parameters β . We define a set of expectation variables h_{jwk} corresponding to the posterior $p(z_{jw} = k | \mathbf{n}, \mathbf{y}, \beta)$:

$$h_{jwk} = p(z_{jw} = k | \mathbf{n}, \mathbf{y}, \beta) \quad (5)$$

$$= \frac{p(y_{jw} | z_{jw} = k, \beta) p(z_{jw} = k | \mathbf{n})}{p(y_{jw} | \mathbf{n}, \beta)} \quad (6)$$

$$= \begin{cases} \frac{n_{jk} \beta_{kw}}{\sum_{i=1}^K n_{ji} \beta_{iw}} & \text{if } y_{jw} = 1 \\ \frac{n_{jk} (1 - \beta_{kw})}{\sum_{i=1}^K n_{ji} (1 - \beta_{iw})} & \text{if } y_{jw} = 0 \end{cases} \quad (7)$$

In the maximization (“M”) step, we find maximum-likelihood estimates of the parameters β given the expected posterior sufficient statistics:

$$\beta_{kw} \leftarrow \mathbb{E}[y_{jw} | z_{jw} = k, \mathbf{h}] \quad (8)$$

$$= \frac{\sum_j p(z_{jw} = k | \mathbf{h}) y_{jw}}{\sum_j p(z_{jw} = k | \mathbf{h})} \quad (9)$$

$$= \frac{\sum_j h_{jwk} y_{jw}}{\sum_j h_{jwk}} \quad (10)$$

By iterating between computing \mathbf{h} (using equation 7) and updating β (using equation 10), we find a set of values for β under which our training data become more likely.

3.4 Generalizing to new songs

Once we have inferred a set of Bernoulli parameters β from our training dataset, we can use them to infer the probability that a tag w will apply to a previously unseen song j based on the counts \mathbf{n}_j of codewords for that song:

$$\begin{aligned} p(y_{jw} | \mathbf{n}_j, \beta) &= \sum_k p(z_{jw} = k | \mathbf{n}_j) p(y_{jw} | z_{jw} = k) \\ p(y_{jw} = 1 | \mathbf{n}_j, \beta) &= \frac{1}{N_j} \sum_k n_{jk} \beta_{kw} \end{aligned} \quad (11)$$

As a shorthand, we will refer to our inferred value of $p(y_{jw} = 1 | \mathbf{n}_j, \beta)$ as s_{jw} .

Once we have inferred s_{jw} for all of our songs and tags, we can use these inferred probabilities both to retrieve the songs with the highest probability of having a particular tag and to annotate each song with a subset of our vocabulary of tags. In a retrieval system, we return the songs in descending order of s_{jw} . To do automatic tagging, we could annotate each song with the M most likely tags for that song. However, this may lead to our annotating many songs with common, uninformative tags such as “Not Bizarre/Weird” and a lack of diversity in our annotations. To compensate for this, we use a simple heuristic: we introduce a “diversity factor” d and discount each s_{jw} by d times the mean of the estimated probabilities $s_{\cdot w}$. A higher value of d will make less common tags more likely to appear in annotations, which may lead to less accurate but more informative annotations. The diversity factor d has no impact on retrieval.

The cost of computing each s_{jw} using equation 11 is linear in the number of codewords K , and the cost of vector quantizing new songs’ feature data using the previously

computed centroids obtained using k-means is linear in the number of features, the number of codewords K , and the length of the song. For practical values of K , the total cost of estimating the probability that a tag applies to a song is comparable to the cost of feature extraction. Our approach can therefore tag new songs efficiently, an important feature for large commercial music databases.

4. EVALUATION

We evaluated our model’s performance on an annotation task and a retrieval task using the CAL500 data set. We compare our results on these tasks with two other sets of published results for these tasks on this corpus: those obtained by Turnbull et al. using mixture hierarchies estimation to learn the parameters to a set of mixture-of-Gaussians models [2], and those obtained by Bertin-Mahieux et al. using a discriminative approach based on the AdaBoost algorithm [9]. In the 2008 MIREX audio tag classification task, the approach in [2] was ranked either first or second according to all metrics measuring annotation or retrieval performance [11].

4.1 Annotation task

To evaluate our model’s ability to automatically tag unlabeled songs, we measured its average per-word precision and recall on held-out data using tenfold cross-validation.

First, we vector quantized our data using k-means. We tested VQ codebook sizes from $K = 5$ to $K = 2500$. After finding a set of K centroids using k-means on a randomly chosen subset of 125,000 of the Delta-MFCC vectors (250 feature vectors per song), we labeled each Delta-MFCC vector in each song with the index of the cluster centroid whose squared Euclidean distance to that vector was smallest. Each song j was then represented as a K -dimensional vector \mathbf{n}_j , with n_{jk} giving the number of times label k appeared in song j , as described in equation 1.

We ran a tenfold cross-validation experiment modeled after the experiments in [2]. We split our data into 10 disjoint 50-song test sets at random, and for each test set

1. We iterated the EM algorithm described in section 3.3 on the remaining 450 songs to estimate the parameters β . We stopped iterating once the negative log-likelihood of the training labels conditioned on β and \mathbf{n} decreased by less than 0.1% per iteration.
2. Using equation 11, for each tag w and each song j in the test set we estimated $p(y_{jw} | \mathbf{n}_j, \beta)$, the probability of song j being characterized by tag w conditioned on β and the vector quantized feature data \mathbf{n}_j .
3. We subtracted $d = 1.25$ times the average conditional probability of tag w from our estimate of $p(y_{jw} | \mathbf{n}_j, \beta)$ for each song j to get a score s_{jw} for each song.
4. We annotated each song j with the ten tags with the highest scores s_{jw} .

Model	Precision	Recall	F-Score	AP	AROC
UpperBnd	0.712 (0.007)	0.375 (0.006)	0.491	1	1
Random	0.144 (0.004)	0.064 (0.002)	0.089	0.231 (0.004)	0.503 (0.004)
MixHier	0.265 (0.007)	0.158 (0.006)	0.198	0.390 (0.004)	0.710 (0.004)
Autotag (MFCC)	0.281	0.131	0.179	0.305	0.678
Autotag (afeats exp.)	0.312	0.153	0.205	0.385	0.674
CBA $K = 5$	0.198 (0.006)	0.107 (0.005)	0.139	0.328 (0.009)	0.707 (0.007)
CBA $K = 10$	0.214 (0.006)	0.111 (0.006)	0.146	0.336 (0.007)	0.715 (0.007)
CBA $K = 25$	0.247 (0.007)	0.134 (0.007)	0.174	0.352 (0.008)	0.734 (0.008)
CBA $K = 50$	0.257 (0.009)	0.145 (0.007)	0.185	0.366 (0.009)	0.746 (0.008)
CBA $K = 100$	0.263 (0.007)	0.149 (0.004)	0.190	0.372 (0.007)	0.748 (0.008)
CBA $K = 250$	0.279 (0.007)	0.153 (0.005)	0.198	0.385 (0.007)	0.760 (0.007)
CBA $K = 500$	0.286 (0.005)	0.162 (0.004)	0.207	0.390 (0.008)	0.759 (0.007)
CBA $K = 1000$	0.283 (0.008)	0.161 (0.006)	0.205	0.393 (0.008)	0.764 (0.006)
CBA $K = 2500$	0.282 (0.006)	0.162 (0.004)	0.206	0.394 (0.008)	0.765 (0.007)

Table 1. Summary of the performance of CBA (with a variety of VQ codebook sizes K), a mixture-of-Gaussians model (MixHier), and an AdaBoost-based model (Autotag) on an annotation task (evaluated using precision, recall, and F-score) and a retrieval task (evaluated using average precision (AP) and area under the receiver-operator curve (AROC)). Autotag (MFCC) used the same Delta-MFCC feature vectors and training set size of 450 songs as CBA and MixHier. Autotag (afeats exp.) used a larger set of features and a larger set of training songs. UpperBnd uses the optimal labeling for each evaluation metric, and shows the upper limit on what any system can achieve. Random is a baseline that annotates and ranks songs randomly.

To evaluate our system’s annotation performance, we computed the average per-word precision, recall, and F-score. Per-word recall is defined as the average fraction of songs actually labeled w that our model annotates with label w . Per-word precision is defined as the average fraction of songs that our model annotates with label w that are actually labeled w . F-score is the harmonic mean of precision and recall, and is one metric of overall annotation performance.

Following [2], when our model does not annotate any songs with a label w we set the precision for that word to be the empirical probability that a word in the dataset is labeled w . This is the expected per-word precision for w if we annotate all songs randomly. If no songs in a test set are labeled w , then per-word precision and recall for w are undefined, so we ignore these words in our evaluation.

4.2 Retrieval task

To evaluate our system’s retrieval performance, for each tag w we ranked each song j in the test set by the probability our model estimated of tag w applying to song j . We evaluated the average precision (AP) and area under the receiver-operator curve (AROC) for each ranking. AP is defined as the average of the precisions at each possible level of recall, and AROC is defined as the area under a curve plotting the percentage of true positives returned against the percentage of false positives returned. As in the annotation task, if no songs in a test set are labeled w then AP and AROC are undefined for that label, and we exclude it from our evaluation for that fold of cross-validation.

4.3 Annotation and retrieval results

Table 1 and figure 2 compare our CBA model’s average performance under the five metrics described above with other published results on the same dataset. MixHier is Turnbull et al.’s system based on a mixture-of-Gaussians model [2], Autotag (MFCC) is Bertin-Mahieux’s AdaBoost-based system using the same Delta-MFCC feature vectors as our model, and Autotag (afeats exp.) is Bertin-Mahieux’s system trained using additional features and training data [9]. Random is a random baseline that retrieves songs in a random order and annotates songs randomly based on tags’ empirical frequencies. UpperBnd shows the best performance possible under each metric. Random and UpperBnd were computed by Turnbull et al., and give a sense of the possible range for each metric.

We tested our model using a variety of codebook sizes K from 5 to 2500. Cross-validation performance improves as the codebook size increases until $K = 500$, at which point it levels off. Our model’s performance does not depend strongly on fine tuning K , at least within a range of $500 \leq K \leq 2500$.

When using a codebook size of at least 500, our CBA model does at least as well as MixHier and Autotag under every metric except precision. Autotag gets significantly higher precision than CBA when it uses additional training data and features, but not when it uses the same features and training set as CBA.

Tables 2 and 3 give examples of annotations and retrieval results given by our model during cross-validation.

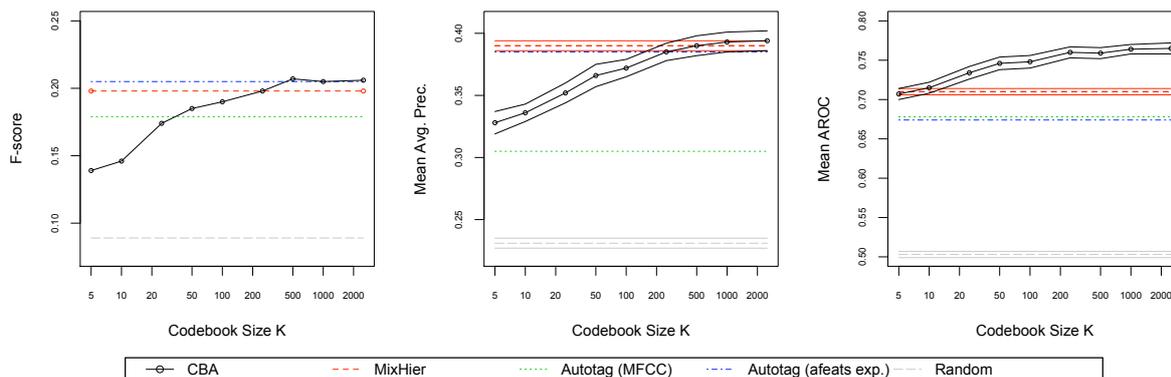


Figure 2. Visual comparison of the performance of several models evaluated using F-score, mean average precision, and area under receiver-operator curve (AROC).

4.4 Computational cost

We measured how long it took to estimate the parameters to CBA and to generalize to new songs. All experiments were conducted on one core of a server with a 2.2 GHz AMD Opteron 275 CPU and 16 GB of RAM running CentOS Linux.

Using a MATLAB implementation of the EM algorithm described in 3.3, it took 84.6 seconds to estimate CBA’s parameters from 450 training songs vector-quantized using a 500-cluster codebook. In experiments with other codebook sizes K the training time scaled linearly with K . Once β had been estimated, it took less than a tenth of a millisecond to predict the probabilities of 174 labels for a new song.

We found that the vector-quantization process was the most expensive part of training and applying CBA. Finding a set of 500 cluster centroids from 125,000 39-dimensional Delta-MFCC vectors using a C++ implementation of k-means took 479 seconds, and finding the closest of 500 cluster centroids to the 10,000 feature vectors in a song took 0.454 seconds. Both of these figures scaled linearly with the size of the VQ codebook in other experiments.

5. DISCUSSION AND FUTURE WORK

We introduced the Codeword Bernoulli Average model, which predicts the probability that a tag will apply to a song based on counts of vector-quantized feature data extracted from that song. Our model is simple to implement, fast to train, generalizes to new songs efficiently, and yields state-of-the-art performance on annotation and semantic retrieval tasks.

We plan to explore several extensions to this model in the future. In place of the somewhat ad hoc diversity factor, one could use a weighting similar to TF-IDF to choose informative words for annotations. The vector quantization preprocessing stage could be replaced with a mixed-membership mixture-of-Gaussians model that could be fit simultaneously with β . Also, we hope to explore principled ways of incorporating song-level feature data describing information not captured by MFCCs, such as rhythm.

Query	Top 5 Retrieved Songs
Tender/Soft	John Lennon—Imagine Shira Kammen—Music of Waters Crosby Stills and Nash—Guinnevere Jewel—Enter From the East Yakshi—Chandra
Hip Hop	Tim Rayborn—Yedi Tekrar Solace—Laz 7 8 Eminem—My Fault Sir Mix-a-Lot—Baby Got Back 2-Pac—Trapped
Piano	Robert Johnson—Sweet Home Chicago Shira Kammen—Music of Waters Miles Davis—Blue in Green Guns n’ Roses—November Rain Charlie Parker—Ornithology
Exercising	Tim Rayborn—Yedi Tekrar Monoide—Golden Key Introspekt—TBD Belief Systems—Skunk Werks Solace—Laz 7 8
Screaming	Nova Express—I’m Alive Rocket City Riot—Mine Tonite Seismic Anamoly—Wreckinball Pizzle—What’s Wrong With My Footman Jackalopes—Rotgut

Table 3. Examples of semantic retrieval from the CAL500 data set. The left column shows a query word, and the right column shows the five songs in the dataset judged by our system to best match that word.

6. REFERENCES

- [1] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Towards musical query-by-semantic-description using the CAL500 data set. In *Proc. ACM SIGIR*, pages 439–446, 2007.
- [2] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Semantic annotation and retrieval of music and

<i>Give it Away</i> the Red Hot Chili Peppers	<i>Fly Me to the Moon</i> Frank Sinatra	<i>Blue Monday</i> New Order	<i>Becoming</i> Pantera
Usage—At a party Heavy Beat Drum Machine Rapping Very Danceable Genre—Hip Hop/Rap Genre (Best)—Hip Hop/Rap Texture Synthesized Arousing/Awakening Exciting/Thrilling	Calming/Soothing NOT—Fast Tempo NOT—High Energy Laid-back/Mellow Tender/Soft NOT—Arousing/Awakening Usage—Going to sleep Usage—Romancing NOT—Powerful/Strong Sad	Very Danceable Usage—At a party Heavy Beat Arousing/Awakening Fast Tempo Drum Machine Texture Synthesized Sequencer Genre—Hip Hop/Rap Synthesizer	NOT—Calming/Soothing NOT—Tender/Soft NOT—Laid-back/Mellow Bass Genre—Alternative Exciting/Thrilling Electric Guitar (distorted) Genre—Rock Texture Electric High Energy

Table 2. Examples of semantic annotation from the CAL500 data set. The two columns show the top 10 words associated by our model with the songs *Give it Away*, *Fly Me to the Moon*, *Blue Monday*, and *Becoming*.

sound effects. *IEEE Transactions on Audio Speech and Language Processing*, 16(2), 2008.

- [3] J.B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. Fifth Berkeley Symp. on Math. Statist. and Prob., Vol. 1*, 1966.
- [4] C. Wang, D. Blei, and L. Fei-Fei. Simultaneous image classification and annotation. In *Proc. IEEE CVPR*, 2009.
- [5] M. Hoffman, D. Blei, and P. Cook. Content-based musical similarity computation using the hierarchical Dirichlet process. In *Proc. International Conference on Music Information Retrieval*, 2008.
- [6] K. Seyerlehner, A. Linz, G. Widmer, and P. Knees. Frame level audio similarity—a codebook approach. In *Proc. of the 11th Int. Conference on Digital Audio Effects (DAFx08), Espoo, Finland, September, 2008*.
- [7] M. Mandel and D. Ellis. LabROSA’s audio classification submissions, mirex 2008 website. http://www.music-ir.org/mirex/2008/abs/AA_AG_AT_MM_CC_mandel.pdf.
- [8] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas. Multilabel classification of music into emotions. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR)*, 2008.
- [9] T. Bertin-Mahieux, D. Eck, F. Maillet, and P. Lamere. Autotagger: a model for predicting social tags from acoustic features on large music databases. *Journal of New Music Research*, 37(2):115–135, 2008.
- [10] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.
- [11] Mirex 2008 website. http://www.music-ir.org/mirex/2008/index.php/Main_Page.