Music Genre Classification with the Million Song Dataset

15-826 Final Report

Dawen Liang,† Haijie Gu,‡ and Brendan O'Connor‡

† School of Music, ‡ Machine Learning Department

Carnegie Mellon University

December 3, 2011

1 Introduction

The field of Music Information Retrieval (MIR) draws from musicology, signal processing, and artificial intelligence. A long line of work addresses problems including: music understanding (extract the musically-meaningful information from audio waveforms), automatic music annotation (measuring song and artist similarity), and other problems.

However, very little work has scaled to commercially sized data sets. The algorithms and data are both complex. An extraordinary range of information is hidden inside of music waveforms, ranging from perceptual to auditory—which inevitably makes large-scale applications challenging. There are a number of commercially successful online music services, such as Pandora, Last.fm, and Spotify, but most of them are merely based on traditional text IR.

Our course project focuses on large-scale data mining of music information with the recently released Million Song Dataset (Bertin-Mahieux et al., 2011),[1] which consists of

[1] http://labrosa.ee.columbia.edu/millionsong/

300GB of audio features and metadata. This dataset was released to push the boundaries of Music IR research to commercial scales. Also, the associated musiXmatch dataset[2] provides textual lyrics information for many of the MSD songs. Combining these two datasets, we propose a cross-modal retrieval framework to combine the music and textual data for the task of *genre classification*:

Given N song-genre pairs: $(S_1, G_N), \ldots, (S_N, G_N)$, where $S_i \in \mathcal{F}$ for some feature space $\mathcal{F}$, and $G_i \in \mathcal{G}$ for some genre set $\mathcal{G}$, output the classifier with the highest classification accuracy on the hold-out test set. The raw feature space $\mathcal{F}$ contains multiple domains of sub features which can be of variable length. The genre label set $\mathcal{G}$ is discrete.

1.1 Motivation

Genre classification is a standard problem in Music IR research. Most of the music genre classification techniques employ pattern recognition algorithms to classify feature vectors, extracted from short-time recording segments into genres. Commonly used classifiers are Support Vector Machines (SVMs), Nearest-Neighbor (NN) classifiers, Gaussian Mixture Models, Linear Discriminant Analysis (LDA), etc. Several common audio datasets have been used in experiments to make the reported classification accuracies comparable, for example, the GTZAN dataset (Tzanetakis and Cook, 2002) which is the most widely used dataset for music genre classification.

However, the datasets involved in those studies are very small comparing to the Million Song Dataset. In fact, most of the Music IR research still focuses on very small datasets, such as the GTZAN dataset (Tzanetakis and Cook, 2002) with only 1000 audio tracks, each 30 seconds long; or CAL-500 (Turnbull et al., 2008), a set of 1700 human-generated musical annotations describing 500 popular western musical tracks. Both of these datasets are widely used in most state-of-the-art research in Music IR, but are far away from practical application.

Furthermore, most of the research on genre classification focuses only on music features, ignoring lyrics (mostly due to the difficulty of collecting large-scale lyric data).

[2] http://labrosa.ee.columbia.edu/millionsong/musixmatch

Nevertheless, besides the musical features (styles, forms), the genre is also closely related to lyrics—songs in different genres may involve different topics or moods, which could be recoverable from word frequencies in lyrics. This motivates us to join the musical and lyrics information from two databases for this task.

1.2 Contribution

To the best of our knowledge, there have been no published works that perform large-scale genre classification using cross-modal methods.

- We proposed a cross-modal retrival framework of *model blending* which combines features of lyrics and audio. The algorithm scales linearly in the number of songs, and evaluate it on a subset of the MSD containing 156,289 songs.
- Our likelihood-based submodel training (Section 3), combining audio sequence features and text features, is novel. There is no previous work on genre classification measuring the likelihood of different genre-based HMMs, or bag-of-words lyric features.
- Finally, we also experimented with methods which have not appeared before in genre classification, such as the spectral method for training HMM's (Section 3.4), and using Canonical Correlation Analysis (CCA) (Section 3.5) to combine features from different domains. Although our results show they do not outperform the state-of-art methods in this particular problem domain, it is worth investigating thoroughly to understand why.

"Double dipping" statement: This project is not related to any of the co-authors' dissertations. Dawen Liang is currently a second-year master's student with no dissertation. His master thesis will be about rehearsal audio segmentation and clustering. Haijie Gu and Brendan O'Connor are first year and second year Ph.D. students, respectively, in the Machine Learning Department, who have not started preparing their dissertation work.

In the following section, we describe the data set used in this project. In Section 3 we present the high-level cross-modal framework and the specific algorithms used for

Genre	Training	Tuning	Test
classic pop and rock	42681	208	1069
classical	3662	200	1027
dance and electronica	8222	241	1003
folk	17369	248	1013
hip-hop †	7909	261	1040
jazz	6478	220	1030
metal	8309	217	1054
pop	8873	231	1046
rock and indie	34972	238	1012
soul and reggae *	5114	249	1093
Totals	143,589	2,313	10,387

Table 1: Genres used for classification experiments, with the number of songs in training, tuning, and test splits. Their names in this table correspond to MusicBrainz tags. Alternate tag names: (*): *soul*, *reggae* (†): *hiphop*, *hip hop*, *rap*.

training each submodel. Section 4 shows experimental results which compare the performance on different models. We conclude in Section 5. Section 6 gives a broader literature review.

2 Dataset

The Million Song Dataset contains 1,000,000 songs from 44,745 unique artists, with user-supplied tags for artists from the MusicBrainz website, comprising 2,321 unique social tags. Their frequencies follow a power law-like distribution. We looked at the full tag list, sorted by frequency, and picked a set of 10 tags that seemed to represent musical genres. Musical genre is notoriously subjective concept, but we tried to follow a genre set used in previous work (Tzanetakis and Cook, 2002), and further suggestions from the MSD author,[3] while trying to derive a qualitatively reasonable representation of the top 100 most frequent tags, with a somewhat balanced distribution of songs per genre. For two genres whose tags had a lower amount of data, we added songs from a few alternate tag names, which include minor spelling variants. The final genres are shown in Table 1.

In retrospect, we are not totally satisfied with this set of genres, since some of the distinctions may be difficult to qualitatively characterize (e.g. *rock and indie* vs. *classic pop*

[3] http://labrosa.ee.columbia.edu/millionsong/blog/11-2-28-deriving-genre-dataset

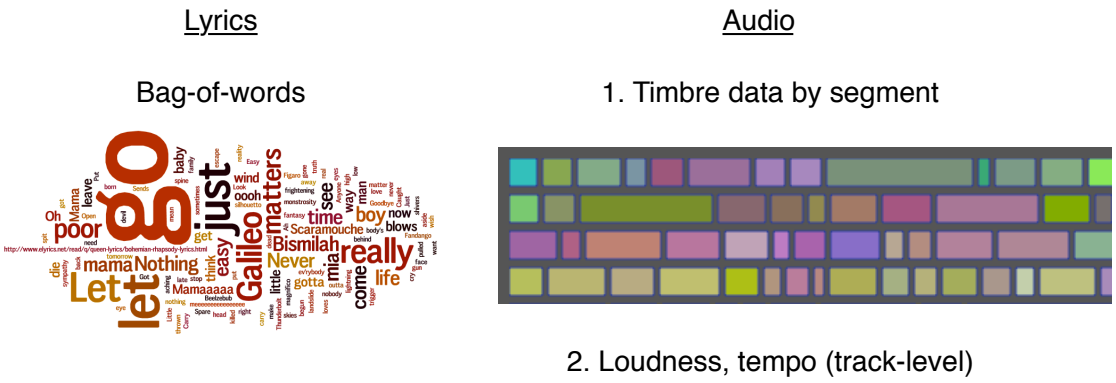


Figure 1: For one track, the lyrics and audio data that we use.

and rock), and there are always more genres that could have been included (e.g. *country*, *punk*, *Latin*, etc.). However, any genre taxonomy is necessarily subjective, incomplete, and forced to balance competing interests—for example, we used both *classic pop and rock* and *rock and indie* because they are extremely frequent tags in the data, and therefore seemed important to model to support information retrieval goals. We believe these genres should be reasonable enough as a testbed for analysis.

There are 156,289 songs having one of the above tags (15% of all MSD). We split the dataset by artist into training, tuning, and test sets. We include all songs of an artist into the artist's assigned split; we randomly selected enough artists to yield 1000 songs per genre for testing, 200 for parameter tuning, and the rest for training. Therefore the final test set has 10,387 songs, while the training set has 143,589. This is orders of magnitude larger than all previously published work in genre classification that we know of.

For our progress report, we conducted experiments on a much smaller set of 1406 songs (derived from the MSD's "10,000 song subset" data release), but in this paper we only report results on the full dataset described above. (Our earlier dataset had a problem of having an unbalanced class distribution, which made training and evaluation trickier.)

3 Approach

3.1 Blend Model

We use several approaches to leverage different types of information about a song into a final classifier. Given the scale of the data we have, any super-linear algorithm such as K-NN or kernelized support vector machines (e.g. with a radial basis kernel) is computationally prohibitive.[4] For one track (i.e. song), we assemble features from several high-level feature extractors into a final classification probability via regularized multi-class logistic regression. For genre $y \in \{1..K\}$ and track audio and lyrics information $(x_{\text{aud}}, x_{\text{lyr}})$:

$$p(y \mid x; w) \propto \exp\left(w_y^{\mathsf{T}} f_{blend}(x_{\text{aud}}, x_{\text{lyr}})\right)$$

where w_y denotes linear feature weights for class y, and f_{blend} is the feature extraction function, outputting a vector size J. The logistic regression learns the weight matrix $w \in \mathbb{R}^{K \times J}$ to optimize log probabilities of the genre labels in the training data.

The feature function f_{blend} has multiple parts: we develop several broad feature classes to capture different acoustic, musical, and textual aspects of the data.

- (Audio) Hidden Markov Model genre probabilities $f_{\text{hmm}}(x_{\text{aud}})$ (Section 3.4), from timbre (sound texture) features $f_{\text{timbre}}(x_{\text{aud}})$ (Section 3.2)
- (Audio) Loudness and tempo of track $f_{\text{LT}}(x_{\text{aud}})$ (Section 3.2)
- (Text) Lyrics bag-of-words submodel probabilities $f_{\text{BOWModel}}(x_{\text{lyr}})$ and emotional valence $f_{\text{emot}}(x_{\text{lyr}})$ (Section 3.3)
- (Combined) Canonical correlation analysis $f_{\text{cca}}\left(f_{\text{timbre}}(x_{\text{aud}}), f_{\text{BOW}}(x_{\text{lyr}})\right)$: reduce dimension of above features to a shared space (Section 3.5)

We can break down the feature extraction function in terms of these broad families of features. Where the song's audio and lyric information are denoted x_{aud} and x_{lyr}, the final

[4] A naive version of K-NN is quadratic time. The training and runtime of a kernelized SVM is less clear; it depends on the number of support vectors, but in noisy data, most of the dataset ends up in the support vector, causing runtime to be similar to nearest neighbors.

feature vector is concatenated from several subcomponents,

$$f(x_{\text{aud}}, x_{\text{lyr}}) = \begin{pmatrix} f_{\text{LT}}(x_{\text{aud}}),\ f_{\text{BOWModel}}(x_{\text{lyr}}),\ f_{\text{emot}}(x_{\text{lyr}}), \\ f_{\text{hmm}}(x_{\text{aud}}),\ f_{\text{cca}}\left(f_{\text{timbre}}(x_{\text{aud}}), f_{\text{BOW}}(x_{\text{lyr}})\right) \end{pmatrix} \tag{1}$$

We test several variants combining different subsets of the features classes; the final model has $J = 32$. We call this the *blend model* (terminology from Netflix Prize systems, e.g. Bell et al. (2007)), since it combines the decisions of submodels and other features.

In Section 3.2, we describe in detail the raw audio features x_{aud} we use. Section 3.3 and 3.4 describe high-level feature extraction of lyrics and audio respectively. Section 3.5 demonstrates Canonical Correlation Analysis (CCA) for combining audio and lyrics features.

3.2 Audio Features

For all music processing tasks, the initial time-series audio signal is heavily processed into *segments*, which approximately correspond to notes or small coherent units of the song—the space between two onsets. Segments are typically less than one or two seconds in length. Figure 1 shows a fragment of a visual representation of one song's segments; the song (*Bohemian Rhapsody*) is 6 minutes long with 1051 segments.[5]

The MSD does not distribute raw acoustic signals (for copyright reasons), but does distribute a range of extracted audio features, many of which can be used for classification.[6] Some audio features, like average loudness or estimated tempo, exist at the track-level and are straightforward to incorporate as classification features.

We note one interesting segment-level feature that touches on fundamental aspects of music. **Timbre** refers to the musical "texture" or type of sound—the "quality that distinguishes different types of musical instruments or voices." This is represented as 12-dimensional vectors that are the principal components of Mel-frequency cepstral coefficients (MFCCs); they represent the power spectrum of sound, and are derived from

[5]This is from an excellent interactive demo at http://static.echonest.com/BohemianRhapsichord/index.html.

[6]They are derived from EchoNest's analysis software: http://developer.echonest.com/docs/v4/_static/AnalyzeDocumentation_2.2.pdf

Fourier analysis and further processing. MFCCs are very commonly used in speech recognition and music information retrieval, as discussed in Muller (2007).

Every track as a different number of segments depending on its length—therefore the timbre data is a matrix in $\mathbb{R}^{12\times N}$, where N varies extremely widely for every song, ranging from 400 to more than 1600 (average around 900). A major challenge is how to properly combine information across segments for the track level. We discuss our solutions in Section 3.4.

Also track level audio features such as loudness and tempo which captures the high level information of the audio. Tempo is defined as number of beats per minute, or BPM and loudness is a real value number describes the general loudness of the song.

3.3 Lyrics Features

The musiXmatch dataset, associated with MSD, provides bag-of-words representations of the lyrics for MSD tracks, where they have performed stemming and other normalizations.[7] There are no alignments to the audio time series. Unfortunately, the full texts of lyrics are not available for copyright reasons.

We seek to use two basic types of word features. First are **bag-of-words** features, that represents a song's lyrics as a vector where every word is a feature. We control for variability and burstiness in word counts in a very simple manner, by normalizing word counts into an indicator variable for whether a word is present or not:

$$f_{BOW}(x_{lyr}) = [\ 1\{\text{ word } w \text{ occurs at least once in song } x\ \}\ ]_w$$

(This could be called "set-of-words"; in previous research projects we have found it is a stable and easy-to-use alternative to other word normalization schemes.)

We do *not* remove stopwords, since (1) the learning algorithm can learn to give them low weights if necessary, and (2) supposedly low-content "stopwords" have been shown to be very useful statistical cues of sentiment (Pang and Lee, 2008) and psychology (Tausczik and Pennebaker, 2009).

[7] Which are of debatable usefulness, as argued by Manning et al. (2008).

Genre	Total	Num. w/lyrics	%
classical	4,889	246	5.0
metal	9,580	5,323	55.6
hiphop	9,210	3,237	35.1
dance	9,466	1,413	14.9
jazz	7,728	414	5.4
folk	18,630	6,631	35.6
soul	6,456	1,751	27.1
rock/indie	36,222	16,029	44.3
pop	10,150	5,053	49.8
classic pop/rock	43,958	17,124	39.0
Totals	156,289	57,221	36.6

Table 2: Amount of lyric data by genre.

We extracted these features from the musiXmatch dataset. As mentioned earlier, this dataset is convenient because it is linked to track ID's from MSD, though it is incomplete; their website reports that they provide lyrics for 77% of all MSD tracks. But within the genres we selected, only 37% of the tracks have lyrics information (Table 2). In some cases, the songs genuinely do not have lyrics—e.g. the low percentage for *classical* and *jazz*, which are highly instrumental genres of music—but in other cases, the data may simply be missing lyrics for that song. Unfortunately, we do not have information to distinguish these cases. In addition to the word features, we include an indicator feature for whether any lyrics exist at all, so the classifier can learn what to do for tracks without lyrics.

The vocabulary (number of unique words) provided by musiXmatch is 5000 words total, therefore f_{BOW} provides 5000-dimensional vectors. To facilitate faster experimentation, we reduce the dimensionality of these features through a submodel: we first train a multiclass logistic regression on these 5000 word features to predict the 10 genre classes, then use the classifiers' log probabilities as a 10-dimensional feature vector:

$$f_{BOWModel}(x) = [\log p(y = 1|f_{BOW}(x)),\ \ldots\ \log p(y = 10|f_{BOW}(x))]$$

where $p(y|h) \propto \exp \sum_{w=1}^{5000} \beta_{yw}[f_{BOW}(x)]_w$, having learned the submodel's weight matrix $\beta \in \mathbb{R}^{10\times5000}$ on the training data. We use L2 regularization and tune the regularization

parameter on the tuning data. Only $f_{BOW\,Model}$, the output of the submodel, is directly used in the final classifier.

The next type of textual features we will use are **emotional valence** features, counting the frequency of words that denote happiness versus unhappiness. We use the ANEW word lists, which were constructed from psychological experiments (Bradley and Lang, 1999), in which people were presented with a word, then rated their evoked feelings on three dimensions: happiness vs. unhappiness ("valence"), excited vs. calm ("arousal"), and feeling-of-control ("dominance"). The lexicon contains averaged judgments for 1034 words, where highly positive words include "triumphant," "paradise," and "love," while highly negative words include "funeral," "rape," and "suicide"; it was used for interesting exploratory analysis of song lyrics in previous work (see Section 6.3).

We calculate song-level statistics from the bag-of-words data, by taking the words in a song that are present in ANEW, and averaging their emotional scores (for each of the three dimensions). We discard the scores if the song has fewer than 5 words present in ANEW. We use these averages, and also the standard deviations of each emotion dimension, as the features, plus an indicator variable if there were too few words. (We are somewhat skeptical whether this procedure necessarily captures the emotional content of a song, but it is a simple technique that is widely used in empirical research, therefore worth testing more rigorously for this task.)

These statistics vary a reasonable amount by genre; Figure 2 shows the distributions of valences. *Metal* is the least happy, while *soul* is the most happy. This roughly corresponds to the exploratory findings in Dodds and Danforth (2009).

3.4 Genre-HMM: Time series acoustic structure model

We use a Hidden Markov Model to model sequential audio timbre data. We assume each genre corresponds to an HMM model, parameterized by the transition matrix T and observation matrix O. We use labeled training data to train one HMM for each genre. At test time, when given a new sequence, we evaluate its likelihoods under each genre-specific HMM, and could classify it by the most likely genre. This approach is analogous to Naive

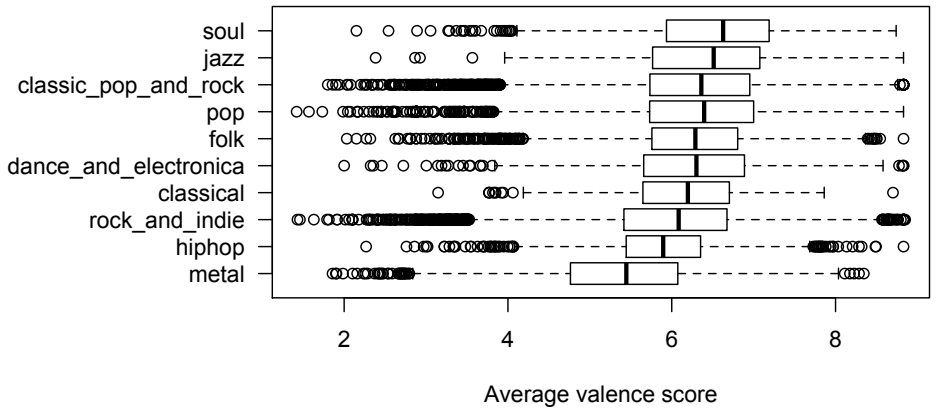


Figure 2: Distribution of song valence scores by genre. Box boundaries and middle line at the 25%, 50%, and 75% percentiles.

Bayes, except with a structured HMM generative distribution. Although this alone is a reasonable classifier, we use the genre-specific likelihoods as intermediate features into the blend model, to combine its evidence with lyrics and other information.

The feature input is the structured time-series timbre data (timbre vectors as mentioned in Section 3.2). For one track, this consists of a matrix of dimensions $(12 \times N)$, where N is the number of the segments in the track, and each column represents a $[0, 1]^{12}$ timbre feature at one time step.

Feature processing: Since HMM's work with discrete observations, we need to process the $12 \times N$ real-valued feature matrix into a one dimensional discrete vector. First, we average the time series over each beat, producing a coarser grained sequence of $12 \times M$, where M is the number of beats in the track. Next, for each beat, we choose the dimension with the largest value as the discrete feature of that time step. For example, if the original vector at $beat_t$ was $[0.4, 0.5, 1, \ldots, 0.1, 0.02]$, the new feature at t becomes 3. We also add a stop state 0.

This process can be interpreted as selecting the dominating power spectrum (1 to 12) for each beat. The dimension of the new feature space is 13. We note that the step of timbre vector dimension reduction for HMM is arguable, and a more appropriate way could be applying K-means. However, our method is simple and faster.

Training: For each genre i, we select its tracks from the training set, and train the

HMM_i using the Baum-Welch EM algorithm (Baum et al., 1970) with 6 hidden states and a uniform prior.

Tuning: The number of hidden states was chosen by maximizing predictive accuracy on the held-out tuning set. To evaluate accuracy, for one sequence of a tuning set track, we run the forward algorithm to filter it for each 10 HMM models, and predict the genre that has the highest likelihood as the predicted label.

Feature Extraction: After we trained the model for each genre, we run forward algorithm to filter the sequence with each model and get a 10 dimensional vector containing the likelihood of this sequence under each genre. This becomes the input to the final classifier. The feature function is:

$$h_{hmm}(timbre(x)) = \frac{1}{Z}[p(x|\text{HMM}_1), \ldots, p(x|\text{HMM}_K)] \tag{2}$$

$$f_{hmm}(timbre(x)) = \log\{h_{hmm}(timbre(x))\} \tag{3}$$

where $p(x|\text{HMM}_i)$ is the likelihood of track x given the HMM trained on genre i, and Z is a normalizing constant so the h_{hmm} vector sums to 1: $Z = \sum_{k=1}^{K} p(x|\text{HMM}_k)$. These normalized likelihoods can be viewed as the genre posterior probabilities a uniform prior: $p(\text{HMM}_i|x) \propto p(x|\text{HMM}_i)p(\text{HMM}_i)$. We use their log posterior probabilities (f_{hmm}) as the features for the blend model.

Complexity: Both Baum-Welch and the Forward algorithm run in $O(S^2N)$ where S is the number of states and N is the length of the sequence. This is linear in the number of tracks in the training data, and runtime classification of the test set is also linear.

3.4.1 Spectral method for learning HMM

One drawback of the Baum-Welch algorithm for learning an HMM is that, being a local-search algorithm, it may get stuck at a poor local minimum. In addition to Baum-Welch, we experiment with an alternative spectral method for learning HMM's, as proposed in (Hsu et al., 2008). Let $\mathcal{O}$ be the space of the observation: $1, 2, \ldots, 13$. Algorithm 1 shows

the spectral HMM learning algorithm.

Algorithm 1: learnHMM

Input: $(X_1, \cdots, X_n)$, m - number of states, N - sample size

Output: HMM Model parameters: $\hat{b}_1$, $\hat{B}_x$, and $\hat{b}_\infty, \forall x \in \mathcal{O}$

- Sample N observation triplets (x_1, x_2, x_3) independently from $(X_1, \ldots, X_n)$, and estimate: $\hat{P}(x_1)$, $\hat{P}(x_2, x_1)$ and $\hat{P}(x_3, x_1|x_2)$.
- Let K be the dimension of observation space, and let $\hat{P}_1$ be the $K \times 1$ vector of $\hat{P}(x_1)$, $\hat{P}_{2,1}$ be the $K \times K$ vector and $\hat{P}_{3,x,1}$ be the $K \times K \times K$ tensor of $\hat{P}(x_3, x_1|x_2)$.
- Compute the SVD of $\hat{P}(x_1, x_2)$, and let $\hat{U}$ be the left singular vectors of the m largest singular values.
  - $\hat{b}_1 = \hat{U}^T \hat{P}_1$
  - $\hat{b}_\infty = (\hat{P}_{2,1}^T \hat{U})^+ \hat{P}_1$, where $+$ denotes the Moore-Penrose pseudoinverse
  - $\hat{B}_x = \hat{U}^T \hat{P}_{3,x,1} (\hat{U}^T \hat{P}_{2,1})^+ \forall x \in \mathcal{O}$

Output: $\hat{b}_1, \hat{B}_x, \hat{b}_\infty$;

After we trained the HMM model, the likelihood of a given sequence is computed using Equation 14. The spectral method has several advantages over the Baum-Welch algorithm. First, it achieves the global optimum, and has nice statistical guarantees. Specifically, Theorem 6 in (Hsu et al., 2008) states that:
For any $0 < \epsilon, \eta < 1$, *and* $t > 1$, *if* $N \geq O(\frac{t^2}{\epsilon^2} \log \frac{1}{\eta})$, *then with probability at least* $1 - \eta$ *that the model returned by Algorithm 1 satisfies:*

$$\sum_{x_1, \ldots, x_t} |Pr[x_1, \ldots, x_t] - \hat{Pr}[x_1, \ldots, x_t]| \leq \epsilon$$

Second, the main computation is the Singular Value Decomposition of the K by K matrix $P_{2,1}$. This is faster than Baum-Welch, especially when the dimension K of the observation space is low.

Note that the HMM model returned by Algorithm 1 is parameterized differently than

the standard HMM—it is based on the Observable Operator Model formulation of HMM's; see (Hsu et al., 2008; Jaeger, 2000b). It does not recover the original HMM parameters, but still gives correct predictions. We provide a detailed review of spectral methods in Section 6.1.

In Section 4 we compare the classification accuracy derived by both, which surprisingly shows that the spectral method performs worse than Baum-Welch for our task. We discuss the reasons for this result.

3.5 CCA: Combining Audio and Lyrics Features by Canonical correlation Analysis

3.5.1 CCA

Our genre classifier combines audio and textual lyric features. Can we usefully define a lower dimensional, shared feature representation? Canonical Correlation Analysis (§6.4) is a technique that seeks to address this problem, by revealing shared linear correlations between two different datasets.

Given two datasets $X \in \mathbb{R}^{n \times d_x}$ and $Y \in \mathbb{R}^{n \times d_y}$, the objective of CCA is to find weights $w_x \in \mathbb{R}^{d_x}$ and $w_y \in \mathbb{R}^{d_y}$ that maximize the correlation between the projections of X and Y, X_{w_x} and Y_{w_y}. This problem can be solved via the following generalized eigenvalue problem:

$$\begin{pmatrix} 0 & X'Y \\ Y'X & 0 \end{pmatrix} \begin{pmatrix} w_x \\ w_y \end{pmatrix} = \lambda \begin{pmatrix} X'X & 0 \\ 0 & Y'Y \end{pmatrix} \begin{pmatrix} w_x \\ w_y \end{pmatrix} \tag{4}$$

In our setting, the data X and Y refer to audio and lyrics features. Initially, the raw audio features exist at the segment level, but we need track-level information both for track classification, as well as to project them into the same space as the lyrics features (which only exist at the track level). Different approaches have been proposed for segment aggregation (McVicar et al., 2011); for example, take the mean and standard deviation of the timbre features for the whole track. This is unsatisfying since music has a very strong temporal property—more so than images or text—therefore averaging may degrade the

dynamics of music texture. But averaging is the simplest approach.

3.5.2 CCA and Logistic Regression experiments

We conducted smaller-scale experiments with CCA on a subset of the data. Since it did not perform well, we did not use CCA for the final model. We report these results in this section—to be clear, this is separate from the main experiments described in all other sections.

We implement three classifiers, with lyrics, timbre, and CCA features:

1. f_{BOW}: Track-level bag-of-words lyrics features (5000-dimensional; see §3.3).
2. $f_{TimbreAvg}$: Track-level audio features from averaging the timbre features across all segments for the whole track (12-dimensional; see §3.2).
3. f_{CCA}: Combine the above features through CCA (§3.5.1). The track-level audio features are 12-dimensional and the track-level bag-of-words lyrics features are 5000-dimensional. CCA will choose the dimension of the joint space as the minimum of the dimensions of all the input feature spaces, which is 12 here.

We construct classifiers solely from BOW and TIMBREAVG as baselines. Theoretically, CCA's combined features could result in improvements since it can learn shared joint information between the text and audio modes. However, as we argued in the previous section, by aggregating the segment-level features, we may lose useful temporal dynamics. Furthermore, as mentioned in McVicar et al. (2011), there is only a minor correlation between the mood expressed in a song's audio, compared to the mood expressed in its lyrics. This suggests that CCA cannot capture mood-based aspects of a song.

The data we use here is a subset from the full dataset described in Section 2. Since about two-thirds of the tracks have no lyrics, and CCA requires them to be present in order to work, we only keep the tracks with lyrics, which leads an imbalanced dataset (Table 2); for example, most of the *classical* and *jazz* tracks have no lyrics. The goal is to test whether CCA can help at all. We select a subset with 1698 tracks for training and 671 tracks for testing. Therefore, the results in this section cannot be directly compared to the main results (Section 4).

	BOW (lyrics)	TIMBREAVG (audio)	CCA (combined)
Acc. %	24.3	42.0	33.0

Table 3: Experimental results on the small dataset, comparing CCA to a limited set of models.

We use multiclass logistic regression (Equation 3.1) with L2 regularization. Though some successful approaches to genre classification have built track-level models like Gaussian Mixture Models (e.g. Tzanetakis and Cook (2002)), an efficient classifier like logistic regression is most suitable given the scale of our full dataset if we had decided to include CCA in our final model.

Our experimental results on this reduced dataset are shown in Table 3. These results follow the negative results from our progress report: CCA doesn't improve beyond the TIMBREAVG baseline. There could be at least two problems: first, averaging the timbre features over the whole track degrades the dynamics of music texture, and second, the reduced dimension representation from CCA obscures the independent signals of individual words. This potentially could be marginally useful, incorporating CCA alongside base features; but these initial results were negative enough that we did not include CCA in our final experiments.

4 Results

We train the blend model on the full dataset, using a number of different feature combinations. We report results for the two different timbre HMM's, bag-of-words lyrics, lyric sentiment, and loudness and tempo features.

Accuracy results on the test set are presented in Table 4, with further breakdowns in Table 5 and Figure 3. We can answer a number of interesting research questions from these results.

Do timbre features work? Yes. Each genre comprises about 10% of the data, while the most basic audio model, using only the timbre Baum-Welch HMM, achieves 31.4% accuracy.

Model	Acc. %	Model	Acc. %
(†)BW	31.4	Sp	28.3
(††)Lyrics	22.1	Sp+BW	32.6
(†)BW+Lyrics	35.2	Sp+Lyrics	33.3
LT+BW+Lyrics	37.5	LT+Sp+Lyrics	36.0
(†)LT+Sp+BW+Lyrics	**38.6**	LT+Sp+BW	34.1
Emot+LT+Sp+BW+Lyrics	38.5	*Most common class*	*10.5*

BW = Baum-Welch HMM, Sp = Spectral HMM, LT = loudness and tempo,
Lyrics = all words from lyrics, Emot = lyric emotion scores

Table 4: Final results from different model combinations. Best result is in bold. Note that 95% confidence intervals are approximately $\pm 1\%$ $(= 1/\sqrt{n})$. (†): These models are shown in Table 5 and Figure 3. (‡): The lyrics-only model has 40.0% accuracy on songs that have lyric data.

Genre	Lyrics	BW	BW+Lyrics	Final Model
classical	*0.0*	75.0	77.7	78.1
metal	*71.3*	65.8	57.8	63.6
hiphop	*67.7*	40.4	45.1	52.2
dance	*6.7*	34.2	45.1	45.7
jazz	*0.0*	18.2	30.1	36.9
folk	*35.9*	41.3	35.5	32.5
soul	*15.6*	19.7	19.1	24.6
rock/indie	*41.2*	6.9	17.5	21.7
pop	*37.4*	7.6	15.5	16.2
classic rock/pop	*21.0*	5.9	10.7	16.1
Totals (Table 4)	*40.0* (22.1)	31.4	35.2	38.6

Table 5: Accuracy (%) results per class. The "Lyrics" column only shows accuracy rates for songs that have lyrics data. These are the same models in Figure 3. The "Final Model" is LT+Sp+BW+Lyrics. Note there are about 1000 songs per class, therefore 95% confidence intervals are approximately $\pm 3\%$.

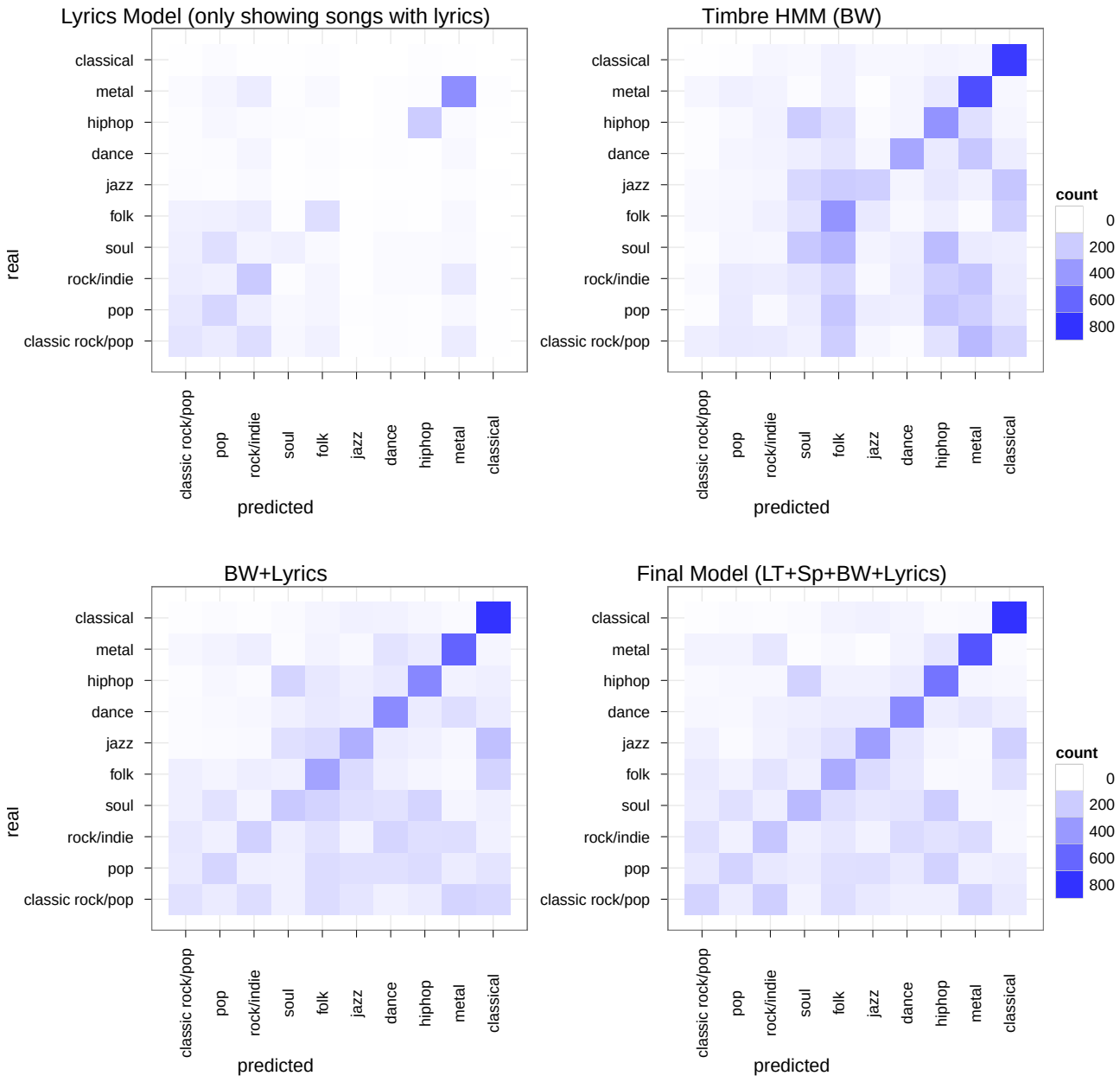


Figure 3: Confusion matrices of models' predictions on the test set. The legend is in terms of counts. There are approximately 1000 songs per class in the test set, so a count of 800 corresponds to 80%. These are the same models shown in Table 5.

When do timbre features work? If we look at the class breakdown, we see the BW-HMM performs best on *classical* music: 75% of the classical tracks are correctly tagged as classical. This makes sense, since classical music involves substantially different instruments than the other genres in our dataset.

Interestingly, the audio features perform quite poorly on the three genres of *pop*, *rock and indie*, and *classic pop and rock*. Looking at the confusion matrix (top-right of Figure 3), we see that those three genres are often confused for *folk* and *metal* (and sometimes *hip-hop*). The HMM is better at predicting those classes, so the logistic regression finds that it optimizes accuracy to be biased towards predicting them.

Do these results tell us anything about the nature of musical genres? Possibly. One hypothesis is that that the pop and rock genres are typified less by musical features (which can be detected in acoustic data) but rather, more by cultural style or historical periods, and therefore should be difficult to detect from the timbre data. These results support this hypothesis, and suggest further investigation.

Do bag-of-words lyric features work? Yes. If we look only at tracks that have lyrics data, the classifier achieves 40% accuracy—higher than the audio models on the full dataset. However, since only one-third of the tracks have lyric data, this is incomplete; forcing it to make predictions on the entire test set (so, all songs without lyrics are chosen to be the most common one) only gets 22% accuracy.

When do bag-of-words lyric features work? They are best for *metal* and *hip-hop*. The features are quite poor for genres with very little lyrics data—the classifier never predicts *classical* or *jazz*, getting 0 accuracy for them. As shown in Section 3.3, these genres have nearly no lyric data, so this is unsurprising.

Does lyric data give different information than audio? Yes, to a certain extent these data sources are orthogonal. While the two best genres for the lyrics model are also served well by the timbre HMM, the lyrics model is good for several genres that the timbre HMM is very bad at—including those three problematic rock and pop genres at the bottom of Table 5. This can be visually seen in the confusion matrices (top row of Figure 3)—there is a little bit of moderate accuracy in areas where the timbre HMM does poorly. The confusion matrix makes it obvious that the lyrics model is very incomplete, making zero

predictions (vertical white bars) for *jazz* and *classical*. The hope is that combining the models can improve overall accuracy, by filling in where each other is weak.

Does combining audio and lyric features help? Yes. Combining the bag-of-words submodel with the timbre HMM achieves 35.2% accuracy, higher than either of the individual models. As can be seen in the confusion matrix (bottom-left of Figure 3), the combined model is able to spread the submodels' confidences into more genres. Indeed, the rock and pop genres all see their accuracy rates increase under the combined model.

While some genres have large improvements, a few see a decrease. This is because the blend logistic regression is tuning the submodels' weights to optimize overall accuracy,[8] so it finds it useful to borrow strength from some classes to help other ones.

Note also, the lyrics information can help even when there are no lyrics, since the lyrics model uses the indicator variable of whether there are any lyrics at all. For example, if there are no lyrics, then *jazz*, *classical*, and *dance* are more likely; we believe this is why these classes see an improvement.

Does adding loudness and tempo help? Yes. From Table 4, we could see that adding loudness and tempo can always bring about 2% to 3% increase in accuracy: LT+BW+Lyrics >BW+Lyrics, LT+Sp+Lyrics>Sp+Lyrics, LT+Sp+BW>Sp+BW. This is reasonable since different genres do vary in tempo and loudness—for example, the tempo of *hip-hop* music is often faster than that of *jazz / folk* music, while *metal* songs tends to be louder than most of the other genres.

Which HMM algorithm is better: Baum-Welch or Spectral? Baum-Welch, but spectral is still useful. Using a spectral HMM alone does worse than the Baum-Welch alone (28.3% vs. 31.4%). Furthermore, we conducted several experiments swapping the spectral for Baum-Welch, and in all cases the BW version wins by at least 1%. (From Table 4: Sp<BW, SP+Lyrics<BW+Lyrics, LT+Sp+Lyrics<LT+BW+Lyrics).

Although the spectral methods discussed in Sections 3.4.1 and 6.1 work well in predicting the future observations, they only recover the transition and observation parameters within a similarity transformation. Furthermore since the spectral methods work in a wider class of models rather than HMM's, our restrictive use of it for HMM's leads to

[8]Technically, it optimizes log-likelihood, a quantity similar to accuracy.

poorly calibrated probability estimates, which results in performance inferior to state-of-the-art Balm-Welch. In other words, in terms of getting the correct likelihood, the spectral methods only work better if the HMM's assumptions are correct. But in other application domains, such as robotics, where observation prediction is the task, spectral HMM's relaxed assumptions can achieve great gains. Therefore we believe our problem may not be a natural fit for the spectral HMM. This served as a good opportunity and valuable lesson in investigating how to use spectral methods for sequence clustering and classification.

However, the spectral HMM is still useful to blend alongside the Baum-Welch HMM: using both models is always better than using one of them alone: Sp+BW>BW, Sp+BW>Sp, LT+Sp+BW+Lyrics>LT+BW+Lyrics, LT+Sp+BW+Lyrics>LT+Sp+Lyrics.

This is an interesting point about **system combinations**. We would expect the spectral HMM and Baum-Welch HMM to make broadly similar predictions—and if we inspect their individual confusion matrices (not shown), they are indeed similar—but they are still different enough that they give complementary views of the data, that can be usefully combined into an even better model. Indeed, the best systems for competitive data mining tasks often use blends of many different submodels—the winning Netflix Prize models used thousands of individual submodels, and found that more models kept helping (Bell et al., 2007, 2008).

Are lyric sentiment features useful? No. They provide basically zero value when added to a model, changing the accuracy rate less than the threshold for statistical significance. We also conducted an experiment of using only the sentiment features by themselves, and they provicded no signal in that setting (approx. 10% accuracy). We even tried a non-linear model (logistic boosted decision trees: Ridgeway (2007); Friedman et al. (2000)), which can learn to correspond different regions of the feature space to different categories. This still did not work. This is a cautionary note—just because these features are qualitatively interesting doesn't mean they are necessarily useful or right for a task.

5 Conclusion

In this project, we propose a cross-modal retrieval framework of *model blending*, which combines features from audio and lyrics for the task of *music genre classification*. Our results show that the blending features performs better than any individual one. Timbre HMM's, lyrics bag-of-words, and loudness/tempo features were all useful; lyric sentiment and CCA were not. This approach is excellent for careful testing and analysis what different submodels contribute.

Our work is somewhat different than previous work on genre classification, because (1) we use a different, novel dataset, and (2) its very large size necessitates the use of robust and scalable algorithms. Since our approach scales linearly with the number of the training examples, it is suitable for large-scale music IR research.

6 Literature Review

6.1 Spectral methods for modeling dynamical systems (Haijie)

Recently, the rising of spectral methods in modeling dynamical systems shows a possible way to tackle this problem efficiently and at large scale (Hsu et al., 2008; Siddiqi et al., 2010; Boots and Gordon, 2011). Here we give a short review on the development of the spectral methods.

Hsu et al. (2008) proposed a spectral method for learning the Hidden Markov Model. Unlike the Baum-Welch/EM (Baum et al., 1970) algorithm, which is slow and subject to poor local optima, the spectral algorithm is statistically consistent. Furthermore, it avoids local optima and also only requires the use of well-understood algorithms for singular value decomposition and matrix multiplications (described below).

Suppose the HMM has m hidden states, and n observation classes, for hidden states $y_t \in \{1..m\}$ and observations $x_t \in \{1..n\}$ for a sequence of observations. Let $\pi \in \mathbb{R}^m$ be the initial state distribution, $T \in \mathbb{R}^{m \times m}$ be the transition matrix, and $O \in \mathbb{R}^{n \times m}$ the emission

distribution. These parameterize the HMM as follows:

$$\pi_i = P(y_1 = i) \tag{5}$$

$$T_{ij} = P(y_t = i \mid y_{t-1} = j) \tag{6}$$

$$O_{aj} = P(x_t = a \mid y_t = j) \tag{7}$$

The joint probability of a sequence of observations $x_1, \ldots x_t$ is:

$$Pr[x_1, \ldots, x_t] = \sum_{y_1} P(y_1)P(x_1 \mid y_1) \sum_{y_2} P(y_2 \mid y_1) \cdots \sum_{y_{t-1}} \cdots \sum_{y_t} P(y_t \mid y_{t-1})P(x_t \mid y_t) \tag{8}$$

For each observation x, we denote $O_x = diag(O_{x,1}, \ldots, O_{x,m})$, and rewrite the probability as:

$$Pr[x_1, \ldots, x_t] = \mathbf{1}TO_{x_1}TO_{x_2} \ldots TO_{x_n}\pi = \mathbf{1}A_{x_1}A_{x_2} \ldots A_{x_t}\pi \tag{9}$$

where $A_x = TO_x \in \mathbb{R}^{m \times m}$ is called the observation operator with respect to observation x. It denotes the possible local likelihood contributions (from transitions and emissions) of an observation, for different combinations of current state i and previous state j: $(A_x)_{ij} = P(y = i \mid y_{-1} = j)P(x \mid y = i)$. Successive matrix multiplications of A_{x_1}, A_{x_2}, etc. correspond to the usual dynamic programming forward algorithm for computing the joint HMM probability of the sequence.

The central idea of the spectral learning algorithm is to represent Equation 9 as:

$$Pr[x_1, \ldots, x_t] = \mathbf{1}S^{-1}(SA_{x_1}S^{-1})(SA_{x_2}S^{-1}) \ldots (SA_{x_t}S^{-1})(S\pi) \tag{10}$$

and to find the invertible matrix $S \in \mathbb{R}^{m \times m}$, such that SA_xS^{-1} is easy to estimate directly from the data. Hsu et al. show that by choosing $S = U^TO$, where U is the 'thin' SVD of the covariance matrix of the observations, the parameters defined in Equation 13 can be

easily estimated from the data.

$$\mathbf{b_1} = (U^T O)\pi \tag{11}$$

$$\mathbf{B_x} = (U^T O)A_x(U^T O)^{-1} \tag{12}$$

$$\mathbf{b}_\infty = \mathbf{1}(U^T O)^{-1} \tag{13}$$

Therefore the joint probability of any sequence of observations can be computed by:

$$Pr[x_1, \ldots, x_t] = \mathbf{b}_\infty B_{x_1} B_{x_2} \ldots B_{x_t} \mathbf{b_1} \tag{14}$$

Similarly, we can compute the conditional probability of a sequence given some history using the observable operator (see Jaeger (2000a)) in Equation 13.

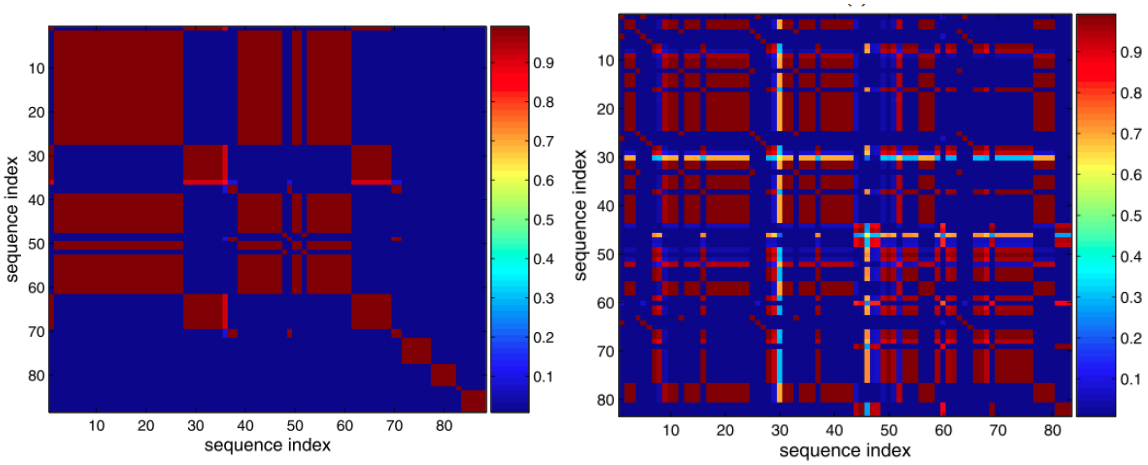
However, in Hsu et al. (2008), the algorithm requires the transition matrix T to be full rank, which is restrictive. Siddiqi et al. (2010) relaxed this full rank constraint, and suggested that we could keep a k-dimension ($k < m$) representation of the state by only taking the k largest eigenvectors as the projection matrix $\hat{U}$. This effectively reduced the dimension of the parameters space.

In fact, these spectral algorithms work for a more expressive model, or richer representation, called the Predictive State Representation by Singh and James (2004), Rosencrantz et al. (2004), or the Transformed Predictive Space Representation (corresponding to the reduced rank HMM). Boots and Gordon (2011) provides an efficient online learning extension of the spectral algorithm which is scalable. This online extension is not necessarily faster than the batch algorithm; however, its streaming nature allows one to deal with a huge feature matrix in a reasonable amount of memory.

6.2 Audio sequence modeling of music structure (Brendan)

Can sequence models recover the high-level structure of music? Ren et al. (2010) seek to accomplish this with a non-parametric Bayesian HMM model. Following previous work, they use vector quantization (VQ) to turn a song's time-series of an acoustic feature set (in their case, MFCCs) into a codebook of 16 discrete states for every 50ms period. Since the

(a) Left: Waveform for "A Day in the Life" by the Beatles. Right: Waveform for Beethoven's Sonata No. 17.



(b) Similarity matrix (across time) of posterior DP HMM subsequence type variables, for each song.

Figure 4: Two songs and their higher-level structures, as inferred by the model of Ren et al. (2010).

signal is completely discretized, they can model it with a Hidden Markov Model. They claim that previous work focuses on small, several-second subsequences, and they seek to extend this by introducing dependencies between subsequences to model the high-level segments and movements of the music. Therefore, they introduce a hierarchical model, where the lowest level models "subsequences" of approximately 5 to 6 seconds, and the upper level models transitions over types of subsequences. This is done in a Bayesian setting with Dirichlet Process prior distributions—an "infinite" mixture model—so that the number of states is automatically chosen, and that the dependence among subsequences can be modeled as an "innovation" random variable that interpolates the current DP to form the base measure defining the prior for the next subsequence's states.

We reproduce their models' outputs in Figure 4. The first song is an early Beatles song that has very simple instrumentation and structure. The authors released a movie that plays the song and marks the position in the similarity matrix.[9] It was apparent to us that the model basically captures the distinction between soft and loud parts of the music,

[9] http://pubs.amstat.org/doi/suppl/10.1198/jasa.2009.ap08497

as can be seen by the predominance of two different colors in the posterior similarity matrix. This is reasonable since the song is quite simple (lots of previous work in music structure has in fact examined similar songs by the same artists). They also present a more complex example with a Beethoven piano sonata, where the model finds a much richer set of states and high-level structure. They qualitatively compare the model's segmentation of the music to one done by a music theory expert and find some similarities.

This work was interesting in that it shows it's possible to capture high-level structure of a song with Markovian sequence modeling. We are skeptical that their technical approach is the only way to do it, and believe it would not work well for the Million Song Dataset—for example, non-parametric Bayesian methods are probably overkill for this problem. Also, they use an MCMC method for fitting the model, which is known to perform quite poorly on large datasets.

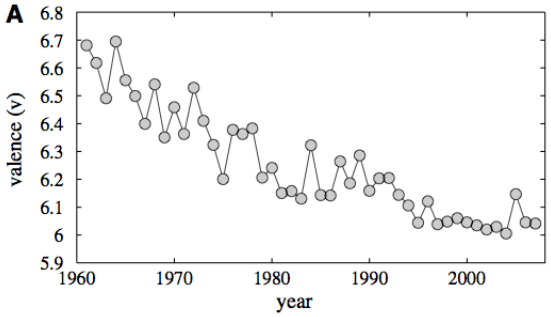
6.3 Emotion analysis of song lyrics (Brendan)

Dodds and Danforth (2009) is an exploratory analysis of mood in song lyrics and other textual corpora. They use the ANEW emotion valence lexicon, described in Section 3.3.

They process a large corpus of song lyrics, computing average happiness scores for songs. While we harbor doubts whether these statistics are truly meaningful,[10] Dodds and Danforth do produce several interesting analyses, several of which we reproduce in Figure 5, which were computed over 232,574 songs and 20,025 artists. In 5(a), they observe a consistent downward trend in valence over time. Second, when looking at top- and bottom-valenced individual artists (5(b)), the results seem to agree with our intuition, at least, of how upbeat or downbeat some of those various artists are. Third, they break down genres by their emotional valence score, and note substantial differences; for example, pop and gospel/soul are most happy, while punk and metal/industrial are less so. Also, there is a suggestion that valences may be shifting over time.

These results are only suggestive, but they do indicate there is interesting information in the emotional content of song lyrics. This motivated our inclusion of lyric sentiment

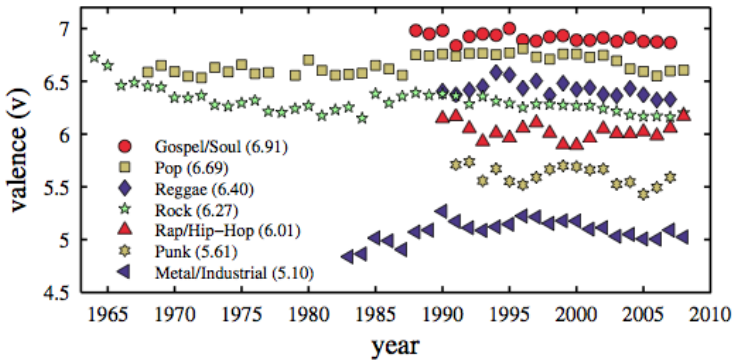
[10] http://brenocon.com/blog/2011/10/be-careful-with-dictionary-based-text-analysis/



(a) Average song valence over time

Rank	Top artists	Valence	Bottom artists	Valence
1	All 4 One	7.15	Slayer	4.80
2	Luther Vandross	7.12	Misfits	4.88
3	S Club 7	7.05	Staind	4.93
4	K Ci & JoJo	7.04	Slipknot	4.98
5	Perry Como	7.04	Darkthrone	4.98
6	Diana Ross & the Supremes	7.03	Death	5.02
7	Buddy Holly	7.02	Black Label Society	5.05
8	Faith Evans	7.01	Pig	5.08
9	The Beach Boys	7.01	Voivod	5.14
10	Jon B	6.98	Fear Factory	5.15

(b) Valence per artist, for selected artists



(c) Valence per genre, over time

Figure 5: Text-based valence analysis of song lyrics, from Dodds and Danforth (2009).

features in our model.

6.4 Cross-Modal Retrieval based on Correlation and Semantic Matching (Dawen)

Rasiwasia et al. (2010) proposed a novel cross-modal framework based on correlation and semantic matching, which was originally used for image/text cross-modal retrieval; e.g., retrieve a Wikipedia page having both descriptive texts and associated images. The underlying approach may be more general for different media. In their setting, two possible cross-modal approaches are proposed: correlation matching and semantic matching, they also provide the results by combining these two.

Correlation matching is based on CCA (Section 3.5), which performs dimension reduction of heterogeneous representations of the same data. Given a pair of image and text, CCA's learned bases will project the feature vectors to a joint space in which different feature spaces are maximally correlated. Therefore, once these basis are obtained, all the retrieval tasks will reduce to the problem in one data space by the projection.

Their second method, semantic matching, uses a predefined set of semantic concepts or topics; for example, broad document classes, such as "History" or "Biology". Then they learn the mapping from individual spaces with logistic regression. This is substantially different than the CCA approach, because they are not trying to learn the latent semantic representation, instead taking it as prior knowledge.

References

Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):pp. 164–171, 1970. ISSN 00034851. URL http://www.jstor.org/stable/2239727.

Robert M. Bell, Yehuda Koren, and Chris Volinsky. The BellKor solution to the

Netflix Prize, 2007. http://www2.research.att.com/~volinsky/netflix/ProgressPrize2007BellKorSolution.pdf.

Robert M. Bell, Yehuda. Koren, and Chris Volinsky. The Bellkor 2008 solution to the Netflix Prize, 2008. http://www2.research.att.com/~volinsky/netflix/Bellkor2008.pdf.

Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

Byron Boots and Geoffrey J. Gordon. An online spectral learning algorithm for partially observable nonlinear dynamical systems. In *AAAI*, 2011.

M. M Bradley and P. J Lang. Affective norms for english words (ANEW): instruction manual and affective ratings. *University of Florida: The Center for Research in Psychophysiology*, 1999.

P. S. Dodds and C. M Danforth. Measuring the happiness of Large-Scale written expression: Songs, blogs, and presidents. *Journal of Happiness Studies*, page 116, 2009.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors). *The annals of statistics*, 28(2): 337407, 2000. ISSN 0090-5364.

Daniel Hsu, Sham M. Kakade, and Tong Zhang. A spectral algorithm for learning hidden markov models. *CoRR*, abs/0811.4413, 2008.

Herbert Jaeger. Observable operator models for discrete stochastic time series. *Neural Computation*, 12(6):1371–1398, 2000a.

Herbert Jaeger. Observable operator models for discrete stochastic time series. *Neural Computation*, 2000b.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, 1st edition, July 2008. ISBN 0521865719.

M. McVicar, T. Freeman, and T. D. Bie. Mining the correlation between lyrical and audio features and the emergence of mood. In *Proceedings of the 12th International Conference on Music Information Retrieval*, 2011.

M. Muller. Information retrieval for music and motion. In *Springer*, 2007.

Bo Pang and Lillian Lee. *Opinion Mining and Sentiment Analysis*. Now Publishers Inc, July 2008. ISBN 1601981503.

N. Rasiwasia, J. C. Pereira, E. Coviello, G. Doyle, G. Lanckriet, R. Levy, and N. Vasconcelos. A new approach to cross-modal multimedia retrieval. In *Proceedings of the international Conference on Multimedia*, 2010.

L. Ren, D. Dunson, S. Lindroth, and L. Carin. Dynamic nonparametric bayesian models for analysis of music. *Journal of the American Statistical Association*, 105(490):458472, 2010.

Greg Ridgeway. Generalized boosted models: A guide to the gbm package, 2007. http://cran.r-project.org/web/packages/gbm/vignettes/gbm.pdf.

Matthew Rosencrantz, Geoff Gordon, and Sebastian Thrun. Learning low dimensional predictive representations. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, pages 88–, New York, NY, USA, 2004. ACM. ISBN 1-58113-838-5. doi: http://doi.acm.org/10.1145/1015330.1015441. URL http://doi.acm.org/10.1145/1015330.1015441.

Sajid Siddiqi, Byron Boots, and Geoffrey J. Gordon. Reduced-rank hidden Markov models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS-2010)*, 2010.

Satinder Singh and Michael R. James. Predictive state representations: A new theory for modeling dynamical systems. In *In Uncertainty in Artificial Intelligence: Proceedings of the Twentieth Conference (UAI)*, pages 512–519. AUAI Press, 2004.

Yla R. Tausczik and James W. Pennebaker. The psychological meaning of words: LIWC and computerized text analysis methods. *Journal of Language and Social Psychology*, 2009. URL http://jls.sagepub.com/cgi/rapidpdf/0261927X09351676v1.

Douglas Turnbull, Luke Barrington, David Torres, and Gert Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech and Language Processing*, 16(2):467–476, February 2008.

G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5), July 2002.