Autotagger: A Model For Predicting Social Tags from Acoustic Features on Large Music Databases

Thierry Bertin-Mahieux
University of Montreal
Montreal, CAN
bertinmt@iro.umontreal.ca

Douglas Eck
University of Montreal
Montreal, CAN
douglas.eck@umontreal.ca

François Maillet
University of Montreal
Montreal, CAN
mailletf@iro.umontreal.ca

Paul Lamere
Sun Labs, Sun Microsystems
Burlington, Mass, USA
paul.lamere@sun.com*

August 15, 2008

Abstract

Social tags are user-generated keywords associated with some resource on the Web. In the case of music, social tags have become an important component of "Web 2.0" recommender systems, allowing users to generate playlists based on use-dependent terms such as *chill* or *jogging* that have been applied to particular songs. In this paper, we propose a method for predicting these social tags directly from MP3 files. Using a set of 360 classifiers trained using the online ensemble learning algorithm FilterBoost, we map audio features onto social tags collected from the Web. The resulting automatic tags (or *autotags*) furnish information about music that is otherwise untagged or poorly tagged, allowing for insertion of previously unheard music into a social recommender. This avoids the "cold-start problem" common in such systems. Autotags can also be used to smooth the tag space from which similarities and recommendations are made by providing a set of comparable baseline tags for all tracks in a recommender system. Because the words we learn are the same as those used by people who label their music collections, it is easy to integrate our predictions into existing similarity and prediction methods based on web data.

1 Introduction

Social tags are a key part of "Web 2.0" technologies and have become an important source of information for recommendation. In the domain of music, Web sites such as Last.fm[1] use social tags to help their users find artists and music (Lamere [20]). In this paper, we propose a method for predicting social tags using audio feature extraction and supervised learning. These automatically-generated tags (or "autotags") can provide information about music for which good, descriptive social tags are lacking. Using traditional information retrieval techniques a music recommender can use these autotags (combined with any available listener-applied tags) to predict artist or song similarity. The tags can also serve to smooth the tag space from which similarities and recommendations are made by providing a set of comparable baseline tags for all artists or songs in a recommender.

This paper presents "Autotagger", a machine learning model for automatically applying text labels to audio. The model is trained using social tags, although it is constructed to work with any training data that fits in a classification framework. This work is an extension of Eck et al. [10, 11] which proposed an AdaBoost-based model for predicting autotags from audio features. The main contributions of this paper are as follows. First, we extend the model from

*Accepted for publication in the Journal of New Music Research (JNMR). Draft.

[1] www.last.fm

[10] to sample data from an arbitrarily large pool of audio files. This is achieved by replacing the AdaBoost batch learning algorithm with the FilterBoost online learning algorithm. Second, we explore two ways to take advantage of correlations among the tags collected from Last.fm in order to improve the quality of our automatically-generated tags. Finally we compare our approach to another approach on a new data set. All experimental results in this paper are new and make use of 360 autotags trained on a data set of approximately 100,000 MP3s.

This paper is organized as follows: in Section 2, we describe social tags in more depth, including a description of how social tags can be used to avoid problems found in traditional collaborative filtering systems, as well as a description of the tag set we built for these experiments. In Section 3, we describe previous approaches to automatic tagging of music and related tasks. In Section 4 we present our algorithm for autotagging songs based on labelled data collected from the Internet. In Sections 5 through 7, we present a series of experiments exploring the ability for the model to predict social tags and artist similarity. Finally, in Section 8, we describe our conclusions and future work.

2 Using social tags for recommendation

As the amount of online music grows, automatic music recommendation becomes an increasingly important tool for music listeners to find music that they will like. Automatic music recommenders commonly use collaborative filtering (CF) techniques to recommend music based on the listening behaviours of other music listeners. These CF recommenders (CFRs) harness the "wisdom of the crowds" to recommend music. Even though CFRs generate good recommendations there are still some problems with this approach. A significant issue for CFRs is the *cold-start* problem. A recommender needs a significant amount of data before it can generate good recommendations. For new music, or music by an unknown artist with few listeners, a CF recommender cannot generate good recommendations. Another issue is the *lack of transparency* in recommendations (Herlocker et al. [16]). A CF recommender cannot tell a listener why an artist was recommended beyond the superficial description: "people who listen to X also listen to Y."

Music listening occurs in many contexts. A music listener may enjoy a certain style of music when working, a different style of music when exercising and a third style when relaxing. A typical CF recommender does not take the listening context into account when recommending music. Ideally, a music listener should be able to request a music recommendation for new music that takes into account the style of the music and the listening context. Since a CF recommender bases its recommendations on listener behaviour, it cannot satisfy a music recommendation request such as "recommend new music with female vocals, edgy guitar with an indie vibe that is suitable for jogging."

An alternative style of recommendation that addresses many of the shortcomings of a CF recommender is to recommend music based upon the similarity of "social tags" that have been applied to the music. Social tags are free text labels that music listeners apply to songs, albums or artists. Typically, users are motivated to tag as a way to organize their own personal music collection. A user may tag a number of songs as *mellow* some songs as *energetic* some songs as *guitar* and some songs as *punk*. Typically, a music listener will use tags to help organize their listening. A listener may play their *mellow* songs while relaxing, and their *energetic* artists while they exercise.

The real strength of a tagging system is seen when the tags of many users are aggregated. When the tags created by thousands of different listeners are combined, a rich and complex view of the song or artist emerges. Table 1 shows the top 21 tags and frequencies of tags applied to the band "The Shins". Users have applied tags associated with the genre (*Indie*, *Pop*, etc.), with the mood (*mellow*, *chill*), opinion (*favorite*, *love*), style (*singer-songwriter*) and context (*Garden State*). From these tags and their frequencies we learn much more about "The Shins" than we would from a traditional single genre assignment such as "Indie Rock".

Using standard information retrieval techniques, we can compute the similarity of artists or songs based on the co-occurrence of tags. A recommender based upon the social tags addresses some of the issues seen in traditional CFRs. Recommendations are transparent — they can be explained in terms of tags. Recommendations can be sensitive to the listening context. A recommender based on social tags is able to cross the semantic gap, and allow a listener to request a recommendation based upon a textual description of the music. The recommender can satisfy a request to "recommend music with female vocals, edgy guitar with an indie vibe that is suitable for jogging". However, a social-tag-based recommender still suffers from the cold-start problem that plagues traditional CFRs. A new artist or song will have insufficient social tags to make good recommendations.

In this paper, we investigate the automatic generation of tags with properties similar to those assigned by social taggers. Specifically, we introduce a machine learning algorithm that takes as input acoustic features and predicts

Tag	Freq	Tag	Freq	Tag	Freq
Indie	2375	The Shins	190	Punk	49
Indie rock	1138	Favorites	138	Chill	45
Indie pop	841	Emo	113	Singer-songwriter	41
Alternative	653	Mellow	85	Garden State	39
Rock	512	Folk	85	Favorite	37
Seen Live	298	Alternative rock	83	Electronic	36
Pop	231	Acoustic	54	Love	35

Table 1: Top 21 tags applied to *The Shins* for a sample of tags taken from Last.fm.

social tags mined from the web (in our case, Last.fm). The model can then be used to tag new or otherwise untagged music, thus providing a partial solution to the cold-start problem.

For this research, we extracted tags and tag frequencies from the social music website Last.fm using the Audioscrobbler web services [2] during the spring of 2007. The data consists of over 7 million artist-level tags applied to 280,000 artists. 122,000 of the tags are unique. Table 2 shows the distribution of the types of tags for the 500 most frequently applied tags. The majority of tags describe audio content. Genre, mood and instrumentation account for 77% of the tags. This bodes well for using the tags to predict audio similarity as well as using audio to predict social tags. However, there are numerous issues that can make working with tags difficult. Taggers are inconsistent in applying tags, using synonyms such as *favorite*, *favourite* and *favorites*. Taggers use personal tags that have little use when aggregated (*i own it*, *seen live*). Tags can be ambiguous; *love* can mean a romantic song or it can mean that the tagger loves the song. Taggers can be malicious, purposely mistagging items (presumably there is some thrill in hearing lounge singer Barry Manilow included in a death metal playlist). Taggers can purposely mistag items in an attempt to increase or decrease the popularity of an item. Although these issues make working with tags difficult, they are not impossible to overcome. Some strategies to deal with these are described in Guy and Tonkin's article [15].

A more difficult issue is the uneven coverage and sparseness of tags for unknown songs or artists. Since tags are applied by listeners, it is not surprising that popular artists are tagged much more frequently than non-popular artists. In the data we collected from Last.fm, "The Beatles" are tagged 30 times more often than "The Monkees". This sparseness is particularly problematic for new artists. A new artist has few listeners, and therefore, few tags. A music recommender that uses social tags to make recommendations will have difficulties recommending new music because of the tag sparseness. This cold-start problem is a significant issue that we need to address if we are to use social tags to help recommend new music.

Overcoming the cold-start problem is the primary motivation for this area of research. For new music or sparsely tagged music, we predict social tags directly from the audio and apply these automatically generated tags (called *autotags*) in lieu of traditionally applied social tags. By automatically tagging new music in this fashion, we can reduce or eliminate much of the cold-start problem. More generally, we are able to use these autotags as part of a music recommender to recommend music from the "long tail" of the distribution [18] over popular artists and thus introduce listeners to new or relatively unknown music.

Given that our approach needs social tag data to learn from, it is not a complete solution for the cold-start problem. For a new social recommender having no user data at all, it would be necessary to obtain some initial training data from an external source. Given that many useful sources such as Audioscrobbler are freely available only for non-commercial use, this may be impossible or may require a licensing agreement.

3 Previous Work and Background

In this section we discuss previous work on music classification and music similarity. In Section 3.1 we carry out an overview of the existing work in genre recognition. Then, in Section 3.2, we discuss issues relating to measuring similarity, focusing on challenges in obtaining ground truth. Finally we provide details about the similarity distance measures used in many of our experiments.

Tag Type	Frequency	Examples
Genre	68%	heavy metal, punk
Locale	12%	French, Seattle, NYC
Mood	5%	chill, party
Opinion	4%	love, favorite
Instrumentation	4%	piano, female vocal
Style	3%	political, humor
Misc	3%	Coldplay, composers
Personal	1%	seen live, I own it

Table 2: Distribution of tag types for the Last.fm tag sample.

3.1 Music Classification and Similarity

A wide range of algorithms have been applied to music classification tasks. Lambrou et al. [19], and Logan and Salomon [23] used minimum distance and K-nearest neighbours. Tzanetakis and Cook [33] used Gaussian mixtures. West and Cox [35] classify individual audio frames by Gaussian mixtures, Linear Discriminant Analysis (LDA), and regression trees. Ahrendt and Meng [1] classify 1.2s segments using multiclass logistic regression. In Bergstra et al. [7], logistic regression was used to predict restricted "canonical" genre from the less-constrained noisy genre labels obtained from the FreeDb web service.

Several classifiers have been built around Support Vector Machines (SVMs). Li et al. [22] reported improved performance on the same data set as [33] using both SVM and LDA. Mandel and Ellis [25] used an SVM with a kernel based on the symmetric KL divergence between songs. Their model performed particularly well at MIREX 2005[2], winning the Artist Recognition contest and performing well in the Genre Recognition contest. While SVMs are known to perform very well on small data sets, the quadratic training time makes it difficult to apply them to large music databases. This motivates research on applying equally well-performing but more time-efficient algorithms to music classification.

The ensemble learning algorithm AdaBoost was used in Bergstra et al. [6] to predict musical genre from audio features. One contribution of this work was the determination of the optimal audio segmentation size for a number of commonly-used audio features and classifiers. This model won the MIREX 2005 genre contest [5] and performed well in the MIREX 2005 artist recognition contest. A similar boosting approach was used in Turnbull et al. [31] to perform musical boundary detection. As mentioned in Section 1, AdaBoost was the algorithm used in Eck et al. [10]. FilterBoost, an online version of AdaBoost which uses rejection sampling to sample an arbitrarily large data set, is used in the current work. See Section 4.2 for details.

Though tasks like genre classification are of academic interest, we argue in our analysis of user tagging behaviour (Section 2) that such winner-take all annotation is of limited real-world value. A similar argument is made in McKay and Fujinaga [27]. For a full treatment on issues related to social tags and recommendation see Lamere's article [20].

3.2 Collecting Ground-Truth Data for Measuring Music Similarity

Measuring music similarity is of fundamental importance for music recommendation. Our approach as introduced in [10] is to use distance between vectors of autotags as a similarity measure. Though our machine learning strategies differ, this approach is similar to that of Barrington et al. [3] which used distance between semantic concept models (similar to our autotags) as a proxy for acoustic similarity. Their approach performed well at MIREX in 2007, finishing third in the similarity task out of 12 with no significant difference among the top four participants. See Section 6 for a comparison of our approach and that of Barrington.

As has long been acknowledged (Ellis et al. [12]), one of the biggest challenges in predicting any attribute about music is obtaining "ground truth" for comparison. For tasks like genre prediction or social tag prediction, obtaining ground truth is challenging but manageable. (For genre prediction an ontology such as provided by AllMusic can be

[2]Music Information Retrieval Evaluation eXchange; Yearly contest pages found at `www.music-ir.org`.

used; for social tag prediction, data mining can be used). The task is more complicated when it comes to predicting the similarity between two songs or artists.

What all researchers want, it is safe to say, is a massive collection of error-free human-generated similarity rankings among all of the songs and artists in the world, in other words a *large and clean* set of ground-truth rankings that could be used both to train and to evaluate models. Though no such huge, pristine similarity data set exists, it is currently possible to obtain either small datasets which are relatively noise-free or large datasets which may contain significant noise.

In general *small and clean* approaches take advantage of a well-defined data collection process wherein explicit similarity rankings are collected from listeners. One option is to use a controlled setting such as a psychology laboratory. For example, Turnbull et al. [29] paid subjects to provide judgements about the genre, emotion and instrumentation for a set of 500 songs. Another increasingly-popular option is to use an online game similar to the now-famous ESP game for images (von Ahn and Dabbish [34]) where points are awarded for describing an image using the same words as another user. Variations of the ESP game for music can be seen in [21, 26, 32].

If one of these games explodes in popularity it has great potential for generating exactly the kinds of large and clean datasets we find useful. In the meantime, large dataset collection techniques are done via data mining of public web resources and thus are not driven by elicited similarity judgements. Our approach uses such a *large and noisy* data collection technique: the word distributions used to train our autotag classifiers come from the Last.fm website, which does nothing to ensure that users consider music similarity when generating tags. Thus it is possible that the word vectors we generate will be noisy in proportion to the noise encountered in our training data. Our belief is that in the context of music similarity a large, noisy dataset will give us a better idea of listener preferences than will a small, clean one. This motivated the construction of our model, which uses classification techniques that scale well to large high-dimensional datasets but which do not in general perform as well on small datasets as do some other more-computationally expensive generative models. We will return this issue of *small and clean* versus *large and noisy* in section 6.

3.3 Measuring Similarity

In our experiments we use three measures to evaluate our ability to predict music similarity. The first, *TopN*, compares two ranked lists: a target "ground truth" list A and our predicted list B. This measure is introduced in Berenzweig et al. [4], and is intended to place emphasis on how well our list predicts the top few items of the target list. Let k_j be the position in list B of the jth element from list A. $\alpha_r = 0.5^{1/3}$, and $\alpha_c = 0.5^{2/3}$, as in [4]. The result is a value between 0 (dissimilar) and 1 (identical top N),

$$s_i = \frac{\sum_{j=1}^{N} \alpha_r^j \alpha_c^{k_j}}{\sum_{l=1}^{N} (\alpha_r \times \alpha_c)^l} \tag{1}$$

For the results produced below, we look at the top $N = 10$ elements in the lists.

Our second measure is Kendall's Tau, a classic measure in collaborative filtering which measures the number of discordant pairs in 2 lists. Let $R_A(i)$ be the rank of the element i in list A, if i is not explicitly present, $R_A(i) = length(A)+1$. Let C be the number of concordant pairs of elements (i, j), e.g. $R_A(i) > R_A(j)$ and $R_B(i) > R_B(j)$. In a similar way, D is the number of discordant pairs. We use Kendall's Tau's definition from Herlocker et al. [17]. We also define T_A and T_B the number of ties in list A and B. In our case, it's the number of pairs of artists that are in A but not in B, because they end up having the same position $R_B = length(B) + 1$, and reciprocally. Kendall's Tau value is defined as:

$$\tau = \frac{C - D}{sqrt((C + D + T_A)(C + D + T_B))} \tag{2}$$

Unless otherwise noted, we analyzed the top 50 predicted values for the target and predicted lists.

Finally, we compute what we call the *TopBucket*, which is simply the percentage of common elements in the top N of 2 ranked lists. Here we compare the top 20 predicted values unless otherwise noted.

4 Autotagger: an Automatic Tagging Algorithm using FilterBoost

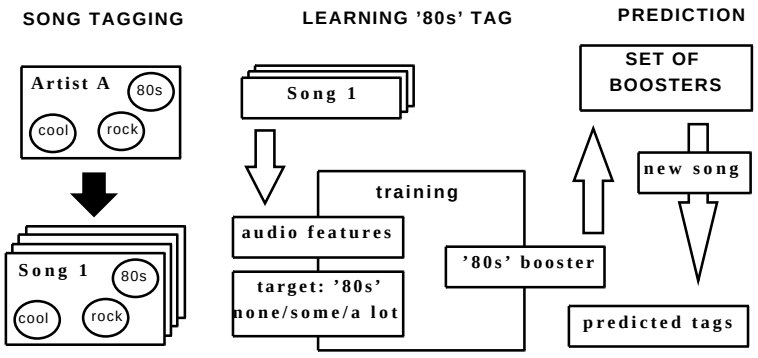

Figure 1: Overview of our model. A booster is learnt for every tag, and can then be use to "autotag" new songs.

We now describe a machine learning model which uses the *meta-learning* algorithm FilterBoost to predict tags from acoustic features. This model is an extension of a previous model [10], the primary difference being the use of FilterBoost in place of AdaBoost. FilterBoost is best suited for very large amounts of data. See Figure 1 for an overview.

4.1 Acoustic Feature Extraction

The features we use include 20 Mel-Frequency Cepstral Coefficients, 176 autocorrelation coefficients of an onset trace sampled at $100Hz$ and computed for lags spanning from 250msec to 2000msec at 10ms intervals, and 85 spectrogram coefficients sampled by constant-Q (or log-scaled) frequency (see previous work [6] or Gold and Morgan [14] for more details). for descriptions of these standard acoustic features.)

The audio features (Figure 2) described above are calculated over short windows of audio ($\sim$100ms with 25ms overlap). This yields too many features per song for our purposes. To address this, we create "aggregate" features by computing individual means and standard deviations (i.e., independent Gaussians) of these features over 5s windows of feature data. When fixing hyperparameters for these experiments, we also tried a combination of 5s and 10s features, but saw no real improvement in results. For reasons of computational efficiency we used random sampling to retain a maximum of 12 aggregate features per song, corresponding to 1 minute of audio data.

4.2 AdaBoost and FilterBoost

AdaBoost [13] is an *ensemble* (or *meta-learning*) method that constructs a classifier in an iterative fashion. It was originally designed for binary classification, and it was later extended to multi-class classification using several different strategies.

In each iteration t, the algorithm calls a simple learning algorithm (the *weak learner*) that returns a classifier $h^{(t)}$ and computes its coefficient $\alpha^{(t)}$. The input of the weak classifier is a d-dimensional observation vector x containing the features described in Section 4.1, and the output of $h^{(t)}$ is a binary vector $x \in \{-1, 1\}^k$ over the k classes. If $h_\ell^{(t)} = 1$ the weak learner "votes for" class ℓ whereas $h_\ell^{(t)} = -1$ means that it "votes against" class ℓ. After T iterations, the algorithm outputs a vector-valued discriminant function

$$g(x) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(x) \tag{3}$$

Assuming that the feature vector values are ordered beforehand, the cost of the weak learning is $O(nkd)$ (n number of examples), so the whole algorithm runs in $O\big(nd(kT + \log n)\big)$ time. Thus, though boosting may need relatively

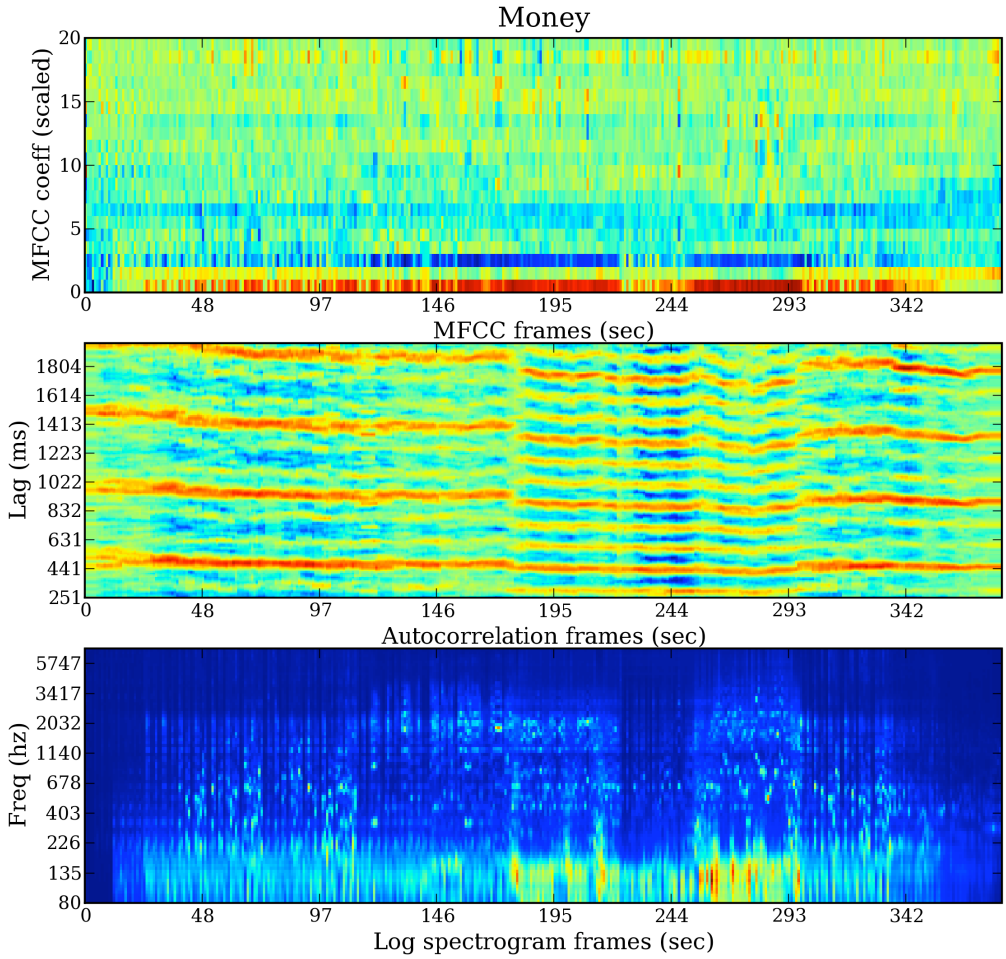

Figure 2: Acoustic features for "Money" by Pink Floyd.

more weak learners to achieve the same performance on a large data set than a small one, the computation time for a single weak learner scales linearly with the number of training examples. Thus AdaBoost has the potential to scale well to very large data sets.

FilterBoost [9] is an extension to AdaBoost which provides a mechanism for doing rejection sampling, thus allowing the model to work efficiently with large data sets by choosing training examples based on their similarity. The addition of rejection sampling makes it possible to use FilterBoost with data sets which are too large or too redundant to be used efficiently in a batch learning context. This is the case for industrial music databases containing a million or more tracks and thus tens or hundreds of millions of audio segments.

Data is presumed to be drawn from an infinitely large source called an "oracle". The filter receives a sample (x, l) from the oracle at iteration $t + 1$ and accepts it with a probability:

$$q_t(x, l) = \frac{1}{1 + e^{l g_t(x)}} \tag{4}$$

l being the true class of x, $l \in \{-1, +1\}$, and $g(x)$ the output of the booster between -1 and 1. Sampling continues until a small data set (usually 3000 examples in our experiments) is constructed, at which time we select the best weak learner $h^{(t+1)}(x)$ on this set, and then evaluate $h^{(t+1)}(x)$ weight by sampling again from the oracle (see [9] for details).

It is also possible to select more than one weak learner each round, using the classical AdaBoost weighting method on the small data set created. Conceptually, all the weak learners selected in a single pass can be considered as one single learner by grouping them. In our experiments, we choose 50 weak learners per round.

Regardless of whether AdaBoost or FilterBoost is employed, when no "a priori" knowledge is available for the problem domain, small decision trees or, in the extreme case, *decision stumps* (decision trees with two leaves) are often used as weak learners. In all experiments reported here, decision stumps were used. In earlier experiments we also tried decision trees without any significant improvement. Because decision stumps depend on only one feature, when the number of iterations T is much less than the number of features d, then the booster acts as an implicit feature extractor that selects the T most relevant features to the classification problem. Even if $T > d$, one can use the coefficients $\alpha^{(t)}$ to order the features by their relevance. Because a feature is selected based on its ability to minimize empirical error, we can use the model to eliminate useless feature sets by looking at the order in which those features are selected. We used this property of the model to discard many candidate features such as chromagrams (which map spectral energy onto the 12 notes of the Western musical scale) because the weak learners associated with those features were selected very late by the booster.

4.3 Generating Autotags using Booster Outputs

Each booster is trained using individual audio segments as input (Figure 1). However we wish to make predictions on the level of tracks and artists. In order to do so we need to integrate segment-level predictions into track and artist level predictions. One way to do this is to take the mean value of the hard discriminant function $\mathbf{sign}(g(x))$ for all segments. Instead we take the mean or median of the raw discriminant function (i.e., the sum of the weak learner predictions) $g(x)$ for all segments. So that the magnitudes of the weak learner predictions are more comparable we normalize the sum of the weak learner weights α to be 1.0. This yields a song-level prediction scaled between 0 and 1 where .5 is interpreted as incertitude.

This normalization is useful in that it allows us to use and compare all words in our vocabulary. Lacking normalization, difficult-to-learn words tend not to have any impact at all because the booster confidences are so low. The undesirable side-effect of our approach is that the impact of very poorly learned tags is no longer attenuated by low learner confidence. In a production system it would be important to filter out such tags so they do not contribute undue noise. Though we used no such filtering for these experiments, it can be easily implemented by discarding tags which fall below a threshold out-of-sample classification rate.

4.4 Second-stage learning and correlation reweighting

As discussed above, each social tag is learned independently. This simplifies our training process in that it allows us to continually update the boosters for a large set of tags, thus spreading out the computation load over time. Furthermore it allows us easily add and subtract individual tags from our set of relevant tags as the social tagging data changes over time. If the tag-level models were dependant on one another this would be difficult or impossible. It is clear, however, that an assumption of statistical independence among tags is false. For example, a track labelled "alternative rock" is also likely to be labelled "indie rock" and "rock". By ignoring these interdependencies, we make the task of learning individual tags more difficult. We test two techniques for addressing this issue.

Our first approach uses a second set of boosted classifiers. These "second-stage" learners are trained using the autotag predictions from the first stage. In other words, each second-stage booster predicts a single social tag using a weighted mixture of acoustically-driven autotags. Since the input includes the results from the first stage of learning, convergence is fast.

In our second approach we calculate the empirical correlation among the social tagging data obtained from Last.fm. We then adjust our predictions for a tag (whether from the single-stage or two-stage approach) by mixing predictions from other tags proportional to correlation. The correlation matrix is computed once for the entire Last.fm data set and applied uniformly to all autotags for all songs.

4.5 Generating Labelled Datasets for Classification from Audioscrobbler

We extracted tags and tag frequencies for nearly 100,000 artists from the social music website Last.fm using the Audioscrobbler web service [2]. From these tags, we selected the 360 most popular tags. Those tags are listed in the appendix (Section 11). Because it was impossible to obtain a sufficient number of song-level tags, only artist tags were used. This is admittedly a questionable practice, especially for artists whose work spans many different styles. For example, the Beatles are currently the number four artist for the tag "psychedelic" yet only a few Beatles songs fit that description. Currently there are many more tags applied to artists than to tracks. As more track-level tags become available we will take advantage of them.

Intuitively, automatic labelling would be a regression task where a learner would try to predict tag frequencies for artists or songs. However, because tags are sparse (many artist are not tagged at all; others like Radiohead are heavily tagged) this proves to be too difficult using our current Last.fm data set. Instead, we chose to treat the task as a classification one. Specifically, for each tag we try to predict if a particular artist would or would not be labelled with a particular tag. The measure we use for deciding how well a tag applies to an artist is:

$$\text{weight} = \frac{\#\text{ times this tag was applied to this artist}}{\#\text{ times any tags was applied to this artist}} \tag{5}$$

In our previous work [11], class labels were generated by dividing tags into three equal-sized sets (e.g. *no rock*, *some rock* or *a lot of rock*). With this strategy, hundreds or even thousands of artists appeared as positive examples. In our current work, we chose to select *positive examples* from only the top 10 artist for any given tag. The remaining artists which received enough tags to make Audioscrobbler's top 1000 list for that tag are treated as *ignore examples* which are not used for learning but which are used to test model performance. The set of *negative examples* are drawn by randomly sampling from all artists in our music collection which did not make the top 1000 list for a tag. With this strategy in many instances such as "rock" a booster is trained on only a tiny proportion of the valid positive artists suggested by Last.fm, resulting in a "rock" autotag that will certainly fail to find some salient characteristics of the genre, having never seen a large number of positive examples. This is in keeping with our goal to generate a large set of autotags which each succeeds at modeling a relatively narrow, well-defined subspace.

Regardless of the specific strategy we use for generating datasets, the set of positive examples for a tag will always be much smaller than the set of negative examples. This extreme imbalance suggests that we should preserve as many positive examples as possible, thus motivating our decision to use all top 10 artist songs for training. In addition, when training we sample equally between positive and negative training examples, thus artificially balancing the sets.

All of the music used in these experiments is labelled using the free MusicBrainz[3] service. The MusicBrainz track, album and artist ids used in our experiments are available on request.

5 Predicting Social Tags

In our first experiment, we measure booster predictions on the positive list, the ignore list and randomly-selected examples from the negative list. Recall that the positive list for a tag T is made of the songs for the 10 artists whose weight (equation 5) for that tag is the highest. The ignore list is made of all the songs for artists with high weight for that tag, but not enough to be in the top 10. Negative examples are drawn from the rest of the database. For a tag like *rock*, we have 10 positive artists, about 900 ignored artists and 3000 negative ones. Results are presented in tables 3 and 4. Table 3 displays success rate as a percentage of correctly classified songs. In parentheses we also show classification rates for second-stage boosting as discussed in section 4.4. Song-level predictions were obtained by taking the median for all segment-level predictions for that song. The results for the positive list can be seen as measuring training error, but the two other measures give an idea of how well we generalize: we did not train on any tracks from the ignore list, and we randomly sampled songs from 120K negative ones during training, so there is little risk of overfitting on 200 random songs. Table 4 provides examples of normalized booster outputs. In both tables, we also provide the results for selected genres, instrument and mood-related terms. Again, parentheses show results for second-stage learning. From a computational point of view, we train 2000 weak learners per word, as we did in earlier work [11]. However, FilterBoost computes them in a couple of hours instead of 1 or 2 days previously.

[3] www.musicbrainz.org

5.1 Second-stage learning

Second-stage learning results were obtained by training FilterBoost for 500 iterations. As there were only 360 autotag values present in the input vector, the booster could capture the influence of every autotag if necessary. The classification results in parentheses (Table 3) show improved performance for positive and negative examples but degraded performance for for the ignore list. The mean normalized booster outputs in Table 4 suggest that the second-stage boosting is more strongly separating the positive and negative classes. For more results on our second-stage learning, see Section 6 and Section 7.

Boosters	Positive (10 artists)	Ignore (100 songs)	Negative (200 songs)
main genres (rock, pop, Hip-Hop, metal, jazz, dance, Classical, country, blues, reggae)	89.1% (90.1%)	80.6% (78.9%)	80.0% (79.1%)
instruments (piano, guitar, saxophone, trumpet)	87.0% (88.7%)	61.0% (60.3%)	82.4% (83.6%)
mood (happy, sad, romantic, Mellow)	87.8% (89.1%)	67.4% (66.8%)	79.0% (81.1%)
all	**88.2**% (90.5%)	**60.0**% (57.2%)	**81.4**% (84.1%)

Table 3: Song classification results, percentage of the songs that are considered positive (for positive and ignore examples) or negative for negative examples. Values in parentheses are for second-stage boosters.

Boosters	Positive (10 artists)	Ignore (100 songs)	Negative (200 songs)
main genres (rock, pop, Hip-Hop, metal, jazz, dance, Classical, country, blues, reggae)	0.540 (0.572)	0.528 (0.544)	0.463 (0.438)
instruments (piano, guitar, saxophone, trumpet)	0.538 (0.570)	0.507 (0.520)	0.470 (0.438)
mood (happy, sad, romantic, Mellow)	0.532 (0.558)	0.511 (0.517)	0.463 (0.433)
all	**0.536** (0.569)	**0.508** (0.509)	**0.466** (0.432)

Table 4: Mean normalized booster output per song for selected tag types. Values in parentheses are for second-stage boosters.

5.2 Correlation reweighting

To investigate the effects of reweighting autotag predictions as a function of empirical correlation, we look at the ordering of the artists taken from the ignore list from our training set. Recall that ignore-list artists for a tag T are the ones that were labelled significantly with tag T, but not enough to appear among the top 10 artists (i.e the positive list for tag T). We assume that having a good ordering of these artists, e.g. "from the most *rock* to the least *rock*", is a good measure of performance.

We generate a ground-truth list by sorting the Last.fm tags by their weight (Equation 5). We then compare this ground truth to three lists: a random list, a list ordered by our autotags and a list ordered by correlation-reweighted autotags. Lists are sorted by the median normalized booster output per song over all songs for an artist. Results are shown in Table 5.

Autotags	TopN 10	Kendall 50	TopBucket 20
random artist order	-0.663	0.003	0.007%
without correlation	-0.577	0.024	5.64%
with correlation	-0.569	0.027	6.33%

Table 5: Ordering of artists (in the ignore list) per tag. Ordering is based on median normalized booster output.

The correlation reweighting yields improved performance on all three measures we tested. However the improvement is relatively minor (e.g. less than 2% for TopBucket 20). Many factors can explain why the improvement is not greater. First, our method for generating a ground-truth list yields only a general idea of good ordering. Second, Audioscrobbler [2] only gives us access to a limited number of tag counts. Having more data would increase the precision of our correlation factors. Third, the tagging is very sparse: most artists are tagged reliably with only a few of the 360 tags we investigated. This can lead to spurious correlation measures for otherwise unrelated tags. Finally, we trained on only 10 *positive* artists. For a general tag like *rock*, it is obvious that 10 artists cannot represent the whole genre. If all positive artists for *rock* are rockers from the 60s, there is not much chance that Radiohead (heavily tagged *rock*) will be correctly placed among others artists in the ignore list, being too different from the positive artists.

6 Comparison with GMM approach

We now compare our model to one developed by the Computer Audition Lab (CAL) group that uses a hierarchical Gaussian mixture model (GMM) to predict tags [29, 30]. We make use of the same dataset ("CAL500") used in their experiments. One goal of this comparison is to investigate the relative merits of the *small and clean* versus *large and noisy* approaches discussed in Section 3.2. The experiments follow closely [30], and GMM results also come from this paper.

6.1 The CAL500 data set

The Computer Audition Lab 500-Song (CAL500) data set (Turnbull et al. [29, 30]) was created by the CAL group by having 66 paid students listen to and annotate a set of songs. The collection is made of 500 recent Western songs by 500 different artists in a wide range of genres. The tags applied to the corpus can be grouped into six categories: instrumentation, vocal characteristics, genre, emotions, acoustic qualities and usage terms. The data was collected by presenting the students with an HTML form comprised of a fixed set of 135 tags. This is quite different from the Last.fm tags used in our previous experiments because the respondents were constrained to a predetermined set of words, resulting in cleaner tags. Some of the tags received a rating on a scale of 1 to 5 (ex.: emotion-related tags), others on a 1 to 3 scale (ex.: presence of a instrument could be marked "yes", "no" or "uncertain") and some other received binary ratings (ex.: genre-related tags). There were a total of 1708 song evaluations collected with a minimum of three per song. All bipolar tags were then broken down into two different tags (thus generating more than the original 135 tags). For example, "The song is catchy" was broken down to "catchy" and "not catchy", with ratings of 1 and 2 counting towards the "not catchy" tag, ratings of 4 and 5 counting towards the "catchy" and the ratings of 3 being simply ignored.

The ground truth was created by assigning a tag to a song if a minimum of two people assigned that tag to the song and if there was an agreement between the different survey respondents. The respondents were considered in agreement if

$$\left[\frac{\#(\text{People who assigned tag to song}) - \#(\text{People who didn't})}{\#(\text{People who evaluated song})}\right] > 0.8. \tag{6}$$

As a final step, all the tags that were assigned to less than five songs were pruned, which resulted in a collection of 174 tags.

6.2 Evaluation

We trained and evaluated our model in the same way as did the CAL group, using 10 fold cross-validation on the 500 songs (i.e., 450-song training set, 50-song test set). We trained one booster per tag using different audio feature sets. First, we trained on the MFCC deltas provided with the CAL500 data set, which were the features used by the CAL group [30]. We then trained on our aggregated audio features (afeats) and on our autotags, creating second-stage autotags (bfeats).

Category	Avg. # positive examples		Avg. # positive after expansion		Avg. # negative examples	
All words	69.16	(74.50)	85.60	(63.42)	382.84	(74.50)
Emotion	128.46	(59.28)	129.21	(58.09)	323.54	(59.28)
Genre	25.06	(25.91)	52.87	(13.96)	435.13	(11.78)
Instrumentation	70.35	(79.42)	84.80	(70.71)	381.65	(79.42)
Solo	12.70	(9.79)	48.59	(1.48)	439.30	(9.79)
Usage	27.38	(27.31)	52.91	(16.59)	424.62	(27.31)
Vocal	34.44	(28.63)	55.13	(17.19)	417.56	(28.63)

Table 6: Average per-fold number of positive, negative and positive examples after expansion in the CAL500 data set. The numbers is parentheses are the standard deviation. The expansion of the "Solo" tags averages to a number inferior to 50 because we did not have enough songs by the artists in the original positives examples in our own music collection. For example, if there is only one positive example and we do not have any additional song by that artist, we will not be able to do any expansion on that fold.

Since the CAL500 data set is relatively small, some tags have very few positive examples (i.e., 5 positives for 445 negatives) in certain folds. To explore the influence of the number of examples, we tried *expanding* the training set by adding random songs from the artists that were already part of the positive examples, so that every fold had at least 50 positive examples. The new songs were taken from our internal research collection. The training on the expanded training set was done using the afeats (afeats exp.). The average number of positive, negative and positive examples after expansion are listed in Table 6. The test set was left unchanged.

6.3 Results

We discuss the results for annotation and for retrieval separately in the following two sections.

6.3.1 Results for Annotation

This section treats the task of annotating a given track with an appropriate set of tags. The annotation evaluation and comparison of the model was done using the two evaluation measures used in Turnbull et al. [30], mean per-tag precision and recall, as well as a third one, the F-score. All three are standard information retrieval metrics. We start by annotating each song in our test set with an arbitrary number of tags that we refer to as the annotation length. Since the CAL group used ten tags, we used that number as well for comparison purposes. Per-tag precision can be defined as the probability that the model annotates relevant songs for a given tag. Per-tag recall can be defined as the probability that the model annotates a song with a tag that should have been annotated with that tag. More formally, for each tag t, $|t_{GT}|$ is the number of songs that have been annotated with the tag in the human-generated "ground truth" annotation. $|t_A|$ is the number of songs that our model automatically annotates with the tag t. $|t_{TP}|$ is the number of "correct" (true positive) tags that have been used both in the ground truth and in the automatic tag prediction. Per-tag recall is $|t_{TP}|/|t_{GT}|$ and per-tag precision is $|t_{TP}|/|t_A|$. The F-score takes into account both recall and precision and is defined as $2 \cdot (\text{precision} \cdot \text{recall})/(\text{precision} + \text{recall})$. No variance is provided for our F-score measure because it was computed from the averaged precision and recall for all words; per-tag precision and recall values were not available for the GMM. All three of these metrics range between 0 and 1 but are upper bounded by a value of less than one in our results because we forced the model to output less tags than the number that are actually present in the ground truth. The upper bound is listed as "UpperBnd" in the results tables.

The results for the precision and recall scores are given in Table 7. In general the models were comparable, with Autotagger performing slightly better on precision for all feature sets while the GMM model performed slightly better for recall. Precision and recall results for different categories are found in the appendix (Section 11). Though a good balance of precision and recall is always desirable, it has been argued that precision is more important for recommender systems. See, for example, Herlocker et al. [16]. Also, training with the second-stage autotags (bfeats) as input to the boosters produced better precision and recall results than the afeats. This suggests that the extra level of abstraction given by the bfeats can help the learning process.

Category	$A/\|V\|$	Model	Precision		Recall		F-Score
All words	$10/174$	Random	0.144	(0.004)	0.064	(0.002)	0.089
		UpperBnd	0.712	(0.007)	0.375	(0.006)	0.491
		GMM	0.265	(0.007)	**0.158**	(0.006)	0.198
		MFCC delta	0.281	(0.066)	0.131	(0.019)	0.179
		afeats	0.266	(0.078)	0.094	(0.018)	0.139
		afeats exp.	**0.312**	(0.060)	0.153	(0.015)	**0.205**
		bfeats	0.291	(0.105)	0.089	(0.034)	0.136

Table 7: Music annotation results. A = Annotation length, $|V|$ = Vocabulary size. Numbers in parentheses are the variance over all tags in the category. GMM is the results of the Gaussian mixture model of the CAL group. MFCC delta, afeats, afeats extended and bfeat are the results of our boosting model with each of these features. Random, UpperBnd and GMM results taken from [30]. Continued in Table 14.

The annotation length of 10 tags used to compute the precision and recall results could be restrictive depending on the context in which the tagging is used. If the goal is to present a user with tags describing a song or to generate tags that will be used in a natural language template as the CAL group did, 10 tags seems a very reasonable choice. However, if the goal is to do music recommendation or compute similarity between songs, a much higher annotation length may give rise to better similarity measures via word vector distance. As shown in Figure 3, our recall score goes

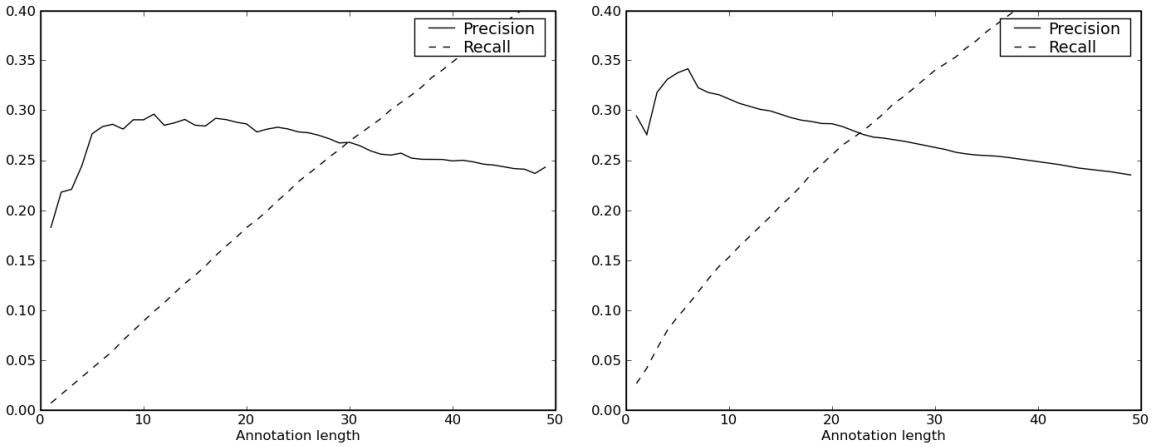

Figure 3: Precision and recall results for different annotation lengths when training on (a) the bfeats and (b) the expanded afeats.

up very quickly while our precision remains relatively stable when increasing the annotation length. This supports the idea that we can scale to larger annotation lengths and still provide acceptable results for music similarity and recommendation.

To provide an overall view of how Autotagger performed on the 35 tags we plot precision in Figure 4 for the different feature sets. This figure, and particularly the failure of the model to predict categories such as "World (Best)" and "Bebop" for certain feature sets, is discussed later in Section 6.4.

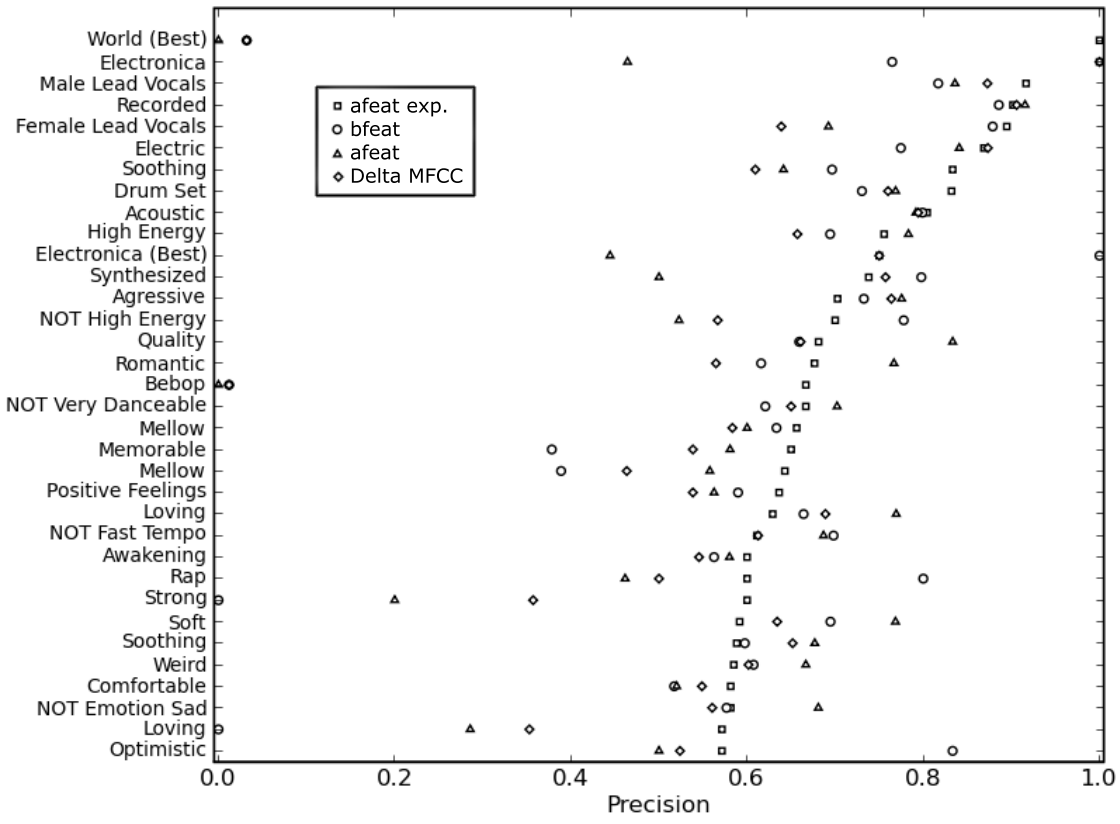

Figure 4: "Autotagger's" precision scores using different feature seats on 35 CAL500 tags ordered by performance when using the expanded afeats. Plot inspired by M. Mandel's one [24].

6.3.2 Results for Retrieval

In this section we evaluate our ability to retrieve relevant tracks for a given tag query. The evaluation measures used are the same as in Turnbull et al. [30]. To measure our retrieval performance, we used the same two metrics as the CAL group. They are the mean average precision and the area under the receiver operating characteristic curve (AROC). The metrics were computed on a rank-ordered test set of songs for every one-tag query t_q in our vocabulary V. Average precision puts emphasis on tagging the most relevant songs for a given tag early on. We can compute the average precision by going down the rank-ordered list and averaging the precisions for every song that has been correctly tagged in regard to the ground truth (true positive). The ROC curve is a plot of the fraction of true positives as a function of the fraction of false positives that we get when moving down our ranked list. The AROC is the area under the ROC curve and is found by integrating it. The mean of both these metrics can be found by averaging the results over all the tags in our vocabulary.

Table 8 shows the retrieval results for these measures. Here it can be seen that the GMM model slightly outperforms the Autotagging model but that results for "All words" are comparable. We also see that again the two-stage learner (bfeats) outperforms the single-stage learner.

Category	$\|V\|$	Model	MeanAP		MeanAROC	
All words	174	Random	0.231	(0.004)	0.503	(0.004)
		GMM	**0.390**	(0.004)	**0.710**	(0.004)
		MFCC delta	0.305	(0.057)	0.678	(0.015)
		afeats	0.323	(0.092)	0.622	(0.013)
		afeats exp.	0.385	(0.06)	0.674	(0.010)
		bfeats	0.340	(0.124)	0.662	(0.020)

Table 8: Retrieval results. $|V|$ = Vocabulary size. Numbers in parentheses are the variance over all tags in the category. GMM is the results of the Gaussian mixture model of the CAL group. MFCC delta, afeats, afeats extended and bfeat are the results of our boosting model with each of these features. Random and GMM taken from [30]. Continued in Table 15.

6.4 Discussion

We now compare the tags generated by Autotagger (using two-stage learning on expanded afeats) to several other lists on the Red Hot Chili Peppers song "Give It Away". Annotations from the GMM model are taken from [30]. The results are presented in Table 9. Here we observe that the Autotagger list includes words which would be difficult or impossible to learn such as "good music" and "seen live." This suggests that filtering out tags with low classification rates would improve performance for annotation.

GMM	CAL500 words	autotags	Last.fm Tags
dance pop	not mellow	good music	Rock
aggressive	not loving	pop rock	90s
not calming	exercising	Funk Rock	Alternative Punk
angry	rapping	crossover	Funk
exercising	monotone	rock	Alternative
rapping	tambourine	USA	Alternative Rock
heavy beat	at work	Favorite Artists	Funk Rock
pop	gravelly	american	Hard Rock
not tender	hard rock	seen live	Punk
male vocal	angry	classic rock	Funk Metal

Table 9: Top 10 words generated for the Red Hot Chili Peppers song *Give it Away* first by the hierarchical Gaussian mixture (GMM) from CAL group, then by our model trained with the extended afeats with CAL500 tags, by our model (using second-stage boosters) with Last.fm tags and finally the top tags on Last.fm. Ordering for GMM is approximated.

An important observation is that our model achieves its best performance with the expanded afeats. This provides (modest) evidence in support of large and noisy datasets over small and clean ones. Despite the fact that these audio examples were not labelled by the trained listeners, by adding them to the training set, we improve our performance on the unmodified test set and end up doing better than the GMM model on almost every evaluation metric. We even improve our precision results for tags like "Acoustic Guitar Solo" by considering all the songs added to the training set as having an acoustic guitar solo, which is an assumption that can potentially be wrong most of the time.

One challenge in working with the CAL500 data is that only 3.4 listeners on average rated each song. For example, the song "Little L" by Jamiroquoi was tagged by three different persons who disagreed strongly on some tags like "Drum machine", which was annotated as "None" by one student and as "Prominent" by another. This may introduce problems when using these annotations as ground truth. This issue is addressed by requiring an agreement of 80% among respondents in order to apply a tag to a song. However with only an average of 3.4 survey respondents per song, most of the time all respondents need to tag a song as positive for the tag to be applied in the ground truth. Since

the survey participants are not professional music reviewers, it is reasonable to assume that there will be significant disagreement. This issue is easily addressed by obtaining many more annotations per song.

One repercussion of this problem is illustrated in Figure 4, which shows our model's precision results on 35 tags using the different feature sets. Since the tags are ordered by their precision score when using the expanded afeats, some of them stand out by having a drastic performance difference when using the expanded afeats or another feature set. In most cases, these outliers stem from having very few positive examples in the training set for these tags. For example, the tags *World (Best)* and *Bebop* respectively have an average of 1.6 and 0.6 per-fold positive examples in the training set. Following the training set expansion and using the afeats in both cases, their precision went from 0.03 to 1.0 for *World (Best)* and from 0.01 to 0.67 for *Bebop*.

Overall we can conclude by looking at Table 7 and Table 8 that the performance of the Autotagger model and the GMM model are comparable, with the GMM performing slightly better at recall while the Autotagger model performs slightly better on precision. However we hasten to add that the best Autotagger results are had *when the expanded feature set is used* and that without more data, the GMM approach performs better. In terms of comparing the algorithms themselves, this is not a fair comparison because it is likely that the GMM approach would also perform better when trained on the expanded feature set. In this sense neither algorithm can be said to be strictly better. A comparison of these two approaches using a realistically-large dataset for music recommendation (> 500K songs and thus millions of audio frames) is called for.

7 Application to Similarity

As mentioned in section 3.3, one key area of interest lies in using our autotags as a proxy for measuring perceived music similarity. By replacing social tag-based similarity with autotag-based similarity we can then address the cold start problem seen in large-scale music recommenders[4]. In the following experiments, we measure our model's capacity to generate accurate artist similarities.

7.1 Ground Truth

As has long been acknowledged (Ellis et al. [12]), one of the biggest challenges in addressing this task is to find reasonable "ground truth" against which to compare our results. We seek a similarity matrix among artists which is not overly biased by current popularity, and which is not built directly from the social tags we are using for learning targets. Furthermore we want to derive our measure using data that is freely available on the web, thus ruling out commercial services such as the AllMusic Guide [5]. Our solution is to construct our ground truth similarity matrix using correlations from the listening habits of Last.fm users. If a significant number of users listen to artists A and B (regardless of the tags they may assign to that artist) we consider those two artists similar. Note that, although these data come from the same web source as our artist-level training data, they are different: we train our system using tags applied to artists, regardless of which user applied the tag.

One challenge, of course, is that some users listen to more music than others and that some artists are more popular than others. We use term frequency-inverse document frequency (TF$\times$IDF) weighting scheme to overcome this issue. The complete description of how we build this ground truth can be found in Eck et al. [11].

We also use a second ground truth which has no connection to Last.fm. The All Music Guide (AMG) is a website containing a lot of information about music made by human experts, in particular lists of similar artists. Based on an idea from Ellis et al. [12] we calculate similarity using Erdös distances. If an artist $A1$ is similar to another artist $A2$, they have a distance of 1. If artist $A3$ is similar to $A2$, $A1$ and $A3$ have a distance of 2, and so on. Put another way, it is the number of steps to go from one artist to another in a connected graph. We mined 4672 artists on AMG for these experiments.

[4] Of course, real recommenders deal with a more complex situation, caring about novelty of recommendations, serendipity and user confidence among others (see Herlocker et al. [17] for more details). However, similarity is essential. We do it on the artist level because the data available to build a ground truth would be too sparse on the album or song level.

[5] `www.allmusic.com`

7.2 Experiments

We construct similarity matrices from our autotag results and from the Last.fm social tags used for training and testing. The similarity measure we used was *cosine similarity* $s_{cos}(A_1, A_2) = A_1 * A_2/(||A_1||\ ||A_2||)$ where A_1 and A_2 are tag magnitudes for an artist. In keeping with our interest in developing a commercial system, we used all available data for generating the similarity matrices, including data used for training. (The chance of overfitting aside, it would be unwise to remove The Beatles from your recommender simply because you trained on some of their songs). The similarity matrix is then used to generate a ranked list of similar artists for each artist in the matrix. These lists are used to compute the measures describe in Section 3.3. Results are found in Table 10.

Groundtruth	Model	TopN 10	Kendall 50	TopBucket 20
Last.fm	social tags	0.437	−0.057	42.98%
	2nd-stage autotags	0.149	−0.361	19.9%
	autotags	0.140	−0.381	18.5%
	random	0.006	−0.626	2.0%
AMG	social tags	0.234	−0.287	26.8%
	2nd-stage autotags	0.109	−0.431	15.3%
	autotags	0.104	−0.445	14.2%
	random	0.006	−0.626	2.0%

Table 10: Performance against Last.fm (top) and AMG (bottom) ground truth.

7.3 Second-Stage Learning

The second-stage autotags (Table 10) are obtained by training a second set of boosted classifiers on the results of the first classifiers (that is, we train using autotags in place of audio features as input). This second step allows us to learn dependencies among tags. The results from the second-stage boosters for similarity are better than those of the first-stage boosters. This leads to the conclusion that there is much to gain from modeling dependencies among tags. However, this is largely an open question that needs more work: what model is the best for second-stage learning, and how can we best take advantage of correlation among tags?

Wilco	
Ground truth Last.fm	Sufjan Stevens, Elliott Smith, The Flaming Lips, The Shins, Modest Mouse
Ground truth AMG	The Bottle Rockets, Blue Rodeo, The Flying Burrito Brothers, Neko Case, Whiskeytown
Last.fm	Calixico, Grandaddy, Modest Mouse, Mercury Rev, Death Cab for Cutie
Autotags	Tuatara, Animal Collective, Badly Drawn Boy, Gomez, Elliott Smith
Autotags 2nd stage	Badly Drawn Boy, Animal Collective, Elliott Smith, Gomez, Tuatara

Table 11: Similar artists to Wilco from a) Last.fm ground truth b) AMG ground truth c) similarity from Last.fm tags d) similarity from autotags e) similarity from autotags second-stage.

The Beatles	
Ground truth Last.fm	Radiohead, The Rolling Stones, Led Zeppelin, Pink Floyd, David Bowie
Ground truth AMG	George Martin, The Zombies, Duane Eddy, The Yardbirds, The Rolling Stones
Last.fm	George Harrison, The Who, The Rolling Stones, Fleetwood Mac, The Doors
Autotags	The Rolling Stones, Creedence Clearwater Revival, Elvis Costello, Elvis Costello & The Attractions, Traffic
Autotags 2nd stage	The Rolling Stones, Creedence Clearwater Revival, Donovan, The Lovin' Spoonful, Elvis Costello

Table 12: Similar artists to The Beatles from a) Last.fm ground truth b) AMG ground truth c) similarity from Last.fm tags d) similarity from autotags e) similarity from autotags second-stage.

7.4 Discussion

It seems clear from these results that the autotags are of value. Though they do not outperform the social tags on which they were trained, it was shown in previous work (Eck et al. [11]) that they do yield improved performance when combined with social tags. At the same time, we showed a way to improve them by a second-stage of learning, and they are driven entirely by audio and so can be applied to new, untagged music.

Finally, we present some similar artists to Wilco and The Beatles in Tables 11 and 12, based on our two ground truths, Last.fm tags, and our two kind of autotags. We can draw two conclusions from these tables: Last.fm ground truth suffers from popularity bias, and our two set of autotag results are very comparable.

8 Conclusions

We have extended our previous autotagging method to scale more efficiently using FilterBoost. This introduces the concept of infinite training data, where an oracle can go and get the examples it needs. This is particularly appealing for web-based music recommenders that have access to millions of songs. The learning can simply be done using social tagging data, and the data we used [2] is freely available for research.

We tried to shed light on differences between *small and clean* versus *large and noisy* data sets. Though we provide no conclusive evidence to support the superiority of large, unreliably-labelled datasets such as our Last.fm data, we did demonstrate improved performance on the CAL500 task by adding audio which was never listened to by the CAL500 subjects and thus was not well-controlled. This is in keeping with the folk wisdom "There's no data like more data" and points towards methods which take advantage of all data available such as, e.g., semi-supervised learning and multi-task learning.

We have compared our method with the hierarchical mixture of Gaussians from the CAL group. This is to our knowledge the first comparison of algorithms especially designed for automatic tagging of music. In summary, Autotagger performed slightly better on precision for all feature sets while the GMM model performed slightly better for recall. We can in no way conclude from these results that one model is superior to the other. Test on larger datasets would be necessary to draw such conclusions. These results do support the conclusion that Autotagger has great potential as the core of a recommender that can generate transparent and steerable recommendation.

9 Future Work

One weakness of our current setup is that we blindly include all popular tags from Last.fm regardless of our ability to predict them. This adds significant noise to our similarity measurements. A solution proposed by Torres et al. [28]

may prove more effective than our proposal to simply remove tags which we cannot classify above some threshold. A comparison of these and other methods is an important direction for future research.

Another direction for future research is that of second-stage learning. Treating tags as being independent is a useful assumption when you train on large scale data and you want to be able to expand your vocabulary easily. However, we show it is still possible to take advantage of the dependencies among tags, which improves our similarity results. We showed modest increases in classification error and also higher booster confidence values with our second-stage learning approach. However, more work is necessary in this area.

Finally, it should be possible to use the similarity space created by our model to create playlists that move smoothly from one artist to another. In addition, we can draw on our autotag predictions to explain the song-to-song transitions. As a first step, we used ISOMAP (see Bishop's book [8] for details) to generate a 2D projection of the artist similarity graph generated from the 360 Last.fm autotags (Table 13). We then calculated the shortest path from one artist to another. The autotag values from Table 3 are shown in Figure 5 for the artist nodes in the shortest path. This is only an illustrative example and leaves many issues uninvestigated such as whether ISOMAP is the right dimensionality reduction algorithm to use. See `www.iro.umontreal.ca/~eckdoug/sitm` to listen to this example and others.

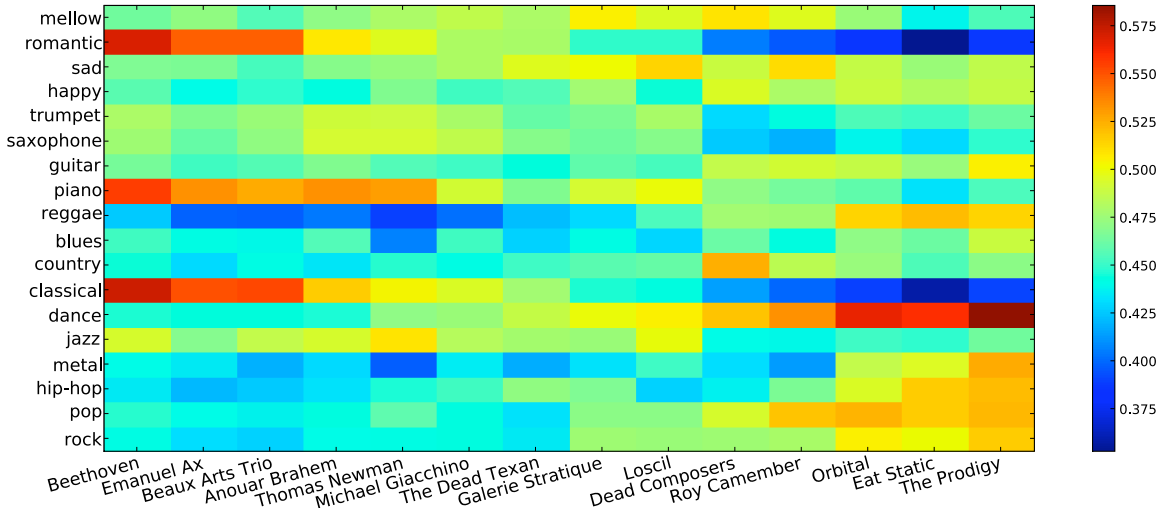

Figure 5: Shortest path between Ludwig van Beethoven and UK electronic music group The Prodigy after dimensionality reduction with ISOMAP. The similarity graph was built using all 360 Last.fm tags (Table 13), but the tags displayed are from Table 3.

10 Acknowledgement

Many thanks to the members of the CAL group, in particular Luke Barrington, Gert Lanckriet and Douglas Turnbull, for publishing the CAL500 data set and answering our numerous questions. Thanks to the many individuals that provided input, support and comments including James Bergstra, Andrew Hankinson, Stephen Green, the members of LISA lab, BRAMS lab and CIRMMT. Thanks to Joseph Turian for pointing us to the phrase "There's no data like more data." (originally from speech recognition, we believe).

References

[1] Peter Ahrendt and Anders Meng. Music genre classification using the multivariate ar feature integration model. Extended Abstract, 2005. MIREX genre classification contest (www.music-ir.org/evaluation/mirex-results).

[2] Audioscrobbler. Web Services described at http://www.audioscrobbler.net/data/webservices/.

[3] L. Barrington, D. Turnbull, D. Torres, and G. Lanckriet. Semantic similarity for music retrieval. *Music Information Retrieval Evaluation Exchange (MIREX)*, Vienna, 2007, available at http://www.music-ir.org/mirex/2007/index.php/Audio_Music_Similarity_and_Retrieval_Results.

[4] A. Berenzweig, B. Logan, D. Ellis, and B. Whitman. A large-scale evaluation of acoustic and subjective music similarity measures. In *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR 2003)*, 2003.

[5] J. Bergstra, N. Casagrande, and D. Eck. Genre classification: Timbre- and rhythm-based multiresolution audio classification. MIREX genre classification contest, 2005.

[6] J. Bergstra, N. Casagrande, D. Erhan, D. Eck, and B. Kégl. Aggregate features and AdaBoost for music classification. *Machine Learning*, 65(2-3):473–484, 2006.

[7] J. Bergstra, A. Lacoste, and D. Eck. Predicting genre labels for artists using freedb. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR 2006)*, pages 85–88, 2006.

[8] C. Bishop. *Pattern Recognition and Machine Learning*. Springer Verlag, 2006.

[9] J. K. Bradley and R. Schapire. Filterboost: Regression and classification on large datasets. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA, 2008.

[10] D. Eck, T. Bertin-Mahieux, and P. Lamere. Autotagging music using supervised machine learning. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, 2007.

[11] D. Eck, P. Lamere, T. Bertin-Mahieux, and S. Green. Automatic generation of social tags for music recommendation. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA, 2008.

[12] D. Ellis, B. Whitman, A. Berenzweig, and S. Lawrence. The quest for ground truth in musical artist similarity. In *Proceedings of the 3th International Conference on Music Information Retrieval (ISMIR 2002)*, 2002.

[13] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[14] B. Gold and N. Morgan. *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*. Wiley, Berkeley, California., 2000.

[15] M. Guy and E. Tonkin. Tidying up tags. *D-Lib Magazine*, 2006. online article: www.dlib.org/dlib/january06/guy/01guy.html.

[16] J. L. Herlocker, J. A. Konstan, and J. T. Riedl. Explaining collaborative filtering recommendations. In *Computer Supported Cooperative Work*, pages 241–250, 2000.

[17] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.

[18] Chr. Hjorth-Andersen. Chris anderson, the long tail: How endless choice is creating unlimited demand. the new economics of culture and commerce. *Journal of Cultural Economics*, 31(3):235–237, September 2007.

[19] T. Lambrou, P. Kudumakis, R. Speller, M. Sandler, , and A. Linney. Classification of audio signals using statistical features on time and wavelet tranform domains. In *Proc. Int. Conf. Acoustic, Speech, and Signal Processing (ICASSP-98)*, volume 6, pages 3621–3624, 1998.

[20] P. Lamere. Semantic tagging and music information retrieval. *Journal of New Music Research*, 2008. (to appear).

[21] E. Law, A. v. Ahn, R. Dannenberg, and M. Crawford. Tagatune: a game for music and sound annotation. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, 2007.

[22] Tao Li, Mitsunori Ogihara, and Qi Li. A comparative study on content-based music genre classification. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 282–289, New York, NY, USA, 2003. ACM Press.

[23] Beth Logan and Ariel Salomon. A music similarity function based on signal analysis. In *2001 IEEE International Conference on Multimedia and Expo (ICME'01)*, page 190, 2001.

[24] M. Mandel. blog at http://blog.mr-pc.org/2008/03/04/autotagging/.

[25] M. Mandel and D. Ellis. Song-level features and support vector machines for music classification. In T. Crawford and M. Sandler, editors, *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005)*, 2005.

[26] M. Mandel and D. Ellis. A web-based game for collecting music metadata. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, 2007.

[27] C. McKay and I. Fujinaga. Musical genre classification: is it worth pursuing and how can it be. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR 2006)*, 2006.

[28] D. Torres, D. Turnbull, L. Barrington, and G. Lanckriet. Identifying words that are musically meaningful. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, 2007.

[29] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Towards musical query-by-semantic-description using the cal500 data set. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 439–446, New York, NY, USA, 2007. ACM.

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

[31] D. Turnbull, G. Lanckriet, E. Pampalk, and M. Goto. A supervised approach for detecting boundaries in music using difference features and boosting. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, 2007.

[32] D. Turnbull, R. Liu, L. Barrington, and G. Lanckriet. A game-based approach for collecting semantic annotations of music. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, 2007.

[33] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, Jul 2002.

[34] L. von Ahn and L. Dabbish. 2004, labeling images with a computer game. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326, New York, NY, USA, 2004. ACM Press.

[35] K. West and S. Cox. Features and classifiers for the automatic classification of musical audio signals. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)*, 2004.

11 Appendix

The 360 Last.fm tags used in our experiments.

00s	drum n bass	humour	progressive rock
60s	dub	icelandic	progressive trance
70s	dutch	idm	prog rock
80s	duyster	idols	proto-punk
90s	east coast	indie	psychedelic
acid jazz	easy listening	indie folk	psychedelic rock
acoustic	ebm	indie pop	psychobilly
acoustic rock	electro	indie rock	psytrance
africa	electroclash	indietronic	punk
alt-country	electro industrial	indietronica	punk-pop
alternative	electronic	industrial	punk rock
alternative country	electronica	industrial metal	quirky
alternative hip-hop	electropop	industrial rock	r and b
alternative metal	elephant 6	instrumental	rap
alternative rock	emo	instrumental rock	rapcore
alt rock	emocore	irish	rasta
amazing	emusic	italian	reggae
ambient	england	jam	relax
american	english	jam band	relaxing
americana	epic metal	japan	rhythm and blues
anarcho-punk	ethereal	japanese	riot grrrl
anime	eurodance	japanese music	rnb
anti-christian	experimental	japanese rock	rock
art rock	experimental rock	jazz	rockabilly
asian	fantasy	jazz fusion	rock and roll
atmospheric	fav	jazz piano	rock n roll
aussie	favorite	jazz vocal	romantic
australian	favorite artists	j-pop	roots
avant-garde	favorite bands	j-rock	roots reggae
avant-garde metal	favorites	kill rock stars	russian
awesome	favourite	krautrock	sad
band	favourite artists	latin	saddle creek
baroque	favourite bands	left-wing	saxophone
beats	favourites	lesser known yet streamable artists	scandinavian
beautiful	female	lo-fi	scottish
belgian	female artists	lounge	screamo
belgium	female fronted metal	love	seattle
big beat	female vocalist	love metal	seen live
bittersweet	female vocalists	male	sexy
black metal	female vocals	male vocalists	sh*t
black music	female voices	math rock	shoegaze
bluegrass	finland	medieval	singer-songwriter
blues	finnish	meditation	ska
blues rock	finnish metal	melancholic	ska punk
bossa nova	folk	melancholy	slowcore
brand new	folk metal	mellow	sludge
brasil	folk-punk	melodic black metal	soft rock
brazil	folk rock	melodic death metal	soul
brazilian	francais	melodic hardcore	soundtrack
breakbeat	france	melodic metal	soundtracks

Continued on next page.

			Continued from previous page.
breakcore	freak folk	metal	southern rock
british	free jazz	metalcore	space rock
britpop	french	mike patton	spanish
britrock	french rock	minimal	speed metal
brutal death metal	fun	minimal techno	stoner
california	funk	motown	stoner rock
canada	funk rock	mpb	straight edge
canadian	funky	music	street punk
celtic	funny	my music	surf
chanson	fusion	neoclassical	sweden
chanson francaise	futurepop	neofolk	swedish
check out	gangsta rap	neo-soul	swedish metal
chill	garage	new age	swing
chillout	garage rock	new romantic	symphonic black metal
christian	gay	new wave	symphonic metal
christian rock	genius	new weird america	synth
classic	gentle	new york	synthpop
classical	german	ninja tune	tango
classic rock	germany	noise	technical death metal
club	girl group	noise rock	techno
cold wave	glam	norway	the beatles
comedy	glam rock	norwegian	the good stuff
composer	glasgow	nu-jazz	the worst thing ever to happen to music
composers	glitch	nu metal	thrash
contemporary classical	goa	nwobhm	thrash metal
cool	good	officially sh*t	traditional
country	good music	oi	trance
crap	goth	oldies	trip-hop
crossover	gothic	old school	trumpet
dance	gothic metal	ost	turntablism
dancehall	gothic rock	pagan metal	twee
danish	goth rock	piano	uk
dark	great lyricists	piano rock	underground hip hop
dark ambient	grind	polish	underground rap
dark electro	grindcore	political	uplifting trance
dark metal	grunge	polskie	urban
darkwave	guilty pleasures	pop	us
death	guitar	pop punk	usa
death metal	guitar virtuoso	pop rock	video game music
deathrock	hair metal	post-grunge	viking metal
deutsch	happy	post-hardcore	visual kei
disco	hardcore	post-punk	vocal
dnb	hardcore punk	post-rock	vocal jazz
doom	hard rock	power metal	vocal trance
doom metal	heavy	power pop	warp
downtempo	heavy metal	prog	world
dream pop	hip-hop	progressive	world music
drone	horror punk	progressive death metal	wristslitters
drum and bass	house	progressive metal	

Table 13: The 360 Last.fm tags used in our experiments

Category	$A/\|V\|$	Model	Precision		Recall		F-Score
Emotion	$4/36$	Random	0.276	(0.012)	0.113	(0.004)	0.160
		UpperBnd	0.957	(0.005)	0.396	(0.010)	0.560
		GMM	0.424	(0.008)	**0.195**	(0.004)	0.267
		MFCC delta	0.444	(0.025)	0.192	(0.016)	**0.268**
		afeats	0.433	(0.03)	0.171	(0.011)	0.245
		afeats exp.	**0.449**	(0.026)	0.176	(0.011)	0.253
		bfeats	0.418	(0.053)	0.156	(0.037)	0.227
Genre	$2/31$	Random	0.055	(0.005)	0.079	(0.008)	0.065
		UpperBnd	0.562	(0.026)	0.777	(0.018)	0.652
		GMM	0.171	(0.009)	**0.242**	(0.019)	0.200
		MFCC delta	0.154	(0.024)	0.168	(0.021)	0.161
		afeats	0.173	(0.048)	0.134	(0.033)	0.151
		afeats exp.	**0.236**	(0.047)	0.234	(0.016)	**0.235**
		bfeats	0.147	(0.027)	0.160	(0.045)	0.153
Instrumentation	$4/24$	Random	0.141	(0.009)	0.195	(0.014)	0.164
		UpperBnd	0.601	(0.015)	0.868	(0.018)	0.710
		GMM	0.267	(0.008)	0.320	(0.022)	0.291
		MFCC delta	0.267	(0.047)	**0.363**	(0.021)	0.308
		afeats	0.294	(0.073)	0.275	(0.074)	0.284
		afeats exp.	0.276	(0.044)	0.350	(0.033)	**0.309**
		bfeats	**0.329**	(0.065)	0.289	(0.084)	0.308
Solo	$1/9$	Random	0.031	(0.007)	0.155	(0.035)	0.052
		UpperBnd	0.197	(0.019)	0.760	(0.052)	0.313
		GMM	**0.060**	(0.012)	0.261	(0.050)	**0.098**
		MFCC delta	0.054	(0.002)	0.374	(0.035)	0.094
		afeats	0.045	(0.002)	0.278	(0.078)	0.078
		afeats exp.	0.056	(0.001)	**0.396**	(0.017)	**0.098**
		bfeats	0.042	(0.002)	0.312	(0.094)	0.074
Usage	$2/15$	Random	0.073	(0.008)	0.154	(0.016)	0.099
		UpperBnd	0.363	(0.014)	0.814	(0.031)	0.502
		GMM	**0.122**	(0.012)	**0.264**	(0.027)	**0.167**
		MFCC delta	**0.122**	(0.011)	0.239	(0.028)	0.162
		afeats	0.103	(0.010)	0.188	(0.054)	0.133
		afeats exp.	0.118	(0.007)	0.237	(0.015)	0.157
		bfeats	0.106	(0.010)	0.209	(0.066)	0.140
Vocal	$2/16$	Random	0.062	(0.007)	0.153	(0.018)	0.088
		UpperBnd	0.321	(0.017)	0.788	(0.019)	0.456
		GMM	**0.134**	(0.005)	**0.335**	(0.021)	**0.191**
		MFCC delta	0.116	(0.011)	0.252	(0.029)	0.159
		afeats	0.130	(0.030)	0.198	(0.050)	0.157
		afeats exp.	0.108	(0.009)	0.228	(0.019)	0.147
		bfeats	0.133	(0.017)	0.212	(0.046)	0.164

Table 14: Music annotation results for specific categories. A = Annotation length, $|V|$ = Vocabulary size. Numbers in parentheses are the variance over all tags in the category. GMM is the results of the Gaussian mixture model of the CAL group. MFCC delta, afeats, afeats extended and bfeat are the results of our boosting model with each of these features. Random, UpperBnd and GMM results taken from [30]. This is a continuation of Table 7.

Category	$\|V\|$	Model	MeanAP		MeanAROC	
Emotion	36	Random	0.327	(0.006)	0.504	(0.003)
		GMM	0.506	(0.008)	**0.710**	(0.004)
		MFCC delta	0.503	(0.031)	0.702	(0.005)
		afeats	0.469	(0.026)	0.652	(0.005)
		afeats exp.	0.478	(0.023)	0.655	(0.006)
		bfeats	**0.565**	(0.048)	0.686	(0.006)
Genre	31	Random	0.132	(0.005)	0.500	(0.005)
		GMM	**0.329**	(0.012)	0.719	(0.005)
		MFCC delta	0.094	(0.013)	0.705	(0.013)
		afeats	0.088	(0.011)	0.626	(0.010)
		afeats exp.	0.117	(0.036)	**0.720**	(0.011)
		bfeats	0.118	(0.032)	0.693	(0.015)
Instrumentation	24	Random	0.221	(0.007)	0.502	(0.004)
		GMM	**0.399**	(0.018)	0.719	(0.006)
		MFCC delta	0.137	(0.022)	0.707	(0.005)
		afeats	0.182	(0.033)	0.658	(0.015)
		afeats exp.	0.173	(0.03)	0.705	(0.006)
		bfeats	0.140	(0.032)	**0.720**	(0.015)
Solo	9	Random	0.106	(0.014)	0.502	(0.004)
		GMM	**0.180**	(0.028)	**0.712**	(0.006)
		MFCC delta	0.052	(0.002)	0.565	(0.025)
		afeats	0.050	(0.002)	0.582	(0.009)
		afeats exp.	0.051	(0.002)	0.650	(0.010)
		bfeats	0.042	(0.003)	0.519	(0.026)
Usage	15	Random	0.169	(0.012)	0.501	(0.005)
		GMM	**0.240**	(0.016)	**0.707**	(0.004)
		MFCC delta	0.12	(0.009)	0.621	(0.022)
		afeats	0.133	(0.022)	0.538	(0.016)
		afeats exp.	0.127	(0.007)	0.637	(0.008)
		bfeats	0.101	(0.014)	0.563	(0.034)
Vocal	16	Random	0.137	(0.006)	0.502	(0.004)
		GMM	**0.260**	(0.018)	**0.705**	(0.005)
		MFCC delta	0.111	(0.012)	0.652	(0.018)
		afeats	0.090	(0.007)	0.586	(0.014)
		afeats exp.	0.105	(0.012)	0.628	(0.009)
		bfeats	0.128	(0.018)	0.630	(0.020)

Table 15: Retrieval results. $|V|$ = Vocabulary size. Numbers in parentheses are the variance over all tags in the category. GMM is the results of the Gaussian mixture model of the CAL group. MFCC delta, afeats, afeats extended and bfeat are the results of our boosting model with each of these features. Random and GMM taken from [30]. This is a continuation of Table 8.