LEARNING SIMILARITY FROM COLLABORATIVE FILTERS

Brian McFee **Luke Barrington*** **Gert Lanckriet***
Computer Science and Engineering *Electrical and Computer Engineering
University of California, San Diego
bmcfee@cs.ucsd.edu lukeinusa@gmail.com gert@ece.ucsd.edu

ABSTRACT

Collaborative filtering methods (CF) exploit the wisdom of crowds to capture deeply structured similarities in musical objects, such as songs, artists or albums. When CF is available, it frequently outperforms content-based methods in recommendation tasks. However, songs in the so-called "long tail" cannot reap the benefits of collaborative filtering, and practitioners must rely on content-based methods. We propose a method for improving content-based recommendation in the long tail by learning an optimized similarity function from a sample of collaborative filtering data. Our experimental results demonstrate substantial improvements in accuracy by learning optimal similarity functions.

1. INTRODUCTION

"Collaborative filtering" (CF) is a popular method for multimedia recommendation applications in which data (*e.g.*, songs, artists, books or movies) are represented and compared in terms of the people who use them. Systems based on collaborative filtering exploit the "wisdom of crowds" to define similarity between items, which can then be used for recommendation. Indeed, collaborative filtering systems benefit from several attractive properties: CF explicitly represents individual users, and is therefore inherently personalized; data collection can be done passively, rather than requiring users to actively tag items; and CF data directly captures usage habits: exactly the quantity that recommendation engines strive to affect.

It is therefore not surprising that CF methods have become an active research topic in recent years, due in no small part to the recently concluded competition for the Netflix Prize [1]. Within the Music Information Retrieval (MIR) community, recent studies have shown that CF systems consistently outperform content-based methods for playlist generation [6] and tag prediction [15]. However, collaborative filtering suffers from the dreaded "cold start" problem: CF methods fail on items which have not yet been used, and are therefore unsuitable for recommendation in the "long tail". While this problem persists for all media (*e.g.*, movies, books, etc.), it is especially deadly in music, due to the relatively large number of unknown songs and artists in the world today. Netflix boasts 100,000 DVD titles [1], while Apple's iTunes store provides access to over 13 million songs [2].

Motivated by the cold-start problem, MIR researchers have worked steadily to improve *content-based* recommendation engines. Content-based systems operate solely on feature representations of music, eliminating the need for human intervention. While this approach naturally extends to long-tail data, the definition of *similarity* in these systems is frequently ad-hoc and not explicitly optimized for the specific task. As a result, it remains unclear if, or to what extent, content-based systems can capture relevant similarity information expressed by collaborative filtering.

In this paper, we pose the question: can we *learn* content-based similarity from a collaborative filter? Empirically, CF data provides a highly reliable means for determining similarity between musical objects. Our main contribution in this paper is a method for optimizing content-based similarity by learning from a collaborative filter.

The proposed method treats similarity learning as an information retrieval problem, where similarity is evaluated according to the ranked list of results in response to a query example, *e.g.*, a list of artists ordered by similarity to "The Beatles". Optimizing similarity for ranking requires more sophisticated machinery than is used in other methods, *e.g.*, genre classifiers. However, it does offer a few key advantages, which we believe are crucial for realistic music applications. First, there are no assumptions of transitivity or symmetry in the proposed method. As a result, "The Beatles" may be considered a relevant result for "Oasis", and not vice versa; this is not possible with other methods in the literature, *e.g.*, the embedding technique described in [21]. Second, CF data can be collected *passively* from users (*e.g.*, via scrobbles [16]) and directly captures their listening habits. Finally, optimizing similarity for ranking directly attacks the main quantity of interest, *i.e.*, the ordered list of retrieved items, rather than potentially irrelevant or overly coarse abstractions (*e.g.*, genre).

Our proposed method is quite general, and can improve similarities derived from semantic descriptions provided by humans or an auto-tagging engine. As we will demonstrate, even hand-crafted song annotations can be optimized to more accurately reflect and predict the similarity structure encoded by collaborative filtering data.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page.
© 2010 International Society for Music Information Retrieval.

1.1 Related work

A significant amount of research has been devoted to the topic of musical similarity in the past decade. Ellis, et al. [9] evaluated similarity metrics derived from various data sources against human survey data. Similarly, Kim, et al. [15] evaluate several sources of artist similarity for a tag prediction task, and observe that methods based on collaborative filtering significantly out-perform acoustic or semantic similarity. However, neither of these works attempt to optimize similarity for a specific task.

Slaney, et al. [21] apply several learning algorithms to find similarity metrics over acoustic features which are optimized to cluster songs of the same artist, album, or that appear on the same blog. Our previous work [19] applies similar techniques to predict human survey data and optimally integrate multiple data sources. The method proposed here falls squarely within this line of work, but differs in that the metric is trained from collaborative filtering, and optimized for ranking performance, rather than classification or (comparatively scarce) human survey data.

There is a large body of work which treats collaborative filtering as a *matrix completion* problem (see, *e.g.*, [24]). In the matrix completion view, the goal is to perform user-centric recommendation by filling in missing entries of the users-by-content matrix, *i.e.*, recommending content to a user based on his or her specific preferences. Our application here is slightly different: rather than trying to complete the matrix, we interpret the collaborative filtering matrix as the ground truth, from which, similarity can be derived. Our goal is to train a content-based system to match similarities derived from CF data. We also stress that our proposed method is *not* a hybrid method: once the metric has been trained, collaborative filtering data is not necessary to compute similarities for unseen, long-tail songs.

2. LEARNING SIMILARITY

Our goal is to learn an optimal similarity function for songs, and as such, we must choose a family of similarity functions over which to optimize. Many families of similarity functions have been proposed in the MIR literature, such as distance between generative models of acoustic [3, 17] or semantic [5] descriptors, and playlist-based similarity [18].

Here, we opt for Euclidean distance between song representations. The primary reason for this choice is that Euclidean distance naturally lends itself to optimization by *metric learning* (see, *e.g.*, [19, 21]). In metric learning, each data point is described by a vector in $\mathbb{R}^d$, and the goal is to learn a linear projection matrix L such that distances after projection ($\|Li - Lj\|$) are small for "similar" pairs (i, j) and large for "dissimilar" pairs. Due to computational issues, optimization is performed not on L, but on a positive semi-definite [1] (PSD) matrix $W = L^\mathsf{T}L \succeq 0$. In the metric defined by W, distance between points (i, j) after projection is denoted by the quadratic form

$$d(i,j) = \|i-j\|_W^2 = (i-j)^\mathsf{T}W(i-j)$$
$$= (i-j)^\mathsf{T}L^\mathsf{T}L(i-j) = \|Li - Lj\|^2. \quad (1)$$

For the present application, we apply the *Metric Learning to Rank* (MLR) algorithm [20]. Here, we provide a brief overview of the algorithm.

[1] A positive semi-definite matrix W, denoted $W \succeq 0$ is square, symmetric, and has non-negative eigenvalues.

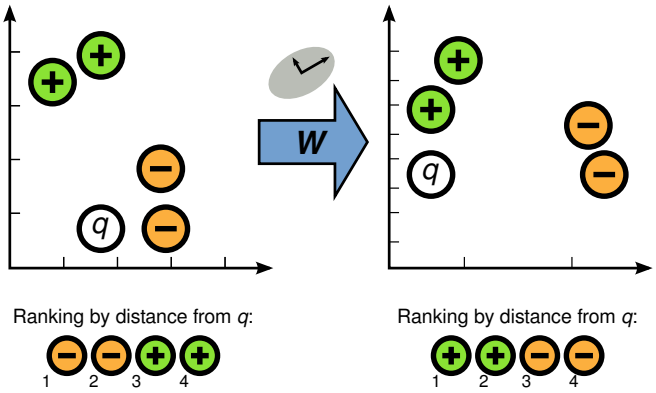

Figure 1. Metric Learning to Rank (MLR) learns a metric (W) so that a query song q is close to relevant results (+) and far from irrelevant results (-). Optimization is performed with respect to the rankings induced by distance from the query.

2.1 Metric learning to rank

Metric Learning to Rank (MLR) [20] is an extension of Structural SVM [13]. Structural SVM has been demonstrated to be an effective method for solving ranking problems in information retrieval systems [8], and the MLR algorithm extends the methodology to the *query-by-example* setting by learning a metric space, rather than a discriminant vector. Specifically, MLR learns a positive semi-definite matrix W such that rankings induced by learned distances are optimized according to a ranking loss measure, *e.g.*, ROC area (AUC) or precision-at-k. In this setting, "relevant" results should lie close in space to the query q, and "irrelevant" results should be pushed far away.

For a query song q, a natural ordering of the database $\mathcal{X}$ is obtained by sorting $x \in \mathcal{X}$ according to increasing distance from q under the metric defined by W (see Figure 1). The metric W is learned by solving a constrained optimization problem such that, for each training query q, a higher score is assigned to the "true" ranking y_q^* than to any other ranking $y \in \mathcal{Y}$ (the set of all rankings):

$$\langle W, \psi(q, y_q^*)\rangle \geq \langle W, \psi(q,y)\rangle + \Delta(y_q^*, y) - \xi_q. \quad (2)$$

Here, the "score" for a query-ranking pair (q, y) is computed by the Frobenius inner product:

$$\langle W, \psi(q,y)\rangle = \mathrm{tr}(W\psi(q,y)). \quad (3)$$

$\psi(q, y)$ is a matrix-valued feature map which encodes the query-ranking pair (q, y), and $\Delta(y_q^*, y)$ computes the loss incurred by predicting y instead of y_q^* for the query q (*e.g.*,

Algorithm 1 Metric Learning to Rank [20]

Input: data $\mathcal{X} = \{q_1, q_2, \ldots, q_n\} \subset \mathbb{R}^d$,
true rankings $y_1^*, y_2^*, \ldots y_n^*$,
slack trade-off $C \geq 0$

Output: $d \times d$ matrix $W \succeq 0$

$$\min_{W \succeq 0, \xi} \quad \mathrm{tr}(W) + C \cdot \frac{1}{n} \sum_{q \in \mathcal{X}} \xi_q$$

$$\text{s.t.} \quad \forall q \in \mathcal{X},\ \forall y \in \mathcal{Y} \setminus \{y_q^*\} :$$
$$\langle W, \psi(q, y_q^*) \rangle \geq \langle W, \psi(q, y) \rangle + \Delta(y_q^*, y) - \xi_q$$
$$\xi_q \geq 0$$

loss in AUC score), essentially playing the role of the "margin" between rankings y_q^* and y. Intuitively, the score for a true ranking y_q^* should exceed the score for any other y by at least the loss $\Delta(y_q^*, y)$. (In the present context, the "true" ranking is any one which places all relevant results before all irrelevant results.) To allow violations of margins during training, a slack variable $\xi_q \geq 0$ is introduced for each query.

MLR encodes query-ranking pairs (q, y) by the *partial order* feature [13]:

$$\psi(q, y) = \sum_{i \in \mathcal{X}_q^+} \sum_{j \in \mathcal{X}_q^-} y_{ij} \left(\frac{\phi(q, i) - \phi(q, j)}{|\mathcal{X}_q^+| \cdot |\mathcal{X}_q^-|} \right), \quad (4)$$

where $\mathcal{X}_q^+$ (resp. $\mathcal{X}_q^-$) is the set of relevant (resp. irrelevant) songs for q, the ranking y is encoded by

$$y_{ij} = \begin{cases} +1 & i \text{ before } j \text{ in } y \\ -1 & i \text{ after } j \end{cases},$$

and

$$\phi(q, i) = -(q - i)(q - i)^\mathsf{T} \quad (5)$$

captures the affinity between the query q and a single item i. Intuitively, ψ is constructed by adding the difference $\phi(q, i) - \phi(q, j)$ whenever y places (relevant) i before (irrelevant) j and subtracted otherwise. This choice of ψ therefore emphasizes directions in the feature space which are correlated with good rankings.

For a test query q', the predicted ranking y is that which achieves the highest score by W, *i.e.*, $\mathrm{argmax}_y \langle W, \psi(q', y) \rangle$. This can be found efficiently by sorting the corpus in descending order of $\langle W, \phi(q', x) \rangle$. Equation 5 defines ϕ so that when taking the inner product with W,

$$\begin{aligned} \langle W, \phi(q', x) \rangle &= -\mathrm{tr}\left(W(q' - x)(q' - x)^\mathsf{T} \right) \quad (6) \\ &= -(q' - x)^\mathsf{T} W (q' - x) = -\|q' - x\|_W^2, \end{aligned}$$

the result is the (negative, squared) distance between q' and x under the metric defined by W. Thus, decreasing $\langle W, \phi(q', x) \rangle$ corresponds to increasing distance from q'.

The MLR optimization problem is listed as Algorithm 1. As in support vector machines, the objective consists of two competing terms: $\mathrm{tr}(W)$ is a convex approximation to the rank of the learned metric, and $1/n \sum \xi_q$ measures the empirical (hinge) loss on the training set, and the two terms are balanced by a trade-off parameter C. Although the full problem includes a super-exponential number of constraints (one for each $y \in \mathcal{Y}$, for each q), [20] describes an efficient approximation algorithm based on cutting planes [14] which works well in practice.

3. DATA

Since our goal is to learn a content-based similarity metric for songs, it would seem logical to derive similarity from CF data relating users to songs. However, in practice, such matrices tend to exhibit high sparsity, which would lead to unstable similarity computations. We instead opt to derive similarity at the *artist* level, and then transfer similarity to the song level. Given a set of artists, and a collaborative filtering matrix over the artists, our experimental procedure is as follows:

1. extract *artist similarity* from the CF data,
2. transfer artist similarity to *song similarity*,
3. construct a feature representation for each song,
4. learn a metric W over song representations to predict *song similarities*, and
5. evaluate W by testing retrieval of similar songs in response to (unseen) test songs.

Steps 1 and 2 are described in Section 3.2, and step 3 is described throughout Section 3.3. Next, we describe the sources of our audio and collaborative filtering data.

3.1 Swat10k

Our experiments use the *Swat10k* dataset of 10,870 songs from 3,748 unique artists [22]. Each song has been weakly-labeled from a vocabulary of 1,053 tags from Pandora's Music Genome Project [2] that include multiple genres and acoustically objective descriptors.

3.2 Collaborative filtering

To define similarity between songs, we use the collaborative filtering (CF) data mined from Last.fm [3] by [7]. The raw collaborative filtering matrix consists of approximately 17.5 million user-song interactions over 359K users and 186K artists with MusicBrainz [4] identifiers (MBIDs).

We first filtered the CF matrix down to include only the Swat10k artists by matching MBIDs, resulting in a reduced CF matrix F:

$$F_{ui} = \begin{cases} 1 & \text{user } u \text{ listened to artist } i \\ 0 & \text{otherwise,} \end{cases} \quad (7)$$

of 356,026 users and 3,748 artists.

From the CF matrix, we define the similarity between artists i and j as the cosine-similarity between the column-vectors F_i and F_j:

$$S_{ij} = \frac{F_i^\mathsf{T} F_j}{\|F_i\| \cdot \|F_j\|}. \quad (8)$$

[2] http://www.pandora.com/mgp.shtml
[3] http://last.fm
[4] http://www.musicbrainz.org/

	Training	Validation	Test	Discard
# Artists	746	700	700	1602
# Songs	1842	1819	1862	5347
# Relevant	39.5	37.7	36.4	

Table 1. Statistics of the Swat10k data. "# Relevant" is the average size of the relevant set for each song.

Intuitively, S_{ij} counts the number of users shared between artists i and j, and normalizes by popularity.

To ensure stable similarity measurements, we discarded all artists from the set which had fewer than 100 users. This leaves 2,146 artists, which we split roughly into thirds for training, validation, and test sets. For each artist, we then define the set of "relevant" artists as the 10 closest training artists according to Equation 8 [5] .

Finally, we convert artist-level relevance to song-level relevance. For each song of an artist a, the relevant set is the union of the sets of songs from each of a's relevant artists. Table 1 summarizes the statistics of the data used in our experiments.

3.3 Features

For each song in our database, we construct three different feature representations: acoustic, auto-tags, and tags provided by human labelers.

3.3.1 Vector quantized MFCCs

Our representation of acoustic features is based upon vector-quantized Mel-Frequency Cepstral Coefficients (MFCCs), and consists of a 3-step process: feature extraction, vector quantization, and kernelization. The method described here is similar to that of [12], and is inspired by similar methods found in the computer vision literature [10].

First, for each song, we extract the first 13 MFCCs from 25ms half-overlapping windows. Each MFCC vector is augmented by appending the first and second instantaneous time derivatives, resulting in a sequence of 39-dimensional delta-MFCC (ΔMFCC) vectors for each song.

Using the songs which were discarded due to insufficient collaborative filtering data, we trained a codebook for use as a vector quantizer. We randomly selected 1000 songs from the discard set, and from each selected song, randomly sampled 1000 ΔMFCC vectors, for a total of 1 million codebook-training vectors. Each training vector v was z-scored, so that the i^{th} coordinate v_i becomes

$$v_i \mapsto \frac{v_i - \mu_i}{\sigma_i}, \tag{9}$$

where (μ_i, σ_i) are the sample mean and standard deviation of the i^{th} coordinate in the codebook-training set. We ran k-means with $k = 5000$ on the z-scored training vectors, using the implementation provided by [11]. The result is a set of 5000 codewords, each of which was subsequently "un"-z-scored by

$$v_i \mapsto \sigma_i v_i + \mu_i. \tag{10}$$

[5] For training artists, we assume self-similarity, so there are technically 11 relevant artists for each training artist.

With the codebook in hand, the ΔMFCC vectors for each song in the training, validation, and test splits were quantized by finding the closest (in Euclidean distance) codeword. Each song was summarized by a histogram over the 5000 codewords, corresponding to the frequency with which a codeword was selected as a quantizer in that song.

Finally, we constructed a χ^2-kernel over songs, so that the similarity between two codeword histograms u and v is calculated as [6]

$$k(u, v) = \exp\left(-\chi^2(u, v)\right) \tag{11}$$

$$\chi^2(u, v) = \sum_{i=1}^{5000} \frac{(u_i - v_i)^2}{u_i + v_i}. \tag{12}$$

(This kernel can itself be viewed as a soft vector quantizer, this time operating at the song-level rather than the feature-level.) Each song is represented by a vector in $\mathbb{R}^{1842}$, where the i^{th} dimension represents similarity to the i^{th} training song. We then compress these vectors by principal components analysis to 35 dimensions, which capture 95% of the variance in the training set.

3.3.2 Auto-tags

We can alternatively represent a song's acoustic information by using descriptive *semantics*. By learning from example songs that humans have labeled with tags, an "auto-tagger" (*e.g.*, [12, 23]) can automatically rate the relevance of these tags to new, unlabeled songs. The resulting "auto-tags" offer a concise description of the song, and semantic similarity between auto-tags has been shown to improve on content-based similarity derived from acoustics alone [5]. We use the auto-tagger described in [23] to label each song with a real-valued vector of 149 auto-tags: the i^{th} dimension of this vector corresponds to the probability that the i^{th} tag applies to the song, given the observed ΔMFCCs.

3.3.3 Human tags

Our third feature describes songs with "human tags" mined from the Music Genome Project by [22] that include descriptors of a song's genre, style, instrumentation, vocals and lyrics. Each song is represented by a 1,053-dimensional binary vector that is "weakly-labeled", meaning that a "1" implies that the tag is relevant but a "0" does not guarantee that the tag does not apply to the song. We consider these "human tags" to be "acoustically objective" as they have been applied by musicological experts and refer only to the acoustic content of the songs. They represent the ideal output that a content-based auto-tagger might achieve.

4. EXPERIMENTS

The MLR algorithm requires that a few parameters be set when training: not only the slack trade-off C, but also the choice of ranking measure to optimize. The implementation described in [20] supports several standard ranking measures: the area under the ROC curve (AUC), mean

[6] For the summation in Equation 12, we adopt the convention $0/0 = 0$.

Data source	AUC	MAP	MRR
MFCC	0.630	0.057	0.249
Optimized MFCC	0.719	0.081	0.275
Auto-tags	0.726	0.090	0.330
Optimized auto-tags	0.776	0.116	0.327
Human tags	0.770	0.187	0.540
Optimized human tags	0.939	0.420	0.636

Table 2. Ranking performance of each data source (MFCC, auto-tags, and human tags), before and after learning with MLR.

reciprocal rank (MRR), mean average precision (MAP), precision-at-k (P@k), and normalized discounted cumulative gain (NDCG); see [8] for a brief summary of these ranking measures. For P@k and NDCG, an additional parameter k must be set, which defines how many songs should be retrieved when evaluating the ranking.

For each data source described in Section 3, we trained metrics with all five variants of MLR. We swept over $C \in \{10^{-2}, 10^{-1}, \ldots, 10^{11}\}$, and for the P@k and NDCG variants, we also swept over $k \in \{2, 4, 8, \ldots, 256\}$. Performance was evaluated on the validation set, and the best-performing metric was then tested on the test set.

4.1 Embedding results

After learning W, we evaluate on the validation and test sets by computing for each query song q, the ranked list of training songs x ordered by increasing $\|q - x\|_W$. The resulting rankings are scored, and scores are averaged over all q to produce a single score for the learned metric. For comparison purposes, we also evaluate rankings derived from *native* metrics (*i.e.*, without learning W). The native metric for auto-tags is taken to be the Kullback-Leibler divergence between auto-tag distributions. For MFCC and human tags, we use standard Euclidean distance.

Table 4 displays some example playlists generated by the native and optimized MFCC spaces. At a high level, the learned metrics successfully de-noise the feature space to generate more cohesive playlists. Table 2 lists ranking performance for each data source, before and after optimization. In all but one case (auto-tag MRR), performance improves across all evaluation criteria. For each data source (MFCC, auto-tags, and human tags), we observe dramatic improvements in accuracy over the corresponding native similarity metric.

Quantitatively, the purely acoustic model improves in AUC score from 0.630 to 0.719. The optimized similarity performs comparably to native auto-tag similarity, but can be constructed entirely from passive data (as opposed to the actively collected data necessary for building auto-tag models). Similarly, optimizing auto-tags improves AUC from 0.726 to 0.776, which is comparable to the native performance of human tags. Finally, optimizing human tags improves AUC substantially, from 0.770 to 0.939. This indicates that even when annotations are hand-crafted by experts, recommendation may still be greatly improved by using an appropriate model of the tag vocabulary.

Top tags		Bottom tags	
1.	LATIN	1044.	TWO-STEP STYLE
2.	A REGGAE FEEL	1045.	UNUSUAL VOCAL SOUNDS
3.	REGGAE	1046.	UPBEAT LYRICS
4.	CHRISTIAN	1047.	CALL-AND-RESPONSE VOCALS
5.	NEW-AGE	1048.	ELECTRIC PIANOS
6.	ROCK ON THE RANGE RADIO	1049.	MODAL HARMONIES
7.	WAKARUSA RADIO	1050.	TONAL HARMONIES
8.	SASQUATCH RADIO	1051.	VOCAL COUNTERPOINT
9.	CMJ MUSIC MARATHON	1052.	VOCAL SAMPLES
10.	REGGAE / CARIBBEAN	1053.	WESTERN SWING

Table 3. The top and bottom 10 tags learned by MLR, ordered by weight. 85 tags receive 0 weight.

4.2 Learning tag weights

Given the substantial improvement observed by optimizing human tags, one may wonder what conclusions can be drawn from the learned metric. In particular, since W can be interpreted as "translation matrix" or vocabulary model, it is natural to ask which tags define the similarity space, and which tags are redundant or non-informative.

Because W contains both positive and negative entries, it is not immediately clear how to interpret a full matrix W in terms of tags. However, placing further restrictions on the form of W can ease interpretability (at the expense of model flexibility). We repeated the "human tags" experiment with a modification of Algorithm 1 that restricts W to be diagonal and non-negative. In the restricted model, the i^{th} element of the diagonal W_{ii} can be interpreted as a weight for the i^{th} tag. The diagonal metric achieves AUC of 0.875 (compared to 0.776 native and 0.939 for a full W).

Table 3 lists the top and bottom 10 tags, ordered by weights W_{ii}. Several interesting observations can be made here: all of the top tags refer either to genre (*e.g.*, LATIN, REGGAE) or streaming radio identifiers (*e.g.*, WAKARUSA, CMJ). This corroborates previous studies which indicate that grouping music by social cues, such as radio playlists or blogs, can assist recommendation [4]. By contrast, the bottom tags are primarily musicological terms (*e.g.*, VOCAL COUNTERPOINT) which apparently convey little useful information for recommendation.

This view of MLR as a supervised learning procedure for vocabulary models suggests comparison to standard, unsupervised techniques, such as TF-IDF with cosine similarity. It turns out that for this data set, using TF-IDF weighting results a *decrease* in AUC from 0.770 to 0.724! From this, we can conclude that it is suboptimal to rely on the natural statistics of tags to define similarity.

5. CONCLUSION

We have proposed a method for improving content-based similarity by learning from a sample of collaborative filtering data. The proposed method learns an optimal transformation of features to reproduce high-quality CF similarity, and can be used to improve the quality of recommendation in the long tail. If songs are described by semantic tags, our method reveals which tags play the most important role in defining an optimal similarity metric. By revealing the most important tags for predicting CF similarity, our method may also be useful for guiding the development of discovery interfaces and automatic tagging algorithms.

	Query song	Native space playlist	Optimized space playlist
MFCC	Chick Corea Elektric Band - Beneath the Mask	Katalyst - Break Up	► Michael Brecker - Two Blocks From The Edge
		Stevie Wonder - Superstition	► Charlie Parker - Wee
		James Brown - Soul Power	Coleman Hawkins - There Is No Greater Love
		Tina Turner - What's Love Got To Do With It	► Miles Davis - Walkin'
		The Whispers - And The Beat Goes On	Clifford Brown - Love Is A Many Splendored Thing
Auto-tags	White Zombie - Electric Head (Pt. 2.) (Remix)	► Sepultura - Apes Of God	► Sepultura - Apes Of God
		Arctic Monkeys - A Certain Romance	► Metallica - Nothing Else Matters (Live)
		Secret Machines - Lightning Blue Eyes	Secret Machines - Lightning Blue Eyes
		Green Day - Longview (Live)	The Warlocks - Gypsy Nightmare
		Perry Farrell - Kinky	Mastodon - Crystal Skull
Human tags	Aaliyah - Miss You	► Ginuwine - In Those Jeans	► Monica - Don't Take It Personal
		► Monica - Don't Take It Personal	► Ginuwine - In Those Jeans
		► Ashanti - Foolish	► Ashanti - Foolish
		Foo Fighters - DOA	► Ne-Yo - Go On Girl
		Say Hi To Your Mom - Northwestern Girls	Jodeci - Freak N You

Table 4. Example playlists in native and optimized MFCC, auto-tag, and human tag spaces. Playlists are generated by finding the five nearest neighbors of the query; relevant results are indicated by ►.

6. REFERENCES

[1] Netflix press release, 2009. http://netflix.mediaroom.com/index.php?s=43&item=307.

[2] Apple itunes, 2010. http://www.apple.com/itunes.

[3] Jean-Julien Aucouturier and François Pachet. Music similarity measures: What's the use? In *Inernational Symposium on Music Information Retrieval (ISMIR2002)*, pages 157–163, 2002.

[4] Claudio Baccigalupo, Justin Donaldson, and Enric Plaza. Uncovering affinity of artists to multiple genres from social behaviour data. In *International Symposium on Music Information Retrieval (ISMIR2008)*, September 2008.

[5] Luke Barrington, Antoni Chan, Douglas Turnbull, and Gert Lanckriet. Audio information retrieval using semantic similarity. *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2007.

[6] Luke Barrington, Reid Oda, and Gert Lanckriet. Smarter than genius? Human evaluation of music recommender systems. In *Proceedings of the 10th International Conference on Music Information Retrieval*, 2009.

[7] O. Celma. *Music Recommendation and Discovery in the Long Tail*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2008.

[8] Soumen Chakrabarti, Rajiv Khanna, Uma Sawant, and Chiru Bhattacharyya. Structured learning for non-smooth ranking losses. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 88–96, New York, NY, USA, 2008. ACM.

[9] D. Ellis, B. Whitman, A. Berenzweig, and S. Lawrence. The quest for ground truth in musical artist similarity. In *Proeedings of the International Symposium on Music Information Retrieval (ISMIR)*, pages 170–177, October 2002.

[10] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, volume 2, 2005.

[11] Peter Gehler. MPIKmeans, 2007. http://mloss.org/software/view/48/.

[12] M. Hoffman, D. Blei, and P. Cook. Easy as CBA: A simple probabilistic model for tagging music. In *Proceedings of the 10th International Conference on Music Information Retrieval*, 2009.

[13] Thorsten Joachims. A support vector method for multivariate performance measures. In *Proceedings of the 22nd international conference on Machine learning*, pages 377–384, New York, NY, USA, 2005. ACM.

[14] Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural svms. *Mach. Learn.*, 77(1):27–59, 2009.

[15] Joon Hee Kim, Brian Tomasik, and Douglas Turnbull. Using artist similarity to propagate semantic information. In *Proceedings of the 10th International Conference on Music Information Retrieval*, 2009.

[16] Last.FM, January 2009. http://www.last.fm/.

[17] B. Logan. Music recommendation from song sets. In *International Symposium on Music Information Retrieval (ISMIR2004)*, 2004.

[18] François Maillet, Douglas Eck, Guillaume Desjardins, and Paul Lamere. Steerable playlist generation by learning song similarity from radio station playlists. In *Proceedings of the 10th International Conference on Music Information Retrieval*, 2009.

[19] Brian McFee and Gert Lanckriet. Heterogeneous embeddding for subjective artist similarity. In *Proceedings of the 10th International Conference on Music Information Retrieval*, 2009.

[20] Brian McFee and Gert Lanckriet. Metric learning to rank. In *Proceedings of the 27th annual International Conference on Machine Learning (ICML)*, 2010.

[21] M. Slaney, K. Weinberger, and W. White. Learning a metric for music similarity. In *International Symposium on Music Information Retrieval (ISMIR2008)*, pages 313–318, September 2008.

[22] D. Tingle, Y. Kim, and D. Turnbull. Exploring automatic music annotation with "acoustically-objective" tags. In *IEEE International Conference on Multimedia Information Retrieval (MIR)*, 2010.

[23] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE TASLP*, 16(2):467–476, Feb. 2008.

[24] K. Yoshii, M. Goto, K. Komatani, T. Ogata, and H.G. Okuno. An efficient hybrid music recommender system using an incrementally trainable probabilistic generative model. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):435–447, 2008.