LEARNING A MUSIC SIMILARITY MEASURE ON AUTOMATIC ANNOTATIONS WITH APPLICATION TO PLAYLIST GENERATION

Linxing Xiao[1,2], *Lie Lu*[1], *Frank Seide*[1], *and Jie Zhou*[2]

[1] Microsoft Research Asia, Beijing, China
[2] State Key Laboratory on Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Automation, Tsinghua University, Beijing 100084, China

ABSTRACT

This paper presents an approach to learn a better music similarity measure and presents an application to music playlist generation. Different from previous work, in our approach, automatically detected music attributes are used to represent each song. A set of kernels is employed in similarity measure, with each kernel measuring on a subset of music attributes and having a different importance weight. In automatic music playlist generation, a ranking method is presented, which considers multiple seed songs and possible outlier seed. Experiments show the effectiveness of the proposed approach, and the quality of the playlist generated based on automatic annotations is comparable to that based on manual annotations.

Index Terms— Music similarity, music annotation, playlist generation, music recommendation

1. INTRODUCTION

With thousands of songs in personal collection, and millions of songs available with online subscription, how to efficiently select favorite songs to listen and effectively discover new songs becomes a challenge. To address this challenge, considerable approaches have been explored to estimate the similarity between songs, and to apply this similarity in various applications, such as music retrieval, music recommendation, music discovery, and music playlist generation.

To estimate music similarity, a number of approaches worked on social data or user data. For example, [1] employed collaborative filtering to measure the similarity between songs based on users' preference rating; and [2] built a graph model to infer music similarity based on their co-occurrence in "authored" streams, such as a music radio program made professionally. However, for those long tail songs or new songs without sufficient user data, these techniques become infeasible. To compensate for this disadvantage, some approaches resorted to analyze music content and use music attributes in similarity measure. For instance, [3] compared low-level music features such as Mel Frequency Cepstrum Coefficients (MFCCs) and spectrum histogram. Although the effectiveness of this method was reported, a major disadvantage is the big semantic gap between low-level features and high-level human perceptions. On the other hand, [4, 5, 6] proposed to estimate music similarity based on mid-level music attributes (or tags), such as author, genre, and emotion. With this representation, the estimated similarity is more likely to match human perception; and it is also computationally efficient for further online retrieval and recommendation. However, most of the previous work, as well as some online services such as *Pandora* and *Last.fm*, relied on manually annotated tags, which is expensive and unscalable in dealing with a large music collection.

In this paper, we propose an alternative method to learn and estimate music similarity based on *automatically* detected music attributes, following our previous work on *automatic* music annotation [7], in order to reduce manual efforts. Moreover, to estimate music similarity based on music attributes, direct binary matching [5] or a simple cosine measure [8] can be further improved with careful consideration. An intuitive idea is that, different attributes of music may play different roles, and thus need have different weights in similarity measure. Following the idea of Kernel Meta-Training (KMT) in [4], which employed a family of Mercer kernels to fit the "underlying" similarity function, we also utilize a set of kernels to learn music similarity measure, with each kernel measuring on a subset of music attributes. The weights of each kernel will be learned from a training data set, representing the importance of different attributes. Furthermore, we applied the learned similarity measure in automatic music playlist generation based on a set of seed songs, where we take into account the possible outlier seed and sets different importance weights to different seed songs, correspondingly.

The remainder of our paper is organized as follows: Section 2 summarizes our previous work on automatic music annotation. Section 3 presents our approach to learn a music similarity measure. The playlist generation and ranking method are described in Section 4, and the evaluation results are presented in Section 5.

2. AUTOMATIC MUSIC ANNOTATION

Music annotation aims to assign tags or labels from a semantic vocabulary to a song, it can be viewed as a multi-label classification problem. Given a vocabulary $\mathcal{V}$ consisting $|\mathcal{V}|$ tags $w_i \in \mathcal{V}$, and given a song represented by a bag of feature vectors $\mathcal{X} = \{x_1, ..., x_T\}$, the goal of music annotation is to find a set of tags $\mathcal{W} = \{w_1, ..., w_A\}$ that best describe the song. It will be convenient to represent the set $\mathcal{W}$ as an annotation vector $\mathbf{y} = \{y_1, ..., y_{|\mathcal{V}|}\}$. While y_i can be a binary variable representing the "presence" or "absence" of w_i, in our approach, y_i is used as a *confidence score* indicating the probability that the song can be labeled by tag w_i. Assuming each tag is independent and has identical prior probability distribution, we have,

*This work was performed when the first author was a visiting student at Microsoft Research Asia

Attribute	Tag / Label	Num
Genre	Blues, Country, Electronica, Folk, Funk, Gospel, HardRock, Jazz, Pop, Punk, Rap, R&B, Rock-roll, SoftRock	1-2
Instrument	Acoustic Guitar, Acoustic Piano, Bass, Drum, Electric Guitar, Electric Piano, Harmonica, Horn, Organ, Percussion, Sax, String	1-5
Texture	Acoustic, Electric, Synthetic	1-2
Vocal	Group, Male, Female, None	1-2
Affective	Positive, Neutral, Negative	1
Arousal	Strong, Middle, Weak	1
Rhythm	Strong, Middle, Weak	1
Tempo	Fast, Moderato, Slow	1
Tonality	Major, Mixed, Minor	1
Production	Studio, Live	1

Table 1. The vocabulary contains 50 tags covering 10 attributes (semantic categories). Each song is annotated using tags from each attribute with number limitation.

$$y_i = P(w_i|\mathcal{X}) = \frac{p(\mathcal{X}|w_i)P(w_i)}{\sum_{w_k} p(\mathcal{X}|w_k)P(w_k)} \propto \Pi_{t=1}^{T} p(x_t|w_i) \quad (1)$$

In our previous work [7], we presented an approach to automatic music annotation. Since there is no standard vocabulary for music annotation, we first built a simplified (but still general) vocabulary, containing 50 commonly used tags and covering 10 different music attributes, as listed in Table 1. A song will be annotated using tags from each attribute with number limitation, for instance, multiple tags can be selected from attributes Genre, Instrument, Texture and Vocal, while for other attributes, the tags are exclusive with each other. A bag of beat-level features are then extracted from each song, including timbre features (e.g. MFCC, spectral contrast) and rhythm features (e.g. average tempo, rhythm regularity), and each tag is modeled by a on-the-shelf GMM to estimate the posterior probability (1). More details about music annotation can be found in [7].

It is noted that, in (1), y_i is normalized across the tags from the same attributes as the tag w_i, instead of across the whole vocabulary. The obtained annotation vector $\mathbf{y} = \{y_i\}$ can be considered as a confidence-based representation of a song. It can also be binarized by selecting several largest y_i in each attribute (set them to 1 and others 0) [7]. In this work, both confidence and binary representation are used and compared in music similarity learning.

3. THE SIMILARITY MEASURE

3.1. Formulation of Similarity Function

In general, different attributes of music play different roles when people determine the similarity between songs. Furthermore, some attributes are strongly correlated so that people tend to combine them together to evaluate the music similarity. Based on these observations, we decide to construct a similarity function by using a set of kernels, each of which measures on (one or multiple) different music attributes. The general form of the similarity function used in our approach is expressed as:

$$\mathrm{sim}(\mathbf{y}_i, \mathbf{y}_j) = \sum_{n=1}^{N_\phi} \beta_n \phi_n(\mathbf{y}_i, \mathbf{y}_j) \quad (2)$$

where $\mathbf{y}_i$ and $\mathbf{y}_j$ are the annotation vectors of two songs, ϕ_n is a predefined kernel which in our approach is selected as cosine kernel, β_n is its corresponding weight, and N_ϕ is the number of kernels.

As mentioned above, each ϕ_n can measure on a subset of music attributes, e.g. only on Genre, or on a combination of Genre and Tempo. Depending on the number of attributes the kernel considers, we group kernels into *Kernel Families*. For example, if a kernel considers a combination of k music attributes, its kernel family is denoted as F_k. Thus, assuming the total number of attributes is N (in our case $N = 10$), there are C_N^k kernels in family F_k, where each kernel considers different combination of k out of N attributes, and is denoted as $\phi_{k,l}$, $l < C_N^k$. Since a particular music attribute corresponds to a sub-vector of the annotation vector $\mathbf{y}$, representing the tags only from this attribute, $\phi_{k,l}(\mathbf{y}_i, \mathbf{y}_j)$ can be computed from the concatenated sub-vectors of $\mathbf{y}_i$ and $\mathbf{y}_j$, representing k attributes.

Considering the kernel family, the similarity function becomes:

$$\mathrm{sim}(\mathbf{y}_i, \mathbf{y}_j) = \sum_{k \in N^s} \sum_{l=1}^{|F_k|} \beta_{k,l} \phi_{k,l}(\mathbf{y}_i, \mathbf{y}_j) \quad (3)$$

where $|F_k|$ is the number of kernels in the k^{th} family, and N^s is kernel family set (a subset of $\{1, 2, \cdots, N\}$), providing the opportunity to select which kernel families to be used in the similarity function. This is important because, adding more kernel families may introduce over-fitting, especially when the training data is small. By selecting N^s, we can control the generalization ability of the similarity function, and the N^s can be determined by experiment.

3.2. Learning Similarity Measure

In this section, we present how to learn the weighting parameter in the similarity function from a set of manually created playlists (i.e., a training set) which specifies the ground truth similarity between songs. Intuitively, the best parameters minimize the difference (squared error) between the ground truth similarity and the learned similarity function. Regardless which kernel families are selected, we use the general form of similarity function (2) in the following cost function,

$$F(\boldsymbol{\beta}) = \frac{1}{2} \sum_{i,j} \left(K_{i,j} - \sum_{n=1}^{N_\phi} \beta_n \phi_n(\mathbf{y}_i, \mathbf{y}_j) \right)^2 \quad (4)$$

where $K_{i,j}$ is the ground truth similarity of the i^{th} and j^{th} songs, and $\boldsymbol{\beta} = [\beta_1, ..., \beta_{N_\phi}]^T$, with constrain $\beta_n > 0$. We apply this constrain to ensure the generalization of the learned similarity function.

The cost function (4) can be re-written in a matrix form:

$$F(\boldsymbol{\beta}) = \frac{1}{2} \boldsymbol{\beta}^T A \boldsymbol{\beta} - b\boldsymbol{\beta} + \|K\|_F \quad (5)$$

where $\|K\|_F$ is the Frobenius norm of ground truth similarity matrix K, and the matrix A and vector b represented by

$$\begin{aligned} A(m,n) &= \sum_{i,j} \phi_m(\mathbf{y}_i, \mathbf{y}_j) \phi_n(\mathbf{y}_i, \mathbf{y}_j) \\ b(m) &= \sum_{i,j} K_{i,j} \phi_m(\mathbf{y}_i, \mathbf{y}_j) \end{aligned} \quad (6)$$

where $1 \le m, n \le N_\phi$. The optimal $\hat{\boldsymbol{\beta}} = \mathrm{argmin}_{\boldsymbol{\beta}} \{F(\boldsymbol{\beta})\}$ can be iteratively solved by using Non-Negative Quadratic Optimization [9], which is mainly based on Gradient Projection method.

4. MUSIC PLAYLIST GENERATION

With the learned similarity measure, we further apply it in a real application - music playlist generation. In this application, users first select *one* or *multiple* seed songs to setup their preference (i.e., what kind of songs they want to listen to), then a sequence of similar songs will be recommended and a playlist is generated correspondingly. Each song in the database can be ranked by a weighted sum of its similarity to each specified seed, as,

$$f_* = \sum_{i=1}^{N} \alpha_i \mathrm{sim}(\mathbf{y}_i, \mathbf{y}_*) \tag{7}$$

where $\mathbf{y}_*$ is a song in the database, f_* is its corresponding ranking score (or the estimated user preference score); $\mathbf{y}_i$ is the i-th seed, α_i is the corresponding weight indicating the importance of seed $\mathbf{y}_i$, and N is the number of seeds. The $\mathrm{sim}(\cdot)$ is the similarity function defined in (2).

The simplest strategy to determine α_i is to use uniform weighting, i.e. $\alpha_i = 1/N$, supposing each seed has the same importance. Alternatively, different seeds may have different importance to represent user preference, so that they can be assigned different weights. For example, [4] applies Gaussian Process Regression (GPR), where the value of α_i is determined by computing the inverse of the similarity matrix of seeds:

$$\boldsymbol{\alpha} = C^{-1}\mathbf{t} \tag{8}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_N]^T$ is a weighting vector, C is the similarity matrix of seeds with $C_{ij} = \mathrm{sim}(\mathbf{y}_i, \mathbf{y}_j)$, and $\mathbf{t} = [t_1, \ldots, t_N]^T$ is the user's preference to seeds ($t_i = 1$ if like the seed; or 0 otherwise). In our case, since users only select the seeds they like, all $t_i = 1$.

However, (8) will encounter a problem if there exists an outlier (noise) seed that is not similar to other selected seeds (i.e. far away from other seeds in the feature space). Intuitively, the weight of such outlier seed should be small, since it can not represent the majority of the user preference. However, in contrast, (8) assigns the largest weight to it. This is mainly because, in a regression problem estimating target (user preference) from a number of samples (seeds), a outlier only refers to a sample whose target is quite different from its neighbors' targets; a seed far away from others is not an outlier, but a sample needs large weight to play its role in prediction.

We make a *heuristic* revision from (8) to address the aforementioned problem, by directly using the similarity matrix C to calculate the weights, as

$$\boldsymbol{\alpha} = C\mathbf{t} \tag{9}$$

where each element $\alpha_i = \sum_j \mathrm{sim}(\mathbf{y}_i, \mathbf{y}_j)$. Apparently, for an outlier seed which is dissimilar to others, its corresponding weight will be smaller than others.

5. EXPERIMENTS

Our previous work [7] built a 5000-song data set for automatic music annotation, on which an annotation accuracy of about 60% is achieved. For music similarity learning in this work, the ideal training data had better indicate the similarity between every two songs in the data set. However, it is very hard to obtain. Alternatively, we collect 380 real user created playlists from Internet, and use them to build the ground truth similarity matrix K in (4). In our experiment, $K_{i,j} = 1$ if two songs co-occur in at least one playlist, otherwise $K_{i,j} = 0$. We also tried different ways to calculate K, for instance, by considering the co-occurrence frequency, however the experiments don't show any improvements.

While these playlists are sufficient for training, a big issue in evaluation is that, we only assume the songs in the same playlist are similar, but the songs not in the same playlist may be also similar. This is because the playlist may only contain a subset of similar songs from a large data set. Thus, if we directly test on the 5000-song data set, a number of "fake" false alarms are generated. For this reason, in our objective evaluation, we only choose (randomly) a small test set (200 songs) from our data set and manually create 23 playlists, in which the songs in different playlists are more perceptually different than those in the same playlist.

The evaluation process is similar to [4]: for each playlist, every S (varies from 1 to 5) songs are selected as seeds, and then match against the remaining songs in the same playlist (which are similar to seeds) and the songs in other playlists (which are dissimilar to seeds). A traditional collaborative filtering (CF) metric [4, 10] is used to measure the quality of the generated playlists. The corresponding score of the j^{th} trial is defined as,

$$R_j = \sum_{i=1}^{N_j} \frac{t_{ji}}{2^{(i-1)/(\beta-1)}} \tag{10}$$

where $t_{ji} = 1$ if the i^{th} song in the generated playlist is from the same playlist as the seeds, otherwise $t_{ji} = 0$. β is a "half-life" of user interest in the playlist (set to 10 here), and N_j is the number of test songs in the j^{th} trial. The score is summed over all trials and normalized:

$$R = 100 \sum_j R_j / \sum_j R_j^{\mathrm{best}}, \tag{11}$$

where R_j^{best} is the best score can be achieved from (10) when all the similar songs are at the head of the generated playlist. Thus, an R score of 100 indicates the best result.

We first carry out an experiment to determine the kernel family set N^S used in (3), using the confidence based annotation and the ranking method (9). While there are many possibilities to select N^S, we only test a subset of them. N^S is first set to $\{10\}$, which contains only one kernel composing of all the attributes, and is usually used in the previous work and also taken as our baseline. Then we incrementally add kernel family 1,2,...,9 into N^s one by one, and finally we obtain and compare with 10 similarity functions whose N^s are $\{10\}$, $\{1, 10\}$,...,$\{1, 2, ..., 9, 10\}$. The comparison result is illustrated in Fig. 1. From the result, we observe that the best performance is achieved when $N^S = \{1, 2, 10\}$, whatever the number of seeds is. With more kernel families added, the performance decreases, mainly due to the over-fitting problem. Therefore, in the following experiments, we fix N^S to be $\{1, 2, 10\}$. We also find the learned weights of the "Genre" related kernels are significantly larger than others, indicating the importance of Genre in determination of music similarity.

In the second experiment, we compare different weighting method in Section 4, including our method (9), GPR (8), and uniform weighting, and compare the results without/with using outlier seed, while keep using confidence based annotation. The comparison results are shown in Table 2. We can see that, in both cases, our method performs best. When an outlier seed is introduced (by replacing one seed in the original seeds with a song from another playlist), our approach has the least degeneration (2 points in average) compared with others. It is interesting to note, in both cases, GPR ranking has the worst performance.

The third experiment compares our method with two baselines: random selection and cosine similarity measure on all the attributes,

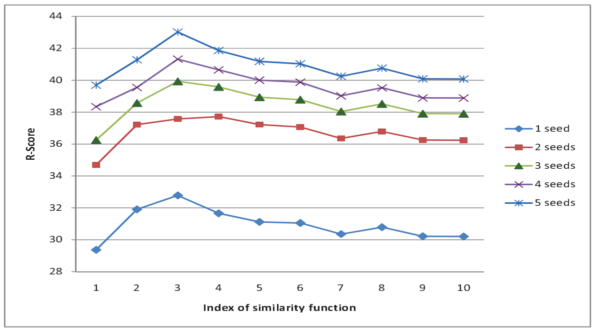

Fig. 1. Evaluation results comparing various kernel families and the number of seeds, using confidence based annotation and the ranking method (9)

#seed	w/o outlier seed			w/ outlier seed		
	ours	*GPR*	*uni.*	*ours*	*GPR*	*uni.*
1	**32.8**	32.8	32.8	NA	NA	NA
2	**37.6**	37.0	37.0	NA	NA	NA
3	**39.9**	38.0	39.2	**32.6**	29.1	31.7
4	**41.3**	39.1	40.8	**37.6**	31.7	36.4
5	**43.0**	39.1	42.7	**40.5**	33.0	39.1

Table 2. Results comparing different ranking methods including our method (*ours*), GPR, and uniform weighting (*uni.*), different seed number, and different seed selection, including *without outlier seed* and *with outlier seed* (only when the seed number is larger than 2)

when fixing $N^S = \{1, 2, 10\}$ and the ranking method (9). We also compare different annotation schemes, including manual annotation (with oracle music attributes), automatic annotation with confidence score or binary score. The results are shown in Table 3, from which we can observe: First, all the approaches work much better than the random selection; and moreover, our method consistently outperforms the direct cosine measure (by 3.3 points in average), indicating the proposed approach can learn something reflecting underlying music similarity. Second, using confidence score in annotation obtains a consistent improvement (about 1 point) compared with the binary annotation. This is understandable since the confidence score contains richer information than binary one. Third, comparing with the manual annotation, the performance based on automatic annotation is comparable, with a gap of 4.8 points (when the annotation accuracy is about 60%); with the number of seeds increases, the performance gets closer to that of manual annotation.

Besides the objective evaluation, we also perform a preliminary subjective evaluation on our 5000-song data set. In this experiment, we first randomly select 10 songs as seeds, and for each seed we generate a playlist containing 10 most similar songs to the seed. Three subjects are invited to rate each song in the generated playlist based on its perceptual similarity to the seed, with 5 scores from 1 (not similar at all) to 5 (very similar). The average rating score across all playlists achieves up to 4.2, which indicates that most of the recommended songs are similar to the seeds in terms of the human perception.

All the experiments also show that, with automatic annotation, we can get encouraging results on music similarity measure and music playlist generation. Considering the advantage of automatic annotation, it is feasible to employ automatic annotations in other music applications, dealing with a large music database.

#seed	Random	Baseline	Kernel + Ranking		
	–	*conf.*	*conf.*	*bin.*	*oracle*
1	8.7	29.4	32.8	31.9	38.0
2	9.0	34.7	37.6	36.8	44.6
3	8.1	36.2	39.9	38.8	44.2
4	8.5	38.1	41.3	40.6	46.8
5	8.0	39.6	43.0	42.0	45.3

Table 3. Results comparing our method with random selection and Cosine baseline, as well as different annotation schemes, including manual annotation with oracle music attributes (*oracle*), automatic annotation with confidence score (*conf.*) and binary score (*bin.*)

6. CONCLUSION AND FUTURE WORK

In this paper, we present an approach to learn a music similarity measure based on automatically detected music attributes, and apply it in automatic music playlist generation. Extensive experiments compare various methods/variants, and show the advantage of the learned similarity measure and show the encouraging results of the playlist generation. With an annotation accuracy of about 60%, the generated playlists is comparable to those based on manual annotations. And, as indicated by the high subjective ranking score, most of the recommended songs are similar to the seeds. There is still considerable room to improve music similarity measure and music playlist generation. For example, we can investigate more music attributes or incorporate social data and other metadata in similarity measure; we will also exploit and utilize user interactions to achieve better music recommendation.

7. REFERENCES

[1] William W. Cohen and Wei Fan, "Web-collaborative filtering: Recommending music by crawling the web," *Computer Networks*, vol. 33, pp. 685–698, 2000.

[2] R. Ragno, C.J.C. Burges, and C. Herley, "Inferring similarity between music objects with application to playlist generation," *Proc. ACM Workshop on Multimedia Information Retrieval*, pp. 73–80, 2005.

[3] E. Pampalk, S Dixon, and G. Widmer, "On the evaluation of perceptual similarity measures for music," *Proc. of Inter. Conf. on Digital Audio Effects*, pp. 7–12, 2003.

[4] J.C. Platt, C.J.C. Burges, S. Swenson, C. Weare, and A. Zheng, "Learning a gaussian process prior for automatically generating music playlists," *NIPS*, pp. 1425–1432, 2001.

[5] J.J. Aucouturier and F. Pachet, "Scaling up playlist generation systems," *Proc. ICME*, pp. 105–108, 2002.

[6] S. Pauws and B. Eggen, "Pats: Realization and evaluation of an automatic playlist generator," *Proc. 3rd International Simposium on Music Information Retrieval*, pp. 222–230, 2002.

[7] Z. Y. Duan, L. Lu, and C. S Zhang, "Collective annotation of music from multiple semantic categories," *Proc. of ISMIR*, pp. 237–343, 2008.

[8] K. Kaji, K. Hirata, and Nagao K., "A music recommendation system based on annotations about listeners' preferences and situations," *IEEE AXMEDIS*, pp. 231–234, 2005.

[9] R.M. Bell, Y. Koren, and C. Volinsky, "Modeling relationship at multiple scales to improve accuracy of large recommendation systems," *ACM KDD*, pp. 95–104, 2007.

[10] J.S. Breese and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," *Uncertainty in Artificial Intelligience*, pp. 43–52, 1998.