PlaylistPlayer: An Interface Using Multiple Criteria to Change the Playback Order of a Music Playlist

Tomoyasu Nakano **Jun Kato** **Masahiro Hamasaki** **Masataka Goto**
National Institute of Advanced Industrial Science and Technology (AIST), Japan
{t.nakano, jun.kato, masahiro.hamasaki, m.goto}@aist.go.jp

ABSTRACT
We propose a novel interface that allows the user to interactively change the playback order of multiple songs by choosing one or more criteria. The criteria include not only the song's title and artist name but also its content automatically estimated by music/singing signal processing and artist-level social analysis. The artist-level social information is discovered from Wikipedia and DBpedia. With regard to manipulating playback order, existing interfaces typically allow the user to change it manually or automatically by choosing one of a few types of criteria. The proposed interface, on the other hand, deals with nine properties and multiple integrations of them (*e.g.*, vocal gender and beats per minute). To realize the ordering by multiple criteria, a distance matrix is computed from the criteria vectors and is then used to estimate paths for ascending, descending, and random orders by applying principle component analysis or to estimate a path for a smooth order by solving the travelling salesman problem.

ACM Classification Keywords
H.5.5. Information Interfaces and Presentation (*e.g.*, HCI): Sound and Music Computing – Signal analysis, synthesis, and processing; H.5.2. Information Interfaces and Presentation (*e.g.*, HCI): User Interfaces – GUI.

Author Keywords
Music playlist; playback order; visualization; vocal gender; musical commonness; social commonness.

INTRODUCTION
Playlists, lists of songs played back automatically, are a popular way of listening to music. The playback order can be determined using metadata such as artist names and song titles, and the "freshness" of a playlist listened to repeatedly can be increased by randomly shuffling the playback order.

We propose a playlist playback interface, *PlaylistPlayer*, that enriches the listener's experience by allowing interactively changing the playback order based on various properties of the songs and their artists. These properties, derived from content and from web, are visualized in the extended playlist interface (Figure 1). The following three kinds of properties can be visualized and used to change the playback order:

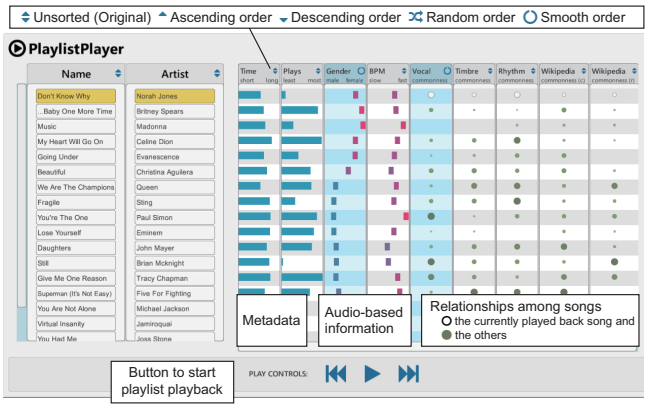

Figure 1. An example PlaylistPlayer screen.

(type 1: absolute values) meta-data:
intrinsic information (song length) and information given by a listener's behavior (the number of views),
(type 2: absolute values) audio-based information:
vocal information (vocal gender) and musical information (beats per minute: BPM),
(type 3: relative values) relationships among songs:
musical commonness [14] (a song's similarity to a set of songs) of vocal timbre, musical timbre, and rhythm; and social commonness (artist-level social relationships between a song and a set of songs) of artist background information on the web.

Any of those properties can be used as a re-ordering criterion, and two or more properties (*e.g.*, vocal gender, BPM and musical commonness) can be used as re-ordering criteria. Integrating multiple properties enables designing a variety of playback orders that match the user's needs.

RELATED WORK
Previous work on playlist generation is related to our interface's changing playback order, and previous work on playlist visualization in order to discover a song is related to our interface's visualization.

Playlist Generation
Common approaches to creating playlists on existing music players such as *iTunes* are to manually select songs, using some conditions to create complex playlists (*Smart Playlist*), or to select songs automatically (*Genius*).

IUI'16, March 07–10, 2016, Sonoma, CA, USA

ACM 978-1-4503-4137-0/16/03...$15.00.
DOI: http://dx.doi.org/10.1145/2856767.2856809

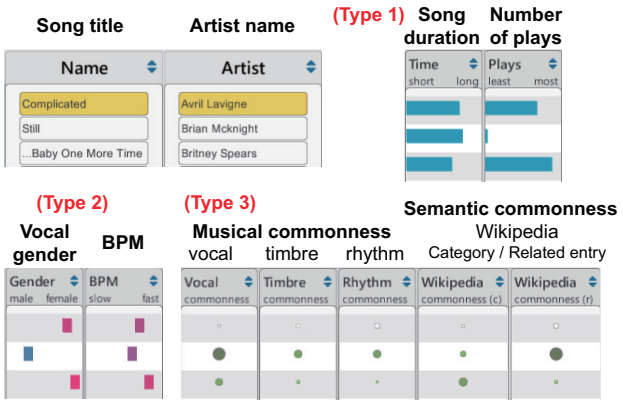

Figure 2. The tabular visualization of all properties.

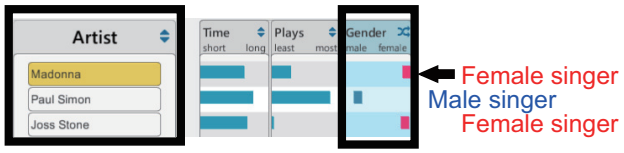

Figure 3. The random order of the vocal gender.

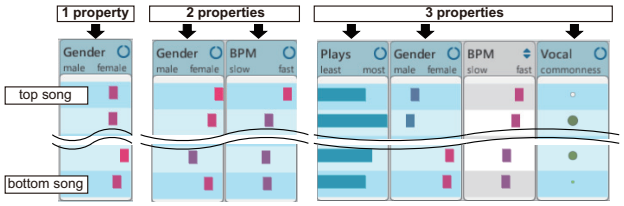

Figure 4. Re-ordering by the smooth order using multiple properties.

On the other hand, much research has also been focused on playlist generation [2, 18]. Research on generating playlists based on smooth transition between one song and the previous and next songs (acoustically similar pieces) [4, 17] is related to our system in terms of playback order. And research considering human behavior during playback (*e.g.*, skipping a song) and generating a playlist by dynamically adapting to the user's preferences [11,15] is related to our system in terms of using human behavior (*i.e.*, the number of views).

PlaylistPlayer, however, uses metadata, audio data, and behavioral data not for 'generating a new playlist but for changing the order in which an existing playlist is played back'. Moreover, multiple criteria can be used by integrating the individual criteria.

Playlist Visualization

There has been much work in which visualization of relationships among songs was used to browse, to explore, and to discover a song [12]. Some research focusing on combining visualization with playlist generation is related to our system, such as interactive playlist generation [20], generating a thumbnail image that represents audio features of a playlist [19], and printing out album covers for use as tangible interfaces of playlists [8].

There has also been work on visualizing artist relationships based on musical background (*e.g.*, female solo-singer), moods, tempo, genres, active years and decades [7, 20], and associated activities (*i.e.*, members of a band) [9].

PlaylistPlayer, in contrast, uses tabular type visualization as an extension of the standard playlist. Our system visualizes not only *static* values (*i.e.*, metadata and acoustic characteristics) but also *dynamically* changing values (*i.e.*, relationships among songs). In addition, artist relationship includes artists' personal information on the web.

PLAYLISTPLAYER: AN EXTENDED PLAYLIST

PlaylistPlayer is a playlist interface which is an extended classical list-view style playlist. To realize PlaylistPlayer, first the musical characteristics of the audio data for all the songs in the playlist are estimated automatically. The characteristics used in the current implementation are BPM, vocal gender, commonness of vocal timbre, musical timbre, and musical rhythm. Second, we crawl the web for artist-level social relationships among all songs in the playlist.

Finally, we implement an interface for interactions during changing playback order and visualization. In this section we describe the interface and the interactions. The re-ordering, signal processing and web mining methods are described in detail in the next section.

Change Playback Order and Visualization

PlaylistPlayer can change playback order according to the three types of properties listed in the INTRODUCTION. Figure 2 shows all three types with some tabular visualization methods by reference to [16]. Type-1 properties are represented by blue rectangles in horizontal bar graphs. Type-2 properties are represented by colored rectangles (blue or red). For example, if a male probability of vocal gender is high, the rectangle is colored blue and placed at the left side. And type-3 properties are represented by green circles whose sizes depend on relationships (musical or social commonness). The reason for using the commonness instead of the similarity between two songs is to fit to the tabular type visualization.

To change the order, sort/unsorted marks, a smooth mark, and a random mark are placed on top of each property column (Fig. 1). A single triangular shape means ascending or descending order, the circle shape means all song transition are smooth (including a transition between top and bottom songs), and the curved arrow means random order. Random order means that all the songs in the playlist are ordered alternately with regard to the selected properties. For example, if the random mark of the "Gender" property is checked (Fig. 3), a song sung by a male singer is played after a song sung by a female singer. When there are two triangular shapes on one property, that property is not considered when determining the order (i.e., the order is the original predefined order).

The listener can change the order by clicking the marks of one or more properties each time. If two or more properties are selected, these marks are coordinated (Fig. 4).

IMPLEMENTATION

To test the interface, we used 19 songs as listed in Figure 5. They were chosen among 415 English songs performed by various types of artists (solo singers, male/female singers,

song ID	Artist name / Song title
1	Avril Lavigne / Complicated
2	Brian Mcknight / Still
3	Britney Spears / ...Baby One More Time
4	Celine Dion / My Heart Will Go On
5	Christina Aguilera / Beautiful
6	Eminem / Lose Yourself
7	Erykah Badu / On & On
8	Evanescence / Going Under
9	Five for Fighting / Superman (It's Not Easy)
10	Jamiroquai / Virtual Insanity
11	John Mayer / Daughters
12	Joss Stone / You Had Me
13	Madonna / Music
14	Michael Jackson / You Are Not Alone
15	Norah Jones / Don't Know Why
16	Paul Simon / You're The One
17	Queen / We Are The Champions
18	Sting / Fragile
19	Tracy Chapman / Give Me One Reason

Figure 5. Songs used in the evaluation.

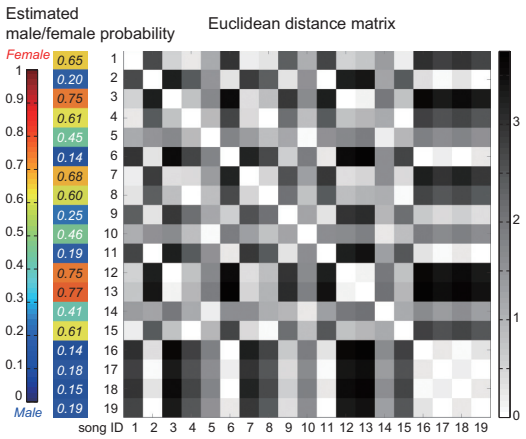

Figure 6. A Euclidean distance matrix calculated from estimated male/female probabilities in the 19 songs.

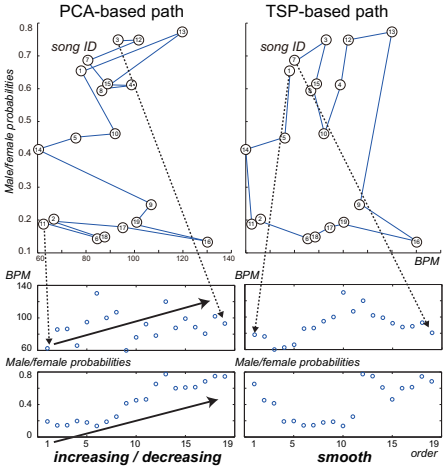

Figure 7. Two types of paths are estimated.

bands, or groups). They were taken from commercial music CDs (*Billboard Top Rock'n'Roll Hits* 1968-1972, *Billboard Top Hits* 1975-1989 and *GRAMMY NOMINEES* 1996-2005).

Re-Ordering using multiple criteria

All properties among songs can be represented as an $N \times N$ Euclidean distance matrix (N is the number of songs of the playlist). If the listener selects two properties as criteria for re-ordering, such as vocal gender and BPM, a distance between two songs is calculated from the 2-dimensional vector. Before calculating the distance, each vector is normalized by subtracting the mean and dividing by the standard deviation. The Euclidean distance matrix A is defined as

$$A = (a_{ij})_{\substack{1 \le i \le N \\ 1 \le j \le N}}, \quad a_{ij} = \sqrt{\sum_{d=1}^{D} (x_i(d)^2 - x_j(d)^2)}, \qquad (1)$$

where $\vec{x_i}$ and $\vec{x_j}$ are the D-dimensional normalized vectors of two songs (see Figure 6 for concrete examples).

For changing playback order according to those properties, we compute a one-dimensional vector from the matrix to realize the ascending/descending/random order. The one-dimensional vector is computed by applying principle component analysis (PCA) for reduction of dimensions of the distance matrix. Then the one-dimensional vector is sorted.

The path for smooth transition is estimated by solving the travelling salesman problem (TSP). Although the TSP was used for getting acoustically-smoothed transition in previous works [4, 17], the originality of the PlaylistPlayer is that the TSP used to integrate multiple criteria (Fig. 4).

The PCA-based and TSP-based methods can scale up to at least hundreds of songs (completed in a second on a personal computer). Moreover, their computational efficiencies can be increased by using the EM algorithm for PCA [1] and some approximate algorithms for TSP [17]. For the PCA, the most computationally demanding steps of the EM algorithm are $O(NDM)$, where M denotes the number of principle components [1]. For the TSP, the *Minimum Spanning Tree* is $O(m \log m)$ with m being the number of edges [17].

Figure 7 shows that path-estimation results obtained by taking into account two properties (vocal gender and BPM).

Music/singing signal processing

BPM (type 2) of each song is estimated by using the median value of the time intervals between individual beats during one song. To estimate beats, we use modules of Songle [6].

Since we focus on vocal timbre, musical timbre, and rhythm, we extract acoustic features representing them from the musical audio signals [14]. These features are used to estimate the vocal gender (type 2), and the musical commonnesses (type 3) of songs.

Vocal Gender Estimation (type 2)

Our system also uses a vocal gender estimation method [10]. Male/female probabilities of each *reliable frame* (see [10]) are estimated using a probabilistic Support Vector Machine (SVM) [3] and are reduced to a single value by taking the median across the SVM output.

Fifteen songs (13 Japanese and 2 English) sung by fifteen male singers and thirteen songs (11 Japanese and 2 English) sung by thirteen female singers are used as the training set of each model. These songs are solo vocal performances and are from the RWC Music Database (RWC-MDB-P-2001) [5].

Musical Commonness Estimation (type 3)

We use a musical commonness estimation method [14] based on latent Dirichlet allocation (LDA). A song-set (playlist) model can be obtained by summing the hyperparameters of the topic distributions of song models [14].

To train the LDA-based topic model, we used the 415 songs.

Web mining

To provide different type of relations among songs, we focus on artist properties. The artist information is obtained from Wikipedia (http://www.wikipedia.org/) and DBpedia [13]. DBpedia has structured information extracted from Wikipedia. From each artist entry (page) are extracted phrases in the Categories section (*e.g.*, Grammy Award-winning artists) and related entries' names that have a link

Figure 8. A Wikipedia artist entry (page) and the kinds of information.

with another entry on Wikipedia. An example of the extracted information is shown in detail in Figure 8.

We investigated social information among 335 artists of the 415 songs. The total numbers of social information are 2,154 categories and 15,658 related entries. There was music-related information (*e.g.*, Musical quartets, American pop singers, and genre), personal information (*e.g.*, 1942 births), and information about awards.

Social Commonness (type 3)

We estimate social commonness by using the categories and the related entries as social annotations. The estimation procedure is similar to the musical commonness estimation procedure based on the LDA. An artist-set (playlist) model can also be obtained by integrating artist models.

Evaluation

Accuracies of the proposed estimation methods are calculated for the 19 songs. To evaluate the vocal gender estimation, estimated male/female probabilities are shown in Figure 6 and they are binarized ("male" if the probabilities are equal to or lower than 0.5, "'female" if the probabilities are greater than 0.5). The classification rate was 94.7% (18/19 songs).

Estimated BPMs are comparable to annotated BPMs by a musician. To evaluate the estimation method, we computed five BPMs for each song based on five beat structure candidates estimated by Songle [6]. Then, we chose the BPM that is the closest to the annotated BPM. A root-mean-square value of the difference between the chosen (estimated) BPMs of 19 songs and the annotated BPMs was 0.78.

Estimated musical commonness and social commonness are evaluated based on the Pearson product-moment correlation coefficients between estimated commonness of each song and the number of songs having high similarity with the song [14]. If the correlation coefficients are high, the estimated commonness is appropriate. This means a song having higher (lower) commonness and it is very similar (is not similar) to songs of a song set. The correlation coefficients of the three musical elements (vocal timbre, musical timbre and rhythm) were 0.87, 0.91 and 0.80, and of the two social information (category and related entry) are 0.81 and 0.85.

USER FEEDBACK

To investigate the capabilities, limitations, and potential of our interaction design, we asked seven users to use the system for 20 minutes and collected preliminary user feedback. We chose the users (six males and one female) so that they have different levels of musical experience/knowledge. Six users had experience with musical instruments, and three users had experience with arrangement/composition. The playlist consisted of the 19 songs (Figure 5). All users knew more than 1 song, and four users knew more than 10 songs. After the trial usage, we asked the users to write comments about the two primary functions of the system: 1) song information visualization and 2) changing the playback order.

Positive comments about the capabilities and potential of the interface are listed below.

Visualization of multiple properties This is helpful when exploring the playback order that meets my expectation.

Automatic re-ordering based on content-based criteria 1) Easy to use and much appreciated. 2) Helpful for finding new songs similar to songs of favorite artists. 3) The smooth order creates smooth transitions between songs which are suitable for background music. 4) The random order gives fresh listening experience for a worn-out set of songs.

Both functions 1) Both were useful for listening songs in a new perspective. 2) Both are likely to be helpful for generating a new playlist. 3) The properties for social commonness were useful because they matched the impression of the songs/artists.

There were also negative comments pointing out the limitations of the current implementation.

Visualization of multiple properties 1) The visualization of the social commonness properties was not comprehensive enough. 2) There are sometimes mismatches between visualization results and auditory impression. 3) The type 3 visualization was sometimes confusing.

Automatic re-ordering based on content-based criteria 1) Since there are a lot of properties (nine), the re-ordering operation by integrating them is a bit more complex. 2) The commonness order is hard to comprehend.

Both functions / Other perspective 1) There is too much information for casual listening. 2) Some properties require prior knowledge of music to comprehend the ordering rules. 3) Functions to compare results of sorting and to filter the results based on multiple criteria are desired. 4) Released date of songs can be added as a new property for sorting.

These comments suggest that the interface is effective, but some visualizations/functions should be improved.

CONCLUSION

This paper presents PlaylistPlayer, a playlist playback interface that can be used to change the playback ordering of existing playlists by using multiple criteria interactively. Although this paper did not discuss the "repeat" function, which is an standard function to use for playlist playback, the proposed criteria (or combinations of them) can be used to change after finishing playback of all songs to create various kinds of repeats. In addition, we intend in future work to further improve the interaction in ways inferred from user feedback.

ACKNOWLEDGMENT

This paper utilized the RWC Music Database (Popular Music). This work was supported in part by CREST, JST.

REFERENCES

1. Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006.
2. Bonnin, G., and Jannach, D. Automated generation of music playlists: Survey and experiments. *Journal of ACM Computing Surveys (CSUR) 47*, 2 (2015), 1–35.
3. Chang, C.-C., and Lin, C.-J. Libsvm: A library for support vector machines. *ACM TIST 2* (2011), 1–39.
4. Flexer, A., Schnitzer, D., Gasser, M., and Widmer, G. Playlist generation using start and end songs. In *Proc. ISMIR 2008* (2008), 173–178.
5. Goto, M., Hashiguchi, H., Nishimura, T., and Oka, R. RWC music database: Popular, classical, and jazz music databases. In *Proc. of ISMIR 2002* (2002), 287–288.
6. Goto, M., Yoshii, K., Fujihara, H., Mauch, M., and Nakano, T. Songle: A web service for active music listening improved by user contributions. In *Proc. ISMIR 2011* (2011), 311–316.
7. Gouyon, F., Cruz, N., and Sarmento, L. A Last.fm and YouTube mash-up for music browsing and playlist edition. In *Proc. ISMIR 2011* (2011), 1–2.
8. Graham, J., and Hull, J. J. iCandy: a tangible user interface for iTunes. In *Proc. CHI2008* (2008), 2343–2348.
9. Hamanaka, M., and Yoshiya, M. BandNavi: Band-member backtracker based on member history information. *Journal of Communication and Computer 9*, 2 (2012), 114–121.
10. Hamasaki, M., Goto, M., and Nakano, T. Songrium: A music browsing assistance service with interactive visualization and exploration of a web of music. In *Proc. WWW 2014* (2014), 523–528.
11. Hu, Y., and Ogihara, M. NextOne Player: A music recommendation system based on user behavior. In *Proc. ISMIR 2011* (2011), 103–108.
12. Langer, T. Music information retrieval & visualization. In *Trends in Information Visualization* (2010), 15–22.
13. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., and Bizer, C. DBpedia - a large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web Journal* (2012), 1–5.
14. Nakano, T., Yoshii, K., and Goto, M. Musical similarity and commonness estimation based on probabilistic generative models. In *Proc. ISM 2015* (2015), 197–204.
15. Pampalk, E., Pohle, T., and Widmer, G. Dynamic playlist generation based on skipping behavior. In *Proc. ISMIR 2005* (2005), 634–637.
16. Perin, C., Dragicevic, P., and Fekete, J.-D. Revisiting bertin matrices: New interactions for crafting tabular visualizations. *IEEE Trans. Visualization and Computer Graphics 20*, 12 (2014), 2082–2091.
17. Pohle, T., Pampalk, E., and Widmer, G. Generating similarity-based playlists using traveling salesman algorithms. In *Proc DAFx' 05* (2005), 1–6.
18. Song, Y., Dixon, S., and Pearce, M. A survey of music recommendation systems and future perspectives. In *Proc. CMMR 2012* (2012), 395–410.
19. Uota, T., and Itoh, T. GRAPE: A gradation based portable visual playlist. In *Proc. IV 2014* (2014), 361–364.
20. van Gulik, R., and Vignoli, F. Visual playlist generation on the artist map. In *Proc. of ISMIR 2005* (2005), 520–523.