# Towards User-Adaptive Structuring and Organization of Music Collections

Sebastian Stober and Andreas Nürnberger

Data & Knowledge Engineering Group
Faculty of Computer Science
Otto-von-Guericke-University Magdeburg, D-39106 Magdeburg, Germany
{nuernb,stober}@iti.cs.uni-magdeburg.de

**Abstract.** We present a prototype system for organization and exploration of music archives that adapts to the user's way of structuring music collections. Initially, a growing self-organizing map is induced that clusters the music collection. The user has then the possibility to change the location of songs on the map by simple drag-and-drop actions. Each movement of a song causes a change in the underlying similarity measure based on a quadratic optimization scheme. As a result, the location of other songs is modified as well. Experiments simulating user interaction with the system show, that during this stepwise adaptation the similarity measure indeed converges to one that captures how the user compares songs. This utimately leads to an individually adapted presentation that is intuitively understandable to the user and thus eases access to the database.

## 1 Introduction

Automatic structuring is one means to ease access to music databases, be it for organization or for exploration. Of even greater help would be a presentation that adapts to the user's way of structuring music collections and thus is intuitively understandable. So far, aspects of individualization have been only a minor issue of research in the field of Music Information Retrieval (MIR) and Music Digital Libraries (MDL). Often, it is assumed that all users of a MIR/MDL system compare music in the same (objective) manner. This assumption, however, is usually not true: A musician, for instance, might especially look after structures, harmonics or instrumentation (possibly paying extra attention to his own instrument). Non-musicians will perhaps rely more on overall timbre or general mood when comparing songs. Others, in turn, may go in for the lyrics as long as they are able to understand the particular language. A MIR/MDL system incorporating this subjectiveness will better comply with the individual needs of its users and consequently gain a higher acceptance. Scenarios where this would be especially helpful are the exploration and organization of a music collection. Here, users would greatly benefit if a system would not just simply structure the collection for easier access but would structure it in a way that is intuitively understandable for the individual user by adapting to its preferences

and needs. This, however, is not yet today's state of the art. At best, interfaces for music collection access allow for adaption by the user. However, they are lacking the ability to learn from user actions and to adapt on their own without explicit intervention of the user. In this paper, we present a prototype system that incorporates user adaptation in a simple yet effective manner.

In the following, we first give an overview on existing MIR/MDL systems that are either adaptable or adaptive as well as other related work sharing similar ideas. In Section 3 we describe our prototype system, specifically the used similarity facets and the adaptation method. Section 4 discusses the outcome of user simulations to objectively evaluate the adaptation method. Finally, we present ideas for future development in Section 5 and conclude with Section 6.

## 2 Related Work

Our approach builds upon the basic idea of adapting a similarity measures according to a user's preferences. This similarity can then be used to group songs in the collection and to generate a visual overview.

The idea of adapting similarity measures is not new: MPeer [3] allows to adjust the weight of three facets of music description in a similarity measure through an intuitive joystick interface for finding a set of similar songs given an anchor song.[1] The facets comprise the audio content, the lyrics and cultural metadata collected from the web. From a study with 10 users, it was concluded that users tend to use nearly similar joystick settings throughout different environments. Though the joystick interface is very intuitive, it is unclear whether it may be applied to more than 3 similarity facets. In the context of this paper, about 20 facets are used. Similarly, the E-Mu Jukebox [21] allows changing the similarity function that is applied to create a playlist from a seed song. Here, five similarity components (sound, tempo, mood, genre and year) are visually represented by adapters that can be dragged on a bull's eye. The closer a component is to the center, the higher is its weight in the similarity computation. This interface is scalable with respect to the number of facets but less intuitive. It may be hard for a user to explicitly specify a weighting scheme for the facets as this is usually something that only subconsciously exists. Especially with an increasing number of facets this is likely to become more difficult. Indeed, a user study with 22 participants showed that the users found the system harder to used but at the same time more useful compared to two control systems.

In contrast to the former systems, PATS (Personalized Automatic Track Selection) [13] is an adapative system for playlist generation that does not require manual adjustment of the underlying similarity measure but learns from user feedback. The system generates a playlist for a specific user context through dynamic clustering. The user can then select songs in the playlist that in his opinion do not fit for the current context-of-use. From this preference feedback, new feature weights in the underlying similarity measure are derived by an inductive learning algorithm based on the construction of a decision tree that

[1] For an online demo visit `http://mpeer.dfki.de`

uncovers the feature values classifying songs into the categories "preferred" and "rejected". Though the basic idea to learn from user feedback is indeed very similar to our approach, the usage scenario and the adaption algorithm are completely different: PATS (and the former systems) aims to generate lists of similar songs given one or more seed songs. In contrast to that, our goal is to structure a whole collection of songs. Hence, we do not classify songs as belonging or not belonging to a playlist but need to assign songs to cells of a self-organizing map (which can be regarded as a 1-of-n classification with each cell as a class).

Another adaptive system for playlist generation called PAPA (Psychology and Purpose-Aware Automatic Playlist Generation) [10] uses sensors that measure certain bio-signals (such as the pulse) as immediate feedback for the music currently played. This information is then used to learn which characteristics of music have a certain effect on the user. Based on this continuously adapting model playlists for certain purposes can be created. Though this method of getting immediate feedback for continuous adaptation is highly interesting, it is not applicable in the usage scenario of our approach.

For similarity-based grouping and visualization we use a growing self-organizing map (SOM) approach that has been already successfully applied for user-adaptive text, image and video retrieval [8, 9, 2]. Other MIR/MDL systems that utilize similar techniques comprise the SOM-enhanced Jukebox (SOMeJB) [14], the "Islands of Music" [12, 11], the MusicMiner [6] and the PlaySOM- and PocketSOM-Player [7], the latter being a special interface for mobile devices. However, in contrast to our approach, none of these systems is based on an adaptive similarity measure.

## 3 Prototype

We have built a prototype that organizes music on song level to demonstrate our adaptation approach. The system is tested on the Beatles corpus containing 282 songs of The Beatles. Though this corpus is rather homogeneous compared to other collections cointaining songs of several artists and genres, it has some advantages: Firstly, the Beatles are well-studied and have already been a subject of MIR research for some time. Secondly, 180 songs (all songs from the 12 official albums) have been manually annotated with chord label at the Queen Mary University, London, and finally, there exists a vast amount of meta-data for the Beatles songs on the web. The following sections cover specific details of the system.

### 3.1 Features

For each song in the corpus, a set of features is extracted describing several facets of music similarity.[2]

[2] Throughout this paper, we use the term facet for a specific similarity that can be computed on a single feature or a combination of features. It is possible to derive

***Audio information:*** Audio data was available for 200 songs.[3] From these songs, audio features were extracted utilizing the capabilities of the frameworks CoMIRVA [17] and JAudio [5]. Currently, CoMIRVA supports two features representing the overall timbre as Gaussian Mixture Models of the Mel Frequency Cepstral Coefficients (MFCCs) according to [1] and [4] and a feature that describes how strong and fast beats are played within specific frequency bands [12]. These complex features have specific distance measures that are not bounded by 1. They were transformed into similarity measures with values in $(0, 1]$ as follows:

$$sim(a, b) = (dist(a, b) + 1)^{-1} \tag{1}$$

JAudio was used to extract a global audio descriptor as described in [19]. The similarity of the resulting 64-dimensional real number vectors that can be interpreted as a perceptual hashes is computed by using the euclidean distance.

***Harmonic information:*** For the 180 songs from the official albums, the available ground truth chord labels were a basis for further feature computation. [4] From the chord labels, a histogram was built, mapping all chords onto the basic major and minor versions. For this histogram, two similarities were computed: Firstly, the cosine similarity from the respective (relative) chord frequency vectors, $v_1$ and $v_2$, defined as

$$sim(v_1, v_2) = \arccos \frac{v_1 \cdot v_2}{\|v_1\| \|v_2\|} \tag{2}$$

and secondly the Jaccard similarity of the sets of chords used (i.e. the non-empty histogram bins). For two sets, $S_1$ and $S_2$, it is defined as the size of the intersection divided by the size of the union:

$$sim(S_1, S_2) = \frac{S_1 \cap S_2}{S_1 \cup S_2} \tag{3}$$

As another feature, the tonal key was used as stated in Alan W. Pollack notes on The Beatles.[5] Again, the key is supposed to be automatically detected together with the chord labels in the next version of the system. The similarity for the key was computed with a modified version of the Jaccard similarity on the set of tones belonging to the key giving extra weight onto the more important ones.

***Production information:*** As e.g. pointed out in [20], the production process plays an important role in defining the sound of a recording. We semi-automatically extracted such information from wikipedia[6] articles on the songs,

several facets from the same feature(s) as different similarities may be computed as e.g. for the chord histogram. However, if not stated explicitly, there is always one facet/similarity for each feature.

[3] The audio data set comprises all songs from the 12 official albums, the Magical Mystery Tour EP, the Yellow Submarine soundtrack, and the "Blue', "Red" and "1" compilation.

[4] As such data is usually not available and can only be generated at high costs, we are currently incorporating an automatic chord recognizer into the system.

[5] `http://www.icce.rug.nl/~soundscapes/DATABASES/ AWP/awp-notes_on.shtml`

[6] `http://en.wikipedia.org`

comprising: creator(s) / composer(s), producer(s) and recording engineer(s), recording location(s) (usually a studio), recording date(s), (guest) musicians and instruments. Additionally, the musicians were further divided regarding their instruments into subgroups for lead-vocals, background-vocals, guitars, bass and drums/percussion. All these features are set-based and compared by using the Jaccard similarity, except for the date where a more sophisticated similarity measure was used to differentiate between (partially or fully) overlapping, adjacent and independent time spans. Furthermore, the year of the first release was used as simple feature. For two values of the year, $y_1$ and $y_2$, the similarity is defined as:

$$sim(y_1, y_2) = \max\left\{0,\ 1 - \frac{|y_1 - y_2|}{3}\right\} \tag{4}$$

This resembles a fuzzy membership function for $y_1$ with a triangular shape [22].

***Textual information:*** Lyrics for all songs were obtained through the web service of LyricWiki[7], filtered for stop words, stemmed and described by document vectors with TFxIDF term weights [16]. As similarity measure, the cosine similarity was used. In the same way, song and album titles were processed and compared.

Finally, album covers were obtained through web search and linked to the songs. However, they are currently only used for labeling as described in Section 3.3.

### 3.2 Grouping

Based on the facet similarities described in the preeceding section, we can define a combined similarity measure that is computed as a weighted sum. The sum of all facet weights is always 1.0. Initially, all weights are equal and later subject to adaption by the method discussed in Section 3.4.

Using this intial similarity measure, songs are clustered by the growing self-organizing map approach presented in [9]. The algorithm starts with a small initial grid composed of hexagon cells, where each hexagon refers to a cluster and is represented by a randomly initialized prototype. Each song is then assigned to the most similar cluster in the grid resulting in an update of the respective prototype and to some extend of the surrounding prototypes (depending on a neighborhood function). Having all songs assigned to the grid, the inner cluster distance can be computed for each cluster. If it exceeds some predefined threshold, the respective cluster is split resulting in a grown map. This process is repeated until no more cells need to or can be split or an empty cell occurs (i.e. a cluster with no assigned songs). It results in a two-dimensional topology preserving the neighborhood relations of the high dimensional feature space. I.e. not only songs within the same cluster are similar to each other but also songs of clusters in the neighborhood are expected to be more similar than those in more distant clusters. Using a growing approach ensures that only as many clusters are created as are actually needed.

[7] `http://lyricwiki.org`


**Fig. 1.** Screenshot of the grid in *Cluster-Size-Mode* and two cell content windows.

### 3.3 Presentation / User Interface

Having generated a hexagonal grid, it's cells can be seen as "virtual folders", each one containing a set of similar songs. A screenshot of the prototype user interface is shown in Fig. 1.[8] Cells are labelled with the album cover(s) that are most frequently linked with the songs contained in the cell. In the default view, only one cover is shown. At higher zoom level, at most four covers are displayed depending on how many different covers are linked with the songs in the cell. Further, the cell background can be colored providing different information:

1. Cluster-Size-Mode:
   The brightness of the cell color on an emerald color scale refers to the size of the respective cluster, where small cells are darker. This is the default mode (see Fig. 1).
2. Cluster-Error-Mode:
   This mode is by default chosen during the initial map learning phase. Cells are colored red with brightness referring to the internal cluster error, i.e. the sum of the distances of the prototype with all songs in the cluster.
3. No color:
   In this mode, the cells are not colored and only the covers are shown.

If the mouse is over a specific cell, this cell is visually enlarged and after some delay, a tooltip with a list of contained songs is displayed. Clicking on a cell opens a window that displays the content of the respective cluster. For each song, the title and (if available) the CD covers are listed.

[8] A demo video is available at `http://www.dke-research.de/aucoma/`

### 3.4 User-Adaptivity

The algorithm outlined in Section 3.2 works in an unsupervised manner. It depends only on the choice of the features for representation and the initial similarity measure how songs are grouped. However, the user has often a better intuition of a "correct" clustering. In cases where he does not agree on the cluster assignment for a specific song, he can change it by dragging the song from a cell's content window and drop it onto a different cell of the grid or another content window. Such an action does immediately result in a modification of the underlying facet weighting scheme and ultimately to a reassignment of all songs to the grid. This way, manually moving a single song may cause automatic movement of several others. The idea of the approach is that, while the user is interacting with the system nn an iterative process of user action and resulting adaptation, the similarity measure more and more converges towards to the (possibly only subconsciously existing) preferences of the user.

For the computation of the new facet weights, a quadratic optimization scheme is used that has been introduced in [18] where it was evaluated on text data. In the following, this method is briefly summarized. The formalization has been adapted to incorporate the concept of facet similarities as introduced in the context of this work.

The similarity $sim(a, b)$ of a pair of objects, $a$ and $b$, is computed as:

$$sim(a, b) = \sum_{i=1}^{n} w_i \cdot sim_i(a, b) \tag{5}$$

where $n$ is the number of facets, $sim_i(a, b)$ is the $i$-th facet similarity of $a$ and $b$ and $w_i$ the respective facet weight. The facet weights are required to be non-negative and the sum of all weights has to be 1:

$$w_i \geq 0 \qquad \forall 1 \geq i \geq n \tag{6}$$

$$\sum_{i=1}^{n} w_i = 1 \tag{7}$$

The problem of dragging objects as described above can be mapped to a problem of cluster reassignement: Assume a object $o$ that is assigned to a cluster $c_s$, i.e. for the similarity holds

$$sim(c_s, o) > sim(c, o) \qquad \forall c \neq c_s \tag{8}$$

If $d$ is dragged by a user to a cluster $c_t$ the underlying facet weights have to be adapted such that now

$$sim(c_t, o) > sim(c, o) \qquad \forall c \neq c_t \tag{9}$$

holds. If the user has already reassigned some object(s), a similar constraint has to be defined for each object to ensure that it stays at its position. Note

that the change of the facet weights should be as small as possible in order to avoid too abrupt changes of the cluster assignments. Therefore, it is additionally demanded that the sum over all (quadratic) deviations of the weights from their initial value $\frac{1}{n}$ should be minimal:

$$\min_{(w_1,\ldots,w_n)\in R^n} \sum_{i=1}^{n} \left( w_i - \frac{1}{n} \right)^2 \tag{10}$$

This can be used as the objective function for a quadratic optimization scheme and equations 6, 7 and 9 give the additional constraints.

Additionally to the above cluster reassignment scenario, we want to motivate another way of incorporating user feedback that can be as well integrated into the quadratic optimization scheme. The following scenario is considered: The user may query the system with a seed song. As a result, the system will display the 10 most similar songs to this seed according to its current weighting scheme. Form the user's point of view, this list may not be in the correct order. It may even not contain any of the songs the user would rank amongst the 10 most similar ones (if he would know all the songs in the collection). Consequently, the user may want to change the order of the songs.[9] From a single re-ranking of a song, additional constraints for the quadratic optimization can be derived as follows: Let $s$ be the seed song, $i$ be the original rank and $j \neq i$ be the modified rank of the re-ranked song $d$ within the retrieved top-10 list. If $j$ is less than $i$, it can be concluded that all songs with original rank $j \leq r < i$ are less similar than $d$ and that the song at rank $j-1$ (if such exists) is more similar than $d$ with respect to the seed song $s$. In the other case, i.e. if $j$ is greater than $i$, all songs with original rank $i < r \leq j$ are more similar than $d$ and the song at rank $j+1$ is less similar than $d$ with respect to the seed song $s$. Every relation "song $a$ is more similar than song $b$ with respect to seed song $s$" can then be expressed by the constraint:

$$sim(a, s) > sim(b, s) \tag{11}$$

Note the important difference with the constraint defined in Equation 10: Here, it is referred to objects instead of clusters.

## 4 Experiments

In order to objectively evaluate the usefulness of the system, we have simulated user interaction with the system as motivated and described in [18]. The basic idea is to assume a fixed random weighting scheme that the simulated user has "in mind". According to such a user model, some objects may be misplaced on the grid. Simulated interaction of the user should result in better adapted similarity measure of the system and thus in a decrease of the number of misplaced

[9] Such an action, e.g. through drag & drop on the ranked list, has not yet been incorporated into the graphical user interface. Currently, it can only be carried out through the simulation interface for evaluation.

objects. As proposed in [18], we use the *average top-10 precision* of the similarity measure for evaluation: A ranked top-10 list of similar objects is computed for each object according to the system's current similarity measure and compared with a top-10 list according to the preferences of the simulated user. The percentage of matches is computed – ignoring the order of the ranking – and averaged over all objects.

Apart from the average top-10 precision used in the earlier experiments, we also take the number of misplaced objects and the *object postion error (OPE)* as measures for the difficulty of the adaptation problem and the performance of the algorithm into account. The latter is defined as the euclidean distance of the currently assigned cluster $c(d) = \left(x_{c(d)}, y_{c(d)}\right)$ to the correct one $t(d) = \left(x_{t(d)}, y_{t(d)}\right)$ on the cell grid, summed over all objects $d$ in the collection $D$, i.e.

$$OPE = \sum_{d \in D} \sqrt{\left(x_{t(d)} - x_{c(d)}\right)^2 + \left(y_{t(d)} - y_{c(d)}\right)^2} \tag{12}$$

We conducted two experiments – the first to verify that the adaptation approach through object repositioning is applicable in this domain, and the second to assess the usefulness of the newly introduced re-ranking constraints.

### 4.1 Experiment 1: object repositioning

In the first experiment, we measured the performance of the adaption method for a simulated user that moves misplaced objects according to his similarity measure. As modification of the simulations in [18], we used a simpler strategy for selection of the object to be moved: In each iteration, the set of misplaced objects is determined and one object of this set is randomly selected to be moved to its correct position. For the user's similarity measure, we consider the following three scenarios:

1. Fully random weights
2. Random binary weights (the weight vector is initialized as a random binary vector and normalized afterwards)
3. Single facet only (the weights for all facets except for a single random facet are zero)

Each scenario was tested on 10 self-organizing maps (learnt on the same data but with different random initializations) for 5 users (random weight vectors) and 5 simulations each (different objects are chosen to be moved manually). A simulation terminated when there were no more misplaced objects on the map. The results of this experiment are shown in Table 1.

The initial values for average top-10 precision, number of misplaced objects and object position error can be interpreted as indicators for the difficulty of the restructuring problem. The uniform weight vector initially assumed by the system and used for training of the map comes closest to a fully random weight vector (1). However, the average top-10 precision already drops by 30% for this

**Table 1.** Results of the first experiment, simulating a user that moves misplaced objects. Mean (variance) computed over 250 random runs (10 maps × 5 users × 5 simulations) for each scenario of user weight initialization.

| scenario (user weights) | (1) fully random | (2) random binary | (3) single facet |
|---|---|---|---|
| initial average top-10 precision | 0.70 (0.01) | 0.55 (0.01) | 0.11 (0.01) |
| initially misplaced objects | 4.87 (2.01) | 11.42 (2.99) | 239.63 (7.78) |
| initial object position error | 7.33 (3.39) | 17.97 (5.52) | 638.32 (42.13) |
| final average top-10 precision | 0.70 (0.01) | 0.57 (0.02) | 0.89 (0.04) |
| number of iterations | 4.76 (1.94) | 11.40 (2.94) | 60.21 (11.08) |

scenario. It drops even further for the scenario using random binary vectors (2) and for single-facet weight vectors (3) it is only 11% resulting in a high number of misplaced objects (about 240 out of 282). For the other scenarios, (1) and (2), the number of initially misplaced objects is surprisingly small. However, as the final value for the average top-10 precision and the number of iterations show, there is not much improvement during the few iterations of scenarios (1) and (2). The few misplaced objects in these scenarios are almost completely moved manually by the user. The weight adaptations resulting from the simulated actions have not a big impact. For scenario (3) – which appears to be the hardest one – far more iterations are necessary but the final average top-10 precision is increased to 89% – the best value throughout all scenarios. Interestingly, in all scenarios the average top-10 precision would never come even close to 100%, even though all objects finally were at their correct position. The explanation for this "glass ceiling" effect lies in the nature of the self-organizing map. All objects are assigned to the correct cluster, as long as there is no other cluster that has a higher similarity (cf. equation 9). Regarding only constraints of this type[10], there may be many valid weighting schemes, the weighting scheme of the simulated user obviously being one of them. However, given only the user actions, any of the weighting schemes in the solution space could be the one of the user. The system has just not enough evidence, to decide which one is the right one, and chooses the one that minimizes the objective function (cf. equation 10). Especially with a large solution space, the chosen solution may significantly differ from the real one which results in the "glass ceiling" for the average top-10 precision. It is important to note, that this problem cannot be overcome by using a different adaptation strategy. As this effect was not evident in the experiment on text data in [18], it can be assumed, that because of the different nature of text data (i.e. high-dimensional but sparse vectors) and possibly because of the additional random data which was not added here, the solution space in these experiments must have been significantly smaller.

[10] The other constrains given by equations 6, 7 can be neglected here as they can be met just by normalizing the weights.

### 4.2 Experiment 2: object re-ranking

The second experiment aimed to assess the usefulness of the constraints derived from re-ranking a song within a retrieved top-10 list of the system. Note, that using such constraints directly optimizes the average top-10 precision and does only indirectly reduce the number of misplaced objects through the adapted similarity measure. Consequently, a significantly higher number of iterations was necessary in these simulations, independently from the scenario of user weight initialization. An additional reason for this could be that the constraints derived from re-ranking may not contain as much information as the constraints from object repositioning – i.e. they may be not as restrictive. Thus, it takes longer for the system's facet weights to converge on the ones of the user. In the worst case (weight initialization as in scenario (1) of the first experiment), it took about 280 iterations until no more objects were misplaced. Figure 2 shows the average top-10 precision and the object position error for this simulation run. Interestingly, the average top-10 precision decreased during the first iterations – possibly mislead by the few constraints derived so far. Only after 46 iterations (244 constraints[11]) the baseline average top-10 precision of the initial solution (uniform weights) is crossed. An average top-10 precision of 99% is reached after 199 iterations (949 constraints). However, at this point, there are still 16 misplaced objects.

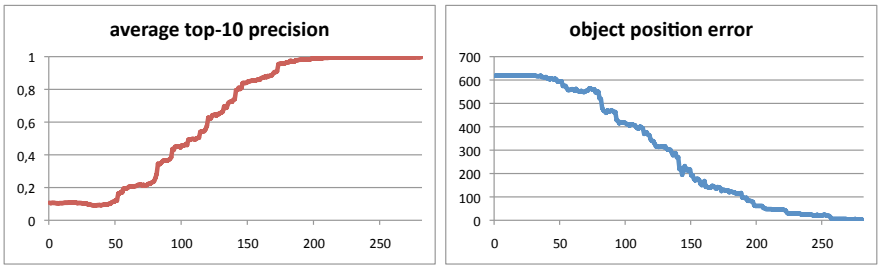


**Fig. 2.** Average top-10 precision and object position error for a simulation of adaptation through re-ranking constraints.

As this experiment shows, using the re-ranking constraints allows better adaptation to the user. However this adaptation comes at significantly higher costs for the user because it requires far more manual actions. Combining both approaches – moving and re-ranking of objects – may be a way to benefit from their specific advantages. This has to be analyzed in further experiments, possibly with real users to find out, which actions are preferred and perceived as being easier.

[11] As described in Section 3.4, multiple constraints in the form of equation 11 can be derived from re-ranking a single song.

## 5 Future Work

The prototype presented here is in an early stage of development and many aspects, especially concerning the automatic feature extraction leave room for improvement. An automatic chord recognizer based on the approach, we describe in [15] will soon be integrated into the system to replace manual annotations. For measuring the harmonicity of a song, possible further refinements are currently elaborated such as using the kurtosis ("peakiness") of an adequately aligned chord histogram as a measure for harmonic simplicity. Regarding the features currently used for facet similarity computation, there are large differences in complexity. The lyrics are for instance a very complex feature as opposed to the year. Possibly, facets referring to complex feature could be subdivided into sub-facets. This would lead to hierarchical similarity measures and pose new challenges for the adaptation method but at the same time possibly add more degrees of freedom. Regarding the initial weighting scheme, there may be much better initializations than just weighting all facets equally. Further, we are investigating other options of similarity based collection visualizations where the quadratic optimization method could also be applied.

## 6 Conclusion

We presented a prototype systems for structuring and exploring music collections that adapts to a user's way of comparing songs by learning from his interaction with the system. The system considers about 20 facets covering e.g. sound, harmonics, lyrics and information about the production process. These facets are combined in a complex similarity measure. For learning user specific facet weights, we extended an existing quadratic optimization approach that has previously been applied in the context of adaptive text retrieval. In a user simulation experiment, we verified that the system is able to approximate a similarity measure according to some unknown user preferences.

## 7 Acknowledgements

This work is supported by the German Research Foundation (DFG) and the German National Merit Foundation. We further would like to thank the developers of CoMIRVA [17] and JAudio [5] for providing their feature extractor code and George Tzanetakis for answering lots of questions concerning his MIREX 07 system [19]. Christopher Harte kindly shared his Beatles chord annotations with us.

## References

1. J.-J. Aucouturier and F. Pachet. Improving timbre similarity: How high is the sky? *Journal of Negative Results in Speech and Audio Sciences*, 1(1), 2004.

2. T. Bärecke, E. Kijak, A. Nürnberger, and M. Detyniecki. Video navigation based on self-organizing maps. In *Proc. of 5th Int. Conf. on Image and Video Retrieval (CIVR'06)*, 2006.
3. S. Baumann and J. Halloran. An ecological approach to multimodal subjective music similarity perception. In *Proc. of CIM'04*, 2004.
4. M. Mandel and D. Ellis. Song-level features and support vector machines for music classification. In *Proc. of ISMIR'05*, 2005.
5. D. McEnnis, C. McKay, I. Fujinaga, and P. Depalle. jAudio: An feature extraction library. In *Proc. of ISMIR'05*, 2005.
6. F. Mörchen, A. Ultsch, M. Nöcker, and C. Stamm. Databionic visualization of music collections according to perceptual distance. In *Proc. of ISMIR'05*, 2005.
7. R. Neumayer, M. Dittenbach, and A. Rauber. PlaySOM and PocketSOMPlayer, alternative interfaces to large music collections. In *Proc. of ISMIR'05*, 2005.
8. A. Nürnberger and M. Detyniecki. Weighted self-organizing maps - incorporating user feedback. In *Artificial Neural Networks and Neural Information Processing - ICANN/ICONIP 2003, Proc. of the joined 13th Int. Conf.*, 2003.
9. A. Nürnberger and A. Klose. Improving clustering and visualization of multimedia data using interactive user feedback. In *Proc. of the 9th Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'02)*, 2002.
10. N. Oliver and L. Kreger-Stickles. PAPA: Psychology and purpose-aware automatic playlist generation. In *Proc. of ISMIR'06*, 2006.
11. E. Pampalk, S. Dixon, and G. Widmer. Exploring music collections by browsing different views. In *Proc. of ISMIR'03*, 2003.
12. E. Pampalk, A. Rauber, and D. Merkl. Content-based organization and visualization of music archives. In *Proc. of ACM MULTIMEDIA'02*, 2002.
13. S. Pauws and B. Eggen. PATS: Realization and user evaluation of an automatic playlist generator. In *Proc. of ISMIR'02*, 2002.
14. A. Rauber, E. Pampalk, and D. Merkl. Using psycho-acoustic models and self-organizing maps to create a hierarchical structuring of music by musical styles. In *Proc. of ISMIR'02*, 2002.
15. J. Reinhard, S. Stober, and A. Nürnberger. Enhancing chord classification through neighbourhood histograms. In *Proc. of the 6th International Workshop on Content-Based Multimedia Indexing (CBMI 2008)*, 2008.
16. G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.
17. M. Schedl. The CoMIRVA Toolkit for Visualizing Music-Related Data. Technical report, Johannes Kepler University Linz, 2006.
18. S. Stober and A. Nürnberger. User modelling for interactive user-adaptive collection structuring. In *Proc, of AMR'07*, 2008.
19. G. Tzanetakis. Marsyas submission to MIREX 2007. In *Proc. of ISMIR'07*, 2007.
20. G. Tzanetakis, R. Jones, and K. McNally. Stereo panning features for classifying recording production style. In *Proc. of ISMIR'07*, 2007.
21. F. Vignoli and S. Pauws. A music retrieval system based on user driven similarity and its evaluation. In *Proc. of ISMIR'05*, 2005.
22. L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.