Everything in its right place?
Learning a user's view of a music collection

Korinna Bade, Andreas Nürnberger, Sebastian Stober
Data & Knowledge Engineering Group, Otto-von-Guericke-University Magdeburg, D-39106 Magdeburg, Germany
Email: {korinna.bade,andreas.nuernberger,sebastian.stober}@ovgu.de

Abstract

Keeping one's personal music collections well organized can be a very tedious task. Fortunately, today, many popular music players (such as AmaroK or iTunes) have an integrated library function that can automatically rename and tag music files and sort them into subdirectories. However, their common approach to stick with some hierarchy of genre, artist name, and album title barely represents the way a user would structure his collection manually. When it comes to organizing a music collection according to a user-specific hierarchy, three things are required: First, the music files have to be described by appropriate features beyond simple meta-tags. This includes content-based analysis but also incorporation of external information sources such as the web. Second, knowledge about the user's structuring preferences must be available. And third, and most importantly, methods for learning personalized hierarchies that can integrate this knowledge are needed. We propose for this task a hierarchical constraint based clustering approach that can weight the importance of different features according to the user perceived similarity. A hierarchy that is built based on this similarity measure reflects a user's view on the collection.

Introduction

One of the big challenges of computer science in the 21st century is the digital media explosion. Steadily growing hard-drives are filled with personal music collections. With increasing collection size maintenance becomes a more and more tedious task, but without manual organization effort it gets harder to access specific pieces of music or even to keep an overview. Typically, a large portion of the digital content is just "collecting dust" because the user has simply forgotten about it. Commonly supported search methods that rely on simple meta-data will not help to solve this problem as they require that the user can specify what he is looking for and further that the music is appropriately tagged. Whilst the latter can be automatized to some extend, the former obviously cannot be assumed when the user is looking for music he has forgotten about. Here, methods are needed that can help to improve awareness and accessibility of such data.

Automatic structuring is one means to ease access to music collections, be it for organization or for exploration. Moreover, users would greatly benefit if a system would not just structure the collection for easier access but would provide a structure that is intuitively understandable for the individual user since it is adapted to personal preferences and needs. Unfortunately, such aspects of individualization have been only a minor issue of research in the field of music information retrieval so far.

In this paper, we describe an approach that can adapt a hierarchical structuring of a music collection to better reflect user specific preferences. This takes into account that individual users handle the same data differently, having different views and a distinct focus. To achieve this goal, the first important step is to model a user's structuring preferences. We present here an approach based on constraints. Subsequently, different ways are shown of how such preferences can be integrated into a hierarchical agglomerative clustering approach in order to obtain a personalized clustering. Finally, we discuss possible generalizations and limitations of the proposed approach – especially with respect to application on music data. To begin with, we start with discussing related work for structuring music data in the following section, before our approach is described.

Related Work

The currently most common approach to structuring music collections is to rely on meta-data – typically the artist, album, year, and title tag – sometimes also genre. With this information, tracks can be sorted into a predefined directory structure. An alternative way is to use a genre hierarchy as predefined structure and organize the tracks according to an automatic genre classification [3]. In contrast to these approaches, the one presented here does not rely on a predefined structure but organizes tracks solely based on their similarity.

Current work on structuring of music collections by similarity is primarily focussed on flat structures. Concerning hierarchical structures, some approaches exist that rely on a growing hierarchical self-organizing map (GHSOM) [6]: The SOM-enhanced Jukebox (SOMeJB) presented in [7] uses a GHSOM to structure a music collection hierarchically by music style. For the computation of song similarities, psycho-acoustic models are applied. Another system that relies on a GHSOM structuring approach is described in [5]. Here, the collection is structured on the artist level instead of individual songs. The similarity of artists is computed using a web search engine and standard text retrieval techniques.[1] The system presented in [4] applies the original GHSOM

[1] Online demo at `http://www.ofai.at/~elias.pampalk/wa/`

approach for structuring the music collection based on audio features. Additionally, web-based text information is used to determine a representative prototype track for each of the resulting clusters.

These GHSOM-based approaches are similar to the one described here in that they automatically construct a hierarchy for a music collection. However, the clustering method chosen here is significantly different (e.g. it is not prototype-based). Further, the mentioned systems are lacking the ability to adapt the structuring according to user preferences.

Modeling Structuring Preferences

In order to constrain the structuring process and influence the final clustering result, structuring preferences of an individual user need to be modeled in a way that allows for their integration into the automatic clustering method. A common approach introduced by Wagstaff et al. [11] uses two types of pairwise constraints that define the relation of two items: must-link and cannot-link constraints. Must-link constraints require the two items to be in the same cluster whereas cannot-link constraints identify pairs of items that have to be assigned to different clusters.

However, such constraints were introduced in the context of flat clustering approaches. When it comes to building cluster hierarchies their absolute formalization of constraints on cluster assignment (i.e., the items are either in the same or a different cluster) is no longer appropriate [10] because items are linked over different hierarchy levels. On lower levels, two items might be separated, but on higher levels, they could belong together. In order to express hierarchical relations, we proposed must-link-before (MLB) constraints [1, 2] with the goal to stress the hierarchical order in which items are linked. To achieve this, item pairs are replaced with triples expressing relative relationships according to hierarchy levels: The constraint $MLB_{xyz} = (i_x, i_y, i_z)$ states that items i_x and i_y should be linked on a lower hierarchy level than items i_x and i_z as shown in Figure 1.

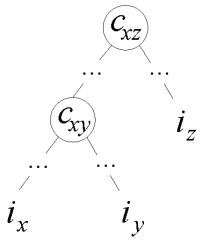

Figure 1: The MLB constraint (i_x, i_y, i_z) in the hierarchy

MLB constraints can be easily extracted from different sources: In [2] we describe how MLB constraints can be derived from labeled data. For the application scenario of this paper, this could be music files previously organized in a folder hierarchy by the user. Such a hierarchy already inherently describes some structuring preferences of the user and should definitely be used. As an example, consider a DJ's music collection that may (partially) be structured as shown in Figure 2. The following constraints reflect the hierarchical relation between the tracks in the example: $(track_2, track_3, track_1)$, $(track_2, track_3, track_4)$, $(track_2, track_1, track_4)$, and $(track_3, track_1, track_4)$. However, as a manually created hierarchy might not always be available or detailed enough, the following alternatives for collection of MLB constraints exist: In an active learning setting, the user is directly asked to specify the hierarchical relationship of an item triple whereas in an interactive setting, the user alters an initially un-personalized structuring and thereby generates constraints. The latter approach is discussed in [9] for flat clusterings of music files.

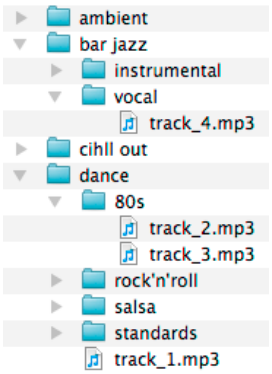

Figure 2: Example of a manually created music hierarchy.

Personalized Clustering

There are two fundamentally different methods to integrate constraints into clustering. First, the individual constraints can be directly integrated into the clustering procedure. This is also called instance-based constrained clustering and can be considered a lazy learning approach. Second, the constraints can be used to adapt the similarity measure that will be applied during clustering. The goal of this approach is to generalize the individual constraints to get a broader formalization of the user's preferences through a personalized similarity measure. For instance in the example above, rhythm and tempo might be more relevant to the DJ than the lyrics or the artist. In this case, rhythm- and tempo-related features would be emphasized by the personalized similarity measure.

In the following, the approaches and their applicability to music data are briefly presented. A more thorough discussion and an evaluation based on text data can be found in [2]. The following approaches are all based on hierarchical agglomerative clustering (HAC), which produces a dendrogram, i.e., a hierarchical tree structure on the data. The goal is to create a dendrogram, which reflects the user's structuring preferences, i.e., the MLB constraints, as much as possible. Since HAC builds the dendrogram by starting with each item in an individual cluster and then repeatedly merges the two closest clusters until a single cluster is left, the only prerequisite for applying HAC is to have a similarity measure defined over the set of items. Hence, any similarity measure on audio data can be used for applying

HAC to music tracks.

Instance-based Constrained Clustering

In comparison to must-link and cannot-link constraints, MLB constraints can be easily integrated into hierarchical clustering. For HAC, the constraints are integrated in the merging step such that merges occur only in accordance with the given MLB constraints (or at least with most of the constraints). This is achieved by merging the two closest clusters from the set of possible merges that do not violate any (or that violate the fewest) given constraints. This means that for all constraints (i_x, i_y, i_z), the cluster containing i_z can only be merged with a cluster that either contains both, i_x and i_y, or neither.

Please note that MLB constraints are item triples and do not pose any restrictions on the item representation. Hence, the instance-based constrained clustering approach can be applied to any kind of data including audio data. Of course, a similarity measure is still required for clustering.

Metric Learning

A personalized similarity measure can be learned in two ways: either independent of the clustering [2] or during clustering [1]. While the former allows to learn the similarity measure off-line, the latter can take into account "unlabeled" data, i.e., items that are not part of any constraint. In the following, we will briefly outline an approach for the first case.

This approach is of course metric dependent. Here, we use a common approach from machine learning, which is based on the representation of the raw data through a numerical feature vector. Then, similarity of the items is computed with the cosine similarity between the respective vectors. Personalization is achieved by weighting the individual dimensions of the vector space according to user specific preferences. This leads to the following adapted cosine similarity of two items:

$$\mathrm{sim}_{\vec{w}}(i_x, i_y) = \frac{\sum_{j=1}^{n} w_j i_{x,j} i_{y,j}}{|\vec{i}_x|_{\vec{w}} |\vec{i}_y|_{\vec{w}}} \tag{1}$$

$$\text{with } |\vec{i}_x|_{\vec{w}} = \sqrt{\sum_{j=1}^{n} w_j i_{x,j}^2},$$

n being the number of dimensions, $i_{x,j}$ being the j-th component of the original item vector, and w_j being the weight of feature j with $w_j \geq 0$, and $\sum_{j=1}^{n} w_j = n$. The first condition makes the solution meaningful, the second avoids extreme case solutions. The valid weighting of 1 for all features reflects the standard similarity.

The weights are learned using a gradient descent search. During learning of the weights, all constraint triples (i_x, i_y, i_z) are presented to the algorithm several times. If the constraint is violated by the current similarity measure, the weighting is updated. A constraint is thereby interpreted as a relation between item similarities:

$$(i_x, i_y, i_z) \rightarrow (\mathrm{sim}(i_x, i_y) > \mathrm{sim}(i_x, i_z)). \tag{2}$$

This means a constraint is violated, if the inequality is violated. As HAC clusters more similar items first, obeying the similarity relations should improve overall cluster quality. However, this depends on the computation of cluster similarities for clusters containing more than one item (linkage method). The impact of the individual item gets smaller the higher the level in the dendrogram.

The goal of the gradient descent is to minimize the error, i.e., the number of constraint violations. This can be achieved by maximizing

$$obj_{xyz} = \mathrm{sim}_{\vec{w}}(i_x, i_y) - \mathrm{sim}_{\vec{w}}(i_x, i_z) \tag{3}$$

for each (violated) constraint. This leads to the weight update rule

$$w_j = w_j + \eta \Delta w_j = w_j + \eta \frac{\partial obj_{xyz}}{\partial w_j} \tag{4}$$

where η is the learning rate defining the step width of each adaptation step. The final computation of Δw_j after differentiation is:

$$\begin{aligned} \Delta w_j = \bar{i}_{x,j}(\bar{i}_{y,j} - \bar{i}_{z,j}) - \frac{1}{2}\mathrm{sim}_{\vec{w}}(i_x, i_y)(\bar{i}_{x,j}^2 + \bar{i}_{y,j}^2) \\ + \frac{1}{2}\mathrm{sim}_{\vec{w}}(i_x, i_z)(\bar{i}_{x,j}^2 + \bar{i}_{z,j}^2) \end{aligned} \tag{5}$$

$$\text{with } \bar{i}_{x,j} = i_{x,j} / |i_x|_{\vec{w}}.$$

However, this computation does not ensure the bounds on w_j given earlier. To achieve this, an additional step is added that, first, sets all negative weights to 0 and then normalizes the weights to sum up to n.

The learned metric is then used for similarity computation during the clustering. Hence, the same metric can be used for clustering different data (based on the same representation) without re-learning. Furthermore, instance-based constrained clustering and metric learning can be nicely combined. In this case, the metric is learned first as just described. Afterwards, the instance-based method is applied using the modified similarity.

Discussion

In the following, we want to discuss the generality of the proposed approach. For a discussion of algorithm-specific details please refer to [2].

The approach presented here was described for the HAC algorithm. However, the fundamental ideas can easily be transfered to other hierarchical clustering methods. As an example, consider the instance-based approach and its application to hierarchical divisive clustering (HDC) like, e.g., GHSOM [6] or Bisecting k-Means [8]. As in HAC, the main goal is to ensure the existence of the intermediate cluster c_{xy} (cf. Figure 1). While for HAC this is achieved by prioritizing the merge of i_x and i_y,

HDC needs to take care of separating i_z from i_x and i_y before splitting the latter.

Furthermore, the metric learning approach was specifically demonstrated for a vector representation in combination with the cosine similarity. This can be generalized to a similarity measure that combines individual feature similarities. Commonly, this is realized through a sum. In this case an integration of feature weights could look like this:

$$\mathrm{sim}_{\vec{w}}(i_x, i_y) = \frac{\sum_{j=1}^{n} w_j \mathrm{sim_j}(i_{x,j}, i_{y,j})}{\sum_{j=1}^{n} w_j} \qquad (6)$$

Generally, any aggregating function could be applied instead of the sum as well. The only requirement for subsequent metric learning is differentiability. Equation 4 is applicable to any of these similarity measures. Naturally, the differentiation result (Equation 5) will be different respectively.

Most hierarchical clustering approaches build a hierarchy, where each item is directly assigned to a single cluster. Of course, an item is also indirectly contained in all superclusters. However, it may not be further assigned to any other cluster. Nevertheless, a user might actually associate a track with several clusters. The underlying reason can be twofold: First, the user might consider different aspects. For instance, the DJ in our example in Figure 2 might want to sort his tracks additionally into the common artist/album-hierarchy to easily handle artist-specific requests from the audience. This can be supported best by several (parallel) hierarchies, each describing a distinct view on the collection based on specific aspects. Each individual hierarchy can be learned separately with the presented approach. Second, a track could have two distinct characteristics regarding a single aspect. For instance, a track may belong to two genres with no dominant one. In such rare cases, the algorithm cannot help. In fact, it is not even clear, how a user would handle such a case manually and it is questionable, whether allowing multiple direct cluster assignments might really be helpful here. Most likely, it would rather confuse the user. Still, this remains an open question that should be studied more thoroughly.

Conclusions

In this paper, we outlined an approach that can adapt a hierarchical structuring of a music collection to better reflect user specific preferences modeled by constraints. Different ways are shown on how such constraints can be integrated into a hierarchical agglomerative clustering approach in order to obtain a personalized clustering. Further, we discussed how the described method can be generalized for application on other hierarchical clustering approaches and pointed out possible limitations.

Acknowledgements

This work is supported by the German Research Foundation (DFG) and the German National Merit Foundation.

References

[1] K. Bade and A. Nürnberger. Personalized hierarchical clustering. In *Proc. of the 2006 IEEE / WIC / ACM Intl. Conf. on Web Intelligence*, 2006.

[2] K. Bade and A. Nürnberger. Creating a cluster hierarchy under constraints of a partially known hierarchy. In *Proc. of the 2008 SIAM Intl. Conf. on Data Mining*, 2008.

[3] S. Brecheisen, H.-P. Kriegel, P. Kunath, and A. Pryakhin. Hierarchical genre classification for large music collections. In *Proc. of the 2006 IEEE Intl. Conf. on Multimedia and Expo (ICME'06)*, 2006.

[4] M. Dopler, M. Schedl, T. Pohle, and P. Knees. Accessing Music Collections via Representative Cluster Prototypes in a Hierarchical Organization Scheme. In *Proc. of the 9th Intl. Conf. on Music Information Retrieval (ISMIR'08)*, 2008.

[5] E. Pampalk, A. Flexer, and G. Widmer. Hierarchical organization and description of music collections at the artist level. In *Proc. of the 9th European Conf. on Research and Advanced Technology for Digital Libraries (ECDL'05)*, 2005.

[6] A. Rauber, D. Merkl, and M. Dittenbach. The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data. *IEEE Transactions on Neural Networks*, 13:1331–1341, 2002.

[7] 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 the 3rd Intl. Conf. on Music Information Retrieval (ISMIR'02)*, 2002.

[8] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. Technical Report 00-034, University of Minnesota, 2000.

[9] S. Stober and A. Nürnberger. Towards user-adaptive structuring and organization of music collections. In *Proc. of 6th Intl. Workshop on Adaptive Multimedia Retrieval (AMR'08)*, 2008.

[10] K. Wagstaff and C. Cardie. Clustering with instance-level constraints. In *Proc. of the 17th Intl. Conf. on Machine Learning*, 2000.

[11] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. In *Proc. of 18th Intl. Conf. on Machine Learning*, 2001.