Morik, Katharina; Mierswa, Ingo; Wurst, Michael

Working Paper

Structuring music collections by exploiting peers' processing

Technical Report, No. 2005,43

Provided in Cooperation with:
Collaborative Research Center 'Reduction of Complexity in Multivariate Data Structures' (SFB 475), University of Dortmund

Suggested Citation: Morik, Katharina; Mierswa, Ingo; Wurst, Michael (2005) : Structuring music collections by exploiting peers' processing, Technical Report, No. 2005,43, Universität Dortmund, Sonderforschungsbereich 475 - Komplexitätsreduktion in Multivariaten Datenstrukturen, Dortmund

This Version is available at:
https://hdl.handle.net/10419/22633

Structuring Music Collections by Exploiting Peers' Processing

Michael Wurst, Ingo Mierswa, Katharina Morik
Artificial Intelligence Unit, University of Dortmund
{wurst,mierswa,morik}@ls8.cs.uni-dortmund.de

Abstract

Music collections are structured in very different ways by different users. There is not one general taxonomy, but individual, user-specific structures exist. Most users appreciate some support in structuring their collection. A large variety of methods has been developed for textual collections. However, audio data are completely different. In this paper, we present a peer to peer scenario where a music collection is enhanced by exploiting peers' processing. Collaborative clustering structures a set of audio data in a node of the user's taxonomy by retrieving (partial) taxonomies of peers. In order to classify audio data into a taxonomy features need to be extracted. Adopting feature extraction to a particular set of classes is effective but not efficient. Hence, we propose again to exploit what has already been done. Well-suited feature extraction for one classification task is transferred to similar tasks using a new distance measure.

1 Introduction

Since the World Wide Web has become a personal source of documents, music plays, and videos, the need for structuring all these data has become demanding not only for companies but also for private persons. Huge archives like, e.g., iTunes offer music for download together with meta-data. This allows users to easily retrieve a certain music play from the archive and navigate within it. The downloaded music then forms the user's collection.

Creating the collection's structure and manually inserting the files into it is up to the single user. Where to some extent creating an own taxonomy for music is fun, it can become a burden if it is to be completed. Why not taking advantage of well developed parts of others? In a peer-to-peer or ad-hoc network users can share not only their songs but also the structure of their collections with other users. Imagine, one user has developed a taxonomy. He now recognizes that one of its nodes has become so large that it would be nice to split it. He may then broadcast the songs of this node to the neighbor peers, asking how they would structure this set of songs. He may receive different proposals from different peers. He selects the best and merges it with his own taxonomy. This bootstrapping approach works as long as different users have worked out different taxonomies, which is very likely. It would only fail, if all users focused on exactly the same structure and ask for structuring exactly the same remaining music plays, which is unlikely.

1.1 The scenario

This distributed approach is currently implemented by a student group at our university[1]. Figure 1 illustrates the exchange of songs and (partial) taxonomies among users in a peer-to-peer network. Each user owns a music collection (drawn as a cylinder) organized in some taxonomies (drawn as a box). The user at the left side of the figure broadcasts a set of unstructured songs, X_q, to his peers (at the right side of the figure). The peers look for parts of their own taxonomies which best fit the query. If a peer finds a good match, she returns into the response set the structure together with the music stored within it. The querying user selects the best taxonomy from the response set and integrates X_q into it. The result is merged into the user's collection.

In this scenario each peer's collection is enriched by the following services:

- Find a taxonomy which fits best a set of music plays.
- Combine taxonomies and integrate a set of music plays in the resulting structure.

[1]The Nemoz system can be visited under `http://nemoz.sourceforge.net`.

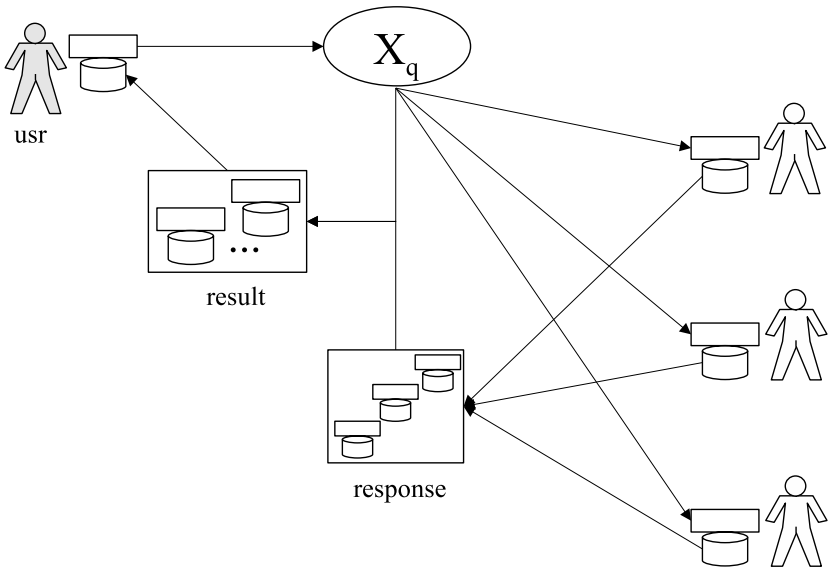

Figure 1: Collaborative structuring scenario where user `usr` sends an unstructured collection X_q of songs to the other peers and receives a structure for it.

1.2 The difficulty of the task

In principle, collections can be structured automatically using clustering algorithms. The advantage is that no taxonomy needs to be given. Hierarchical clustering builds-up a taxonomy for a given set of items based on a global similarity measure. However, if we look more closely into music collections, we encounter some problems.

First, music collections are organized in very different ways by different persons. A user study reports several organization principles found in music collections [17], among them the time of day or the situations in which the music is best listened to, the year in which a song has been the favorite, the recency of acquiring the music play. We asked our students to build-up taxonomies for the Garage Band collection of music which is now available as a benchmark dataset [15]. We received 39 taxonomies which vary a lot. Time of day, situations, mood, instruments, taste, and individually designed genres were used to structure the collection. There is not one general taxonomy, but individual, user-specific structures, which can hardly be predicted. This means, that standard clustering does not meet the users' requirements. There

is no global similarity measure. A clustering which is perfect according to a particular similarity measure might be considered inacceptable by another user. Hence, we cannot prepare our learning algorithm for a certain general task, but have to flexibly solve unforeseen tasks. The problem of diverse ad hoc clusterings becomes even more demanding because of the real-time requirement in a peer to peer network. There is to our best knowledge no standard algorithm which is capable of structuring audio data in real-time.

The first problem could show up in several domains, but audio data pose additional challenges. Our second problem is crucial for all music collections which are structured not only on the basis of the meta-data or song texts: it is hard to find a feature set which is valid for the overall collection. Using the raw audio data does not allow to cluster or classify music. Several features for different aspects have been manually designed [22, 43, 13, 40]. It has been shown, however, that no set of extracted features is well-suited for all classification tasks. For instance, clustering rock music requires completely different features than does clustering music according to the mood [30]. If there would exist one complete set of features, from which each learning task selects its proper part, the feature problem could be reduced to feature selection. However, there is no tractable feature set to select from. The number of possible feature extractions is so large - virtually infinite - that it would be intractable to enumerate it. Learning feature extraction for a particular classification task delivers good results [24, 44]. However, training the feature extraction is time-consuming and demands a sufficient set of examples.

We look for solutions to the difficult problem of structuring music collections following a bootstrapping approach. Some users manually structured part of their collection. We want to use this for enhancing the own structuring and have developed a new collaborative clustering algorithm. This clustering goes beyond the co-occurrence of favorite songs as employed by Amazon, in that it delivers a structure together with music plays integrated to it, instead of just returning a set of correlated songs as does the collaborative filtering of Amazon.

In the same spirit we try to approach the feature problem. Some audio data have already been used for learning. We want to transfer the feature extraction to similar learning tasks. We have developed a new similarity measure of learning tasks from audio data which is based on already learned feature weights.

The paper is organized as follows. First we introduce our notation for

music plays and taxonomies. In Section 3, we describe our approach to enhance the structure of the own collection by taxonomies of peers, the collaborative clustering. Experiments show that, indeed, collaborative clustering outperforms standard clustering methods. Section 4 describes our approach to exploit already performed learning in order to find a well-suited feature set for a certain learning task. Experiments show that this approach almost achieves the performance of learning feature extraction for a certain learning task. Furthermore, they show also, that the approach is fast and its results are better than merely using a fixed set of (weighted) features.

2 The notation

Before we can explain our methods, we introduce our notation.

Definition 1. *A* TAXONOMY C *is a tree given by the tuple*

$$\langle ID, top, Successors(C) \rangle$$

where

- *the ID is the label of the node,*
- *top are the music plays stored at the root node, and*
- *the Successors of C are defined by recursively calling the function $dirSuccessors(C_i)$ which is $\{\}$ if C_i is a node without successor (leaf node). Otherwise*

  $$dirSuccessors(C_i) = \left\{ C_j | C_i \succ C_j, \nexists C'_j \text{ s. t. } C'_j \succ C_j, C_i \succ C'_j \right\}$$

  using some partial ordering $\succ$.

Hence, each successor of a taxonomy is again a taxonomy and possibly stores objects. Note, that due to this definition objects are not only stored at the leaf nodes, but can be collected at intermediate nodes, as well. We

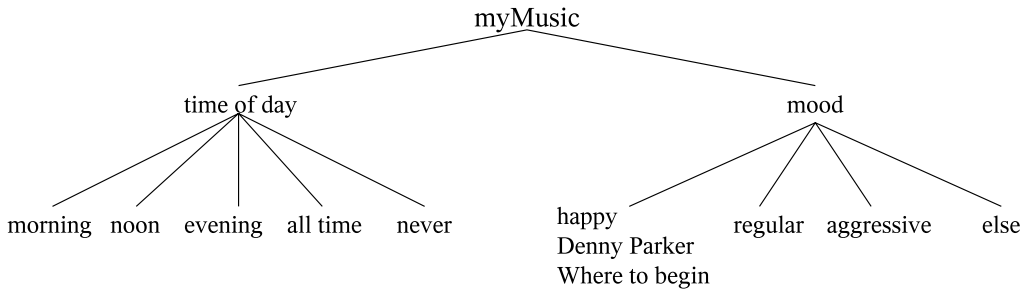

Figure 2: A taxonomy with just one song stored under the node "happy".

restrict taxonomies to trees. An example of a taxonomy is:

$$\begin{aligned}
&\langle myMusic, \{\}, \\
&\quad \langle timeofday, \{\}, \\
&\quad\quad \langle morning, \{\}, \{\}\rangle \\
&\quad\quad \langle noon, \{\}, \{\}\rangle \\
&\quad\quad \langle evening, \{\}, \{\}\rangle \\
&\quad\quad \langle alltime, \{\}, \{\}\rangle \\
&\quad\quad \langle never, \{\}, \{\}\rangle\rangle \\
&\quad \langle mood, \{\}, \\
&\quad\quad \langle happy, \{Denny_Porter_WhereToBegin\}, \{\}\rangle \\
&\quad\quad \langle regular, \{\}, \{\}\rangle \\
&\quad\quad \langle aggressive, \{\}, \{\}\rangle \\
&\quad\quad \langle else, \{\}, \{\}\rangle\rangle
\end{aligned}$$

Obviously, this taxonomy combines two different aspects under which music is organized, namely the time when to listen and the mood of the listener. It is an almost empty taxonomy, i.e., there is only one song stored within it. Figure 2 shows the same taxonomy.

Definition 2. *The* EXTENSION OF A TAXONOMY *is a mapping from the universe of taxonomies to the universe of music.* $ext(C_i) = top$ *if* C_i *has no successors, otherwise* $ext(C_i) = top \cup \bigcup ext(Successors(C_i))$.

Due to this definition, the extension of a sub-tree is entailed in the extension of the overall tree so that the regular meaning of subsumption is valid: If $C_i \succ C_j$, then $ext(C_j) \subseteq ext(C_i)$.

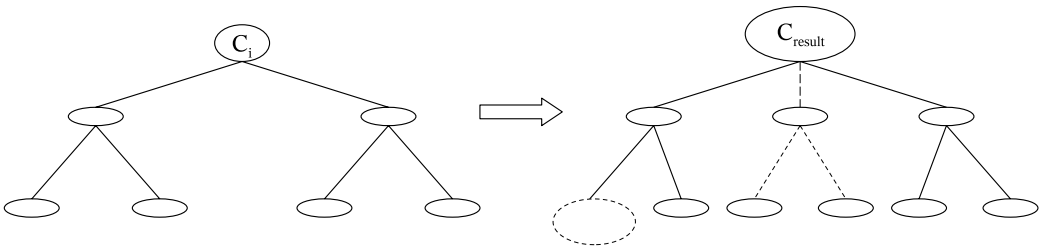

Figure 3: C_i is extended (dotted circle) and its structure (dotted lines with nodes) is enhanced. This leads to C_{result}.

A music play x is a vector of feature values.

Further on (Section 4), we need access to the feature itself and not to its value. We define F_{ij} as the $j-$th feature of taxonomy C_i. Since a feature is the set of all its possible values, it is considered a random variable. We define a weighting function $w : F_B \to \Re$ which assigns a weight to each feature from the set F_B. This gives us the vector w_i of feature weights at taxonomy C_i. The taxonomy is enriched by the features and their weights:

Definition 3. *The* ENRICHED MUSIC TAXONOMY *is defined by the tuple*

$$C = \langle ID, top, F, w, Successors(C) \rangle .$$

3 Collaborative clustering

Before presenting the Nemoz approach to collaborative clustering, let us first define the general task. We consider a taxonomy a hierarchical cluster model. The typical cluster conditions are assumed to hold: all items within a taxonomy node (cluster) are more similar to each other than to items in another node (cluster). In addition to a similarity measure, *collaborative clustering* uses taxonomies of others in order to enhance a cluster model, either by covering more objects (here: songs) or by introducing new nodes. In other words, the enhancement either introduces more music plays into the taxonomy, or more structure. In both cases, the user benefits from the collections of others. Figure 3 illustrates the effect of collaborative clustering. The dotted node is one which contains more items, the dotted structure refines the top node's structure.

More formally, collaborative clustering is defined as clustering with given hierarchical cluster models (taxonomies) as background knowledge.

Definition 4. *Given a set of taxonomies* $CM = \{C_0, C_1, ..., C_k\}$, *where* $X = \bigcup_{i=0,k} ext(C_i)$, *and a set of items* $X_q \subseteq X$, COLLABORATIVE CLUSTERING *delivers a small number of taxonomies* $\{C_{result_1}, ...C_{result_n}\}$ *such that the clustering conditions hold and there is some* $C_i \in CM$ *and a* C_{result_j} *with*

- $ext(C_i) \subset ext(C_{result_j})$ *or*
- $ext(C_i) = ext(C_{result_j})$ *but the number of nodes in* C_{result_j} *is larger than the number of nodes in* C_i.

In contrast, hierarchical clustering returns some C_{result} on the basis of X_q alone, and collaborative filtering returns one cluster node c of maximal similarity for X_q together with its extension $X_q \subseteq ext(c)$. We have placed collaborative clustering into our distributed scenario (see section 1.1). This means, that one user with the current taxonomy C_i is seeking for enhancements with respect to a set of songs X_q and makes use of $\{C_0, C_1, ..., C_{i-1}, C_{i+1}, ..., C_k\}$. At the side of the peers, the process of finding taxonomies which best fit the query has to be implemented (Section 3.1). At the side of the user invoking the query, the received taxonomies have to be assessed and the best one incorporated into the user's own (Section 3.2).

3.1 Finding taxonomies with best fit

The fit of a taxonomy C with respect to a set of items X_q can be measured by precision and recall, known from Information Retrieval. There, two sets of items, namely the retrieval result and the intended result, are compared. In analogy, *precision* here means the fraction of items in the tree C that are contained in the set of query items X_q as well:

$$precision(ext(C), X_q) = \frac{|ext(C) \cap X_q|}{|ext(C)|}.$$

Recall here means the fraction of items in X_q contained in the tree C:

$$recall(ext(C), X_q) = \frac{|ext(C) \cap X_q|}{|X_q|}.$$

The straight-forward definition of the fit is the F-measure between precision and recall.

However, using the possibly very large extension of C makes the comparison slow. Hence, we select a set of items Z_C for representing C. $Z_C = ext(C)$

would be, again, the representation as used in the basic agglomerative clustering methods. It is inefficient and sensitive to outliers. $|Z_C| = 1$ represents every (sub-)tree by exactly one point, not necessarily a member of $ext(C)$. This representation of clusters is chosen, e.g., for k-means clustering. It is very efficient, but decreases the result's accuracy. We follow the idea of using a set of "well scattered points" to represent a cluster: $1 < |Z_C| < |ext(C)|$ represents a tree by a set of points as used in the clustering method CURE [12]. It is efficient and accurate. The selection of the points is easy in our case, because the items in the extension of C are already clustered (otherwise this taxonomy would not qualify for collaborative clustering). We choose the scattered points by sampling items from each node of the taxonomy locally.

The straight-forward definition has an additional disadvantage: the strict intersection in precision and recall constrains the fit too much. Hence, we replace the intersection of Z_C and X_q by a standard similarity measure. We define the generalized precision *prec* as a function that compares every item in Z_C with all items in X_q and selects $x_i \in X_q$ with the highest similarity:

$$prec(Z_C, X_q) = \frac{1}{|Z_C|} \sum_{z \in Z_C} max\{sim(x, z) | x \in X_q\}.$$

For generalized recall, we compare every item in X_q with all items in Z_C, again choosing the maximal similarity:

$$rec(Z_C, X_q) = \frac{1}{|X_q|} \sum_{x \in X_q} max\{sim(x, z) | z \in Z_C\}.$$

Please note that using a similarity function which maps identical items to 1 and 0 otherwise leads to the usual definition of precision and recall. The fit between a taxonomy and a set of items now becomes:

$$fit(C, X_q) = \frac{(\beta^2 + 1) rec(Z_C, X_q) prec(Z_C, X_q)}{\beta^2 rec(Z_C, X_q) + prec(Z_C, X_q)}. \quad (1)$$

So far, we have ignored the hierarchical structure of C. We can make use of the structure to find the best fit in a more efficient bottom-up manner. The fitness values of all C_j which are direct successors of C_i are combined with the fitness of the top node of C_i to become the fitness of C_i w.r.t. X_q. Remember, that $ext(C_i)$ was defined as the objects *top* at the root node,

plus all the objects covered by its successors. The fitness is weighted by the cardinality of a (sub-)tree's extension.

$$prec(Z_C, X_q) = \frac{|Z_{top}|}{|Z_C|} prec(Z_{top}, X_q) + \sum_{C_j \in dirSuccessors(C)} \frac{|Z_{C_j}|}{|Z_C|} prec(Z_{C_j}, X_q).$$

Note, that the bottom-up procedure allows for pruning. We normalize the precision of $Successors(C_i)$ by the cardinality of the overall taxonomy's extension, $|ext(C)|$, and the not yet considered items above. This gives us an estimation of how far we have got. If this estimate is below our current best precision value, we stop the bottom-up procedure.

A user at site v starts a query by broadcasting it to the neighborhood. Each site in this neighborhood performs the matching procedure locally and sends the result back to querying site. The answers form the response set.

The time complexity for answering a query is constituted by the amount of time used for finding the best matches and the message delay. Using the extensional representation, time is linear in the number of query items and in the number of taxonomy nodes at a site v. Representing each concept by a fixed number of points in feature space, finding the best matches is linear in the number of query items, taxonomy nodes, and features F used for similarity calculation.

3.2 Sequential covering

The neighbor users have sent taxonomies which best fit the query X_q. Regarding Figure 1, we now want to produce the result from the response. The response set can be ordered with respect to the fitness (equation 3.1). The user selects the best one. However, even for the top ranked taxonomy we cannot expect a perfect recall. Hence, several items in X_q remain uncovered. How to incorporate these into the taxonomy?

A simple and fast method is a hierarchical version of the nearest neighbor algorithm. However, there are two problems with this approach. First, it requires well-suited features. We set aside this problem until Section 4. Assuming that we can apply the same set of features to all taxonomies, we are still left with the second disadvantage. Classifying the remaining items of

X_q into the taxonomies of the response set does not change the taxonomies' structure. The user has to select just one and insert it into his own taxonomy.

We can do better by combining taxonomies. There might be some taxonomies fitting X_q equally well but due to high performance on different subsets of X_q. In such a situation, the user would want to have these taxonomies combined. Moreover, a certain subset of items could possibly be not well covered by any taxonomy in the response. In this case, the user might want to broadcast this certain subset and receive k taxonomies for it. The results of the two queries will then be combined. In both cases, the combination becomes a new taxonomy which the user incorporates into his. Combinations have the long-term advantage that the number of structurally distinct taxonomies in the overall peer to peer network increases. Therefore, we have implemented the sequential covering algorithm which combines taxonomies and covers X_q by the result.

Sequential covering consists of the processes

- $sortIn(X_q, C)$, which puts the items in X_q at those nodes C_j in C where the similarity between item and the current well chosen points Z_{C_j} exceeds a user-given threshold.
- $merge(C, C')$ combines two taxonomies by creating a *merge* node whose successors C and C' become.

If X_q can be completely sorted into a taxonomy C – be it the own, already existing one, or one of the taxonomies received from the peers – there is no need for further processing. The user's taxonomy is enriched by an enhanced extension at the nodes, or it is enhanced by adding the received taxonomy as a successor of the node whose extension had been X_q (see Figure 3).

If X_q can only partially be covered by any of the received taxonomies, it is split into $X_{covered}$ and $X_{uncovered}$. For $X_{covered}$ there exists already a taxonomy, say C. $X_{uncovered}$ is broadcasted to the peers, new taxonomies C'_i are received and tentatively merged with C. The best combination according to the fit (equation 3.1) is added as a successor of the node, which had X_q. Figure 4 illustrates this. The user has received two taxonomies, C_1 and C_2, from his peers. X_q is partially covered by C_1 and the user incorporates C_1 into his own taxonomy, classifying the covered items in X_q into C_1 nodes by $sortIn$. The uncovered items form a new query to the peers. The user receives C'_1 and C'_2, selects the latter and merges it with C_1, so that the combined taxonomies replace the node with X_q as extension.

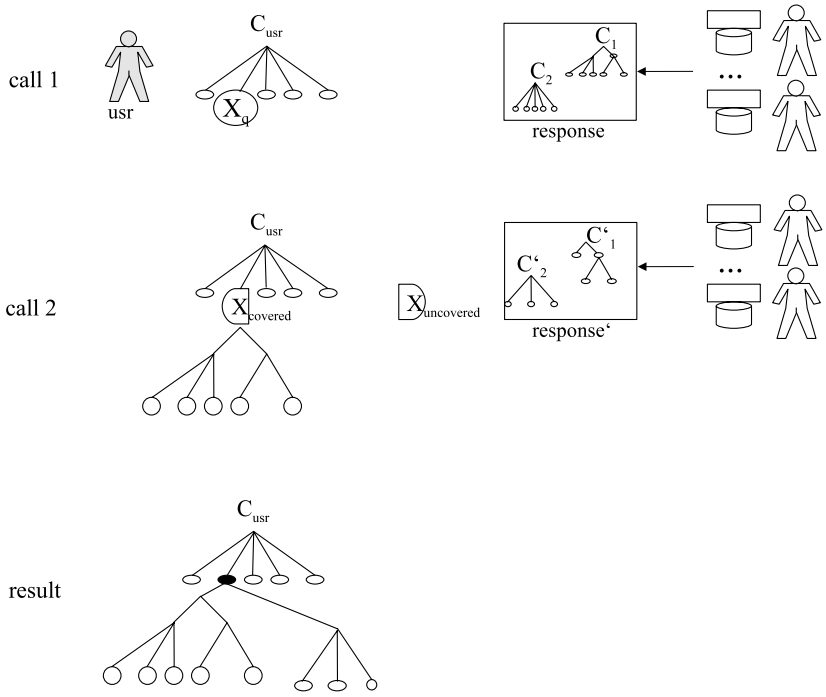

Figure 4: The neighbor users have sent taxonomies which best fit the query X_q. Sequential Covering sorts items of X_q into C_1 according to a similarity threshold. The remaining items form a second query set $X_{uncovered}$ leading to a new response set. The user selects another taxonomy C_2' which is added together with C_1 under a new merge node (black).

The user's taxonomy now can become different from all already existing taxonomies, thus enhancing the pool of taxonomies in the peer to peer network.

3.3 Experiments

The evaluation of collaborative clustering is performed on a real world benchmark dataset gathered in a student project on distributed audio classification based on peer-to-peer networks (Nemoz). The data set contains 39 taxonomies and overall 1886 songs [15][2]. All experiments described in this

[2]Available at http://www-ai.cs.uni-dortmund.de/audio.html.

paper were performed with the machine learning environment YALE [11][3].

Evaluation is performed by leave one taxonomy out and then use it for evaluation. The extension is the same for the left-out and the learned taxonomy. Hence, we evaluate structural similarity. We construct a matrix for each taxonomy, where the tree distance between all items is entered. The difference between the tree distance in the left-out and the one in the learned taxonomy for each pair of items is summed up for all item pairs. The sum of the differences divided by the number of items gives us the evaluation criterion *absolute distance*. A second criterion is the *correlation* between the tree distance in the left-out with the one in the learned taxonomy. Finally, for each cluster in the left-out taxonomy we search for the best corresponding cluster in the learned taxonomy. The average performance over all user-given clusters is then used as evaluation measure (*FScore*).

One major problem of using FScore on hierarchical cluster models is, that it is not normalized with respect to the number of created clusters. Finer grained structures therefore always lead to equal or better performance than their coarse grained variants. This however does often not reflect the similarity to the user-given cluster model.

We compare our approach with single-link agglomerative clustering using cosine measure, top down divisive clustering based on recursively applying kernel k-means [8] (TD), and with random clustering.

Collaborative clustering was applied using cosine similarity as inner similarity measure. The parameters for all algorithms were chosen globally optimal. TD and random clustering were started five times with different random initializations. We use a set of 20 features which where shown to work well in a wide range of applications [27] as underlying audio features. For single-link agglomerative clustering and top down divisive clustering based on kernel k-means, we use an additional feature set (audio/co). This set contains the audio features plus a set of 'collaborative features', which are constructed by adding a feature for each taxonomy node in the training set. A feature has a value of one for an item, if the corresponding item is covered by this node, zero otherwise. In this way we incorporate co-occurrence information, which is a straightforward possibility to make use of the existing cluster models [37, 39].

Table 1 shows the results. As can be seen, the collaborative clustering approach performs best. Note however, that absolute distance does not lead

[3]Available at http://yale.cs.uni-dortmund.de.

Method	Correlation	Absolute distance	FScore
Collaborative Clustering	0.44	0.68	0.63
TD audio	0.19	2.2	0.51
TD audio/co	0.23	2.5	0.55
single-link audio	0.11	9.7	0.52
single-link audio/co	0.17	9.9	0.60
random	0.09	1.8	0.5

Table 1: The results for different evaluation measures.

Representation	Correlation	Absolute distance	FScore
all points	0.44	0.68	0.63
$\|Z_{zop}\| = 10$	0.44	0.68	0.63
$\|Z_{top}\| = 5$	0.41	0.69	0.63
$\|Z_{top}\| = 3$	0.40	0.69	0.62
centroid	0.19	1.1	0.42

Table 2: The influence of concept representation and the number $|Z_{top}|$ of representative points.

to results that are representative for agglomerative clustering, because the corresponding taxonomies are usually quite deep, while the user constructed taxonomies were rather shallow. Applying user defined taxonomies as the basis for a collaborative similarity measure (audio/co) does indeed improve the correlation for both, TD and single-link clustering. However, absolute distance is not improved. We found the resulting taxonomies were relatively unbalanced. This increases absolute distance while correlation is barely affected.

A second experiment inspects the influence of the representation on the accuracy. The results of collaborative clustering with different numbers of instances at a node are shown in Table 2. Representing clusters (taxonomy nodes) by all points performs best. Using a single centroid for representing a subtree leads to inferior results, as we already expected. Well scattered points performed well. We obtain good results even for a very small number of representative items at each node of the cluster model.

We also evaluated how the number of taxonomies in the response set

Alternatives	Correlation	Absolute distance	FScore
5	0.44	0.68	0.63
3	0.38	0.73	0.60
1	0.34	0.85	0.56

Table 3: The influence of response set cardinality.

influences the quality of the result. The result should be clearly inferior with a decreasing number of alternatives. Table 3 shows the result. On one hand, we observe that even with just one model returned, collaborative clustering still outperforms the other methods with respect to tree distance. We can conclude that our approach can be applied even in domains where only one taxonomy can be received. On the other hand, the results are, indeed, getting worse with less alternatives. Providing alternative solutions seems to be essential for improving the quality of results at least in heterogeneous settings as the one discussed here. This approves our general hypothesis.

4 Feature transfer for learning from audio data

The distributed collaborative clustering approach introduced in the last section can be used for the retrieval of taxonomies. The user can select the most suitable taxonomy C for a query set X_q. The task now is to classify the items of the query set according to the classification labels induced be the successors of C. In many cases the classifier for this taxonomy can simply make use of the extracted features F delivered with the taxonomy structure (see section 2). This feature set was created on purpose to support the classification task associated with the taxonomy at hand. However, in some cases these feature sets cannot be used for classification:

- Users might create own taxonomies and sort their music into these taxonomies themselves. However, new songs should be automatically sorted and hence a tailored feature set is required.
- During sequential covering (section 3.2) new nodes were created to merge different taxonomies. Initially, no optimal feature set exists for these nodes.

It was already stated in the introduction that no generic feature set exists for all audio classification tasks which are posed by the different taxonomies. Learning the feature extraction for a particular classification task delivers good results [24, 44]. However, training the feature extraction is time-consuming and demands a sufficient set of examples. We again try to tackle our problem by exploiting the work of other users. Some audio data have already been used for learning. We want to transfer the feature extraction to similar learning tasks. We have developed a new similarity measure of learning tasks from audio data which is based on already learned feature weights. The idea is to transfer extracted features between similar taxonomies. This transfer should speed up the search for the best extraction method in such cases in which a successful feature set has already been generated by another user.

In general, many inductive learning problems cannot be solved accurately by using the original feature space. This is due to the fact that standard learning algorithms cannot represent complex relationships as induced, e.g., by trigonometric functions. For example, if only base features F_1 and F_2 are given but the target function depends highly on $F_c = \sin(F_1 \cdot F_2)$, the construction of the feature F_c would ease learning – or is even necessary for any reasonable prediction at all [4, 7, 20].

In this work, we propose a case-based approach to feature extraction. Taxonomies are stored together with a corresponding set of extracted features. These features are delivered together with the taxonomies. The essential part of our method is a new representation model for taxonomy classification tasks and a corresponding distance measure. This allows the transfer in the cases described above where no extension based similarity can be calculated.

4.1 Basic definitions for feature transfers

As stated in section 2 the taxonomy is enriched by the features and their weights, becoming the tuple: $C = \langle ID, top, F, w, Successors(C) \rangle$.

Before we state the conditions which must be met by any method which compares the classification tasks induced by a taxonomy using feature weights only, we first introduce some basic definitions. Let F_i be the feature set for taxonomy C_i. The taxonomy C_i also induces the classification labels, i.e. the target variable. The items of F_i are called *features* F_{ik}. The objective of every taxonomy classification task is to describe the probability $Pr(C_i|F_i)$.

We assume that each set of features F_i is partitioned into a set of *base features* F_B which are common to all taxonomies C_i and a set of *extracted features* $F_i \setminus F_B$.

We now introduce a simple model of feature relevance and interaction. The feature F_{ik} is assumed to be irrelevant for a taxonomy C_i if it does not improve the classification accuracy:

Definition 5. *A feature F_{ik} is called* IRRELEVANT *for a taxonomy C_i iff F_{ik} is not correlated to the target variable of C_i, i. e. if $Pr(C_i|F_{ik}) = Pr(C_i)$.*

The set of all irrelevant features for a learning task C_i is denoted by IF_i.

Two features F_{ik} and F_{il} are alternative for a taxonomy C_i, denoted by $F_{ik} \sim F_{il}$ if they can be replaced by each other without affecting the classification accuracy. For linear learning schemes this leads to the linear correlation of two features:

Definition 6. *Two features F_{ik} and F_{il} are called* ALTERNATIVE *for a taxonomy C_i (written as $F_{ik} \sim F_{il}$) iff $F_{il} = a + b \cdot F_{ik}$ with $b > 0$.*

This is a very limited definition of alternative features. However, we will show that most weighting algorithms are already ruled out by conditions based on this simple definition.

4.2 Efficient comparison of classification tasks

The objective of our work is to speed up feature extraction and improve prediction accuracy by exploiting the extracted features of other users. To this end, a taxonomy C_i is completely represented by a feature weight vector w_i. The vector w_i is calculated from the base features F_B only. The ideal features, extracted especially for this learning task, usually are not present in the base features. Our concern here is to compare or index extracted features by their approximation through weights of the basic features. The idea is that a given learning scheme approximates similar extracted features by a set of base features using similar weights. Hence, the weight vector indicates the similarity of the extracted features. This motivation is derived from the field of feature construction from non-series data but our experiments show that it holds for audio data, too.

Our approach works as follows: for a taxonomy task C_i we first calculate the relevance of all base features F_B. We then use a distance function

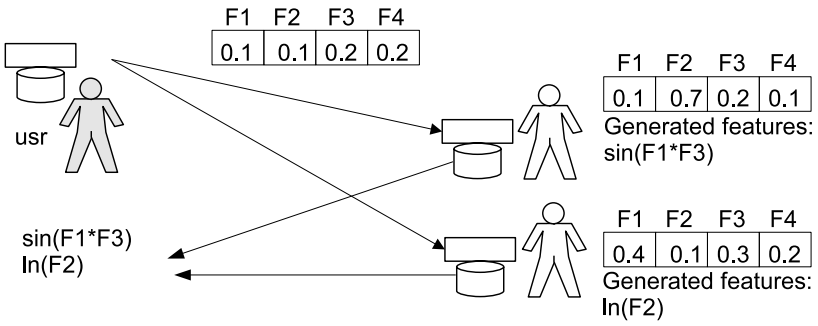

Figure 5: Overview of the distributed feature extraction process. The user *usr* sends a query consisting of feature weights only and retrieves a small set of descriptions how to extract additional features.

$d(C_i, C_j)$ to find the k most similar taxonomies. Please note that this distance measure does not use any extensional description of the taxonomies and works in all cases, even if no items are the same for two taxonomies. Finally, we create a set of extracted features as union of the extracted features associated with the retrieved taxonomies. See Figure 5 for an overview.

While feature weighting and feature extraction are well studied tasks, the core of our algorithm is the calculation of d using only the relevance values of the base features F_B. In a first step, we define a set of conditions which must be met by feature weighting schemes. In a second step, a set of conditions for learning a task distance is defined which makes use of the weighting conditions.

Weighting Conditions. *Let w be a* WEIGHTING FUNCTION $w : F_B \to \mathbb{R}$. *Then the following must hold:*

(W1) $w(F_{ik}) = 0$ *if* $F_{ik} \in F_B$ *is irrelevant*

(W2) $F_i \subseteq F_B$ *is a set of alternative features. Then*

$$\forall S \subset F_i, S \neq \emptyset : \sum_{F_{ik} \in S} w(F_{ik}) = \sum_{F_{ik} \in F_i} w(F_{ik}) = \hat{w}$$

(W3) $w(F_{ik}) = w(F_{il})$ *if* $F_{ik} \sim F_{il}$

(W4) Let AF be a set of features where

$$\forall F_{ik} \in AF : (F_{ik} \in IF_i \vee \exists F_{il} \in F_B : F_{ik} \sim F_{il}) .$$

Then

$$\forall F_{il} \in F_B : \nexists F_{ik} \in AF : F_{il} \sim F_{ik} \wedge w'(F_{il}) = w(F_{il})$$

where w' is a weighting function for $F'_B = F_B \cup AF$.

These conditions state that irrelevant features have weight 0 and that the sum of weights of alternative features must be constant independently of the actual number of alternative features used. Together with the last conditions this guarantees that a set of alternative features is not more important than a single feature of this set. Obviously, this is a desired property of a weighting function for the comparison of classification tasks. In the following we assume that for a modified space of base features F'_B the function w' denotes the weighting function for F'_B according to the definition in (W4).

Additionally, we can define a set of conditions which must be met by distance measures for classification tasks which are based on feature weights only:

Distance Conditions. *A* DISTANCE MEASURE *d for classification tasks is a mapping $d : T \times T \to \mathbb{R}^+$ which should fulfill at least the following conditions:*

(D1) $d(t_1, t_2) = 0 \Leftrightarrow t_1 = t_2$

(D2) $d(t_1, t_2) = d(t_2, t_1)$

(D3) $d(t_1, t_3) \leq d(t_1, t_2) + d(t_2, t_3)$

(D4) $d(t_1, t_2) = d(t'_1, t'_2)$ *if* $F'_B = F_B \cup IF$ *and* $IF \subseteq IF_1 \cap IF_2$

(D5) $d(t_1, t_2) = d(t'_1, t'_2)$ *if*
$F'_B = F_B \cup AF$ *and* $\forall F_k \in AF : \exists F_l \in F_B : F_k \sim F_l$

(D1)–(D3) represent the conditions for a metric. These conditions are required for efficient case retrieval and indexing, using e. g. M-Trees [5]. (D4) states that irrelevant features should not have an influence on the distance. Finally, (D5) states that adding alternative features should not have an influence on distance.

4.3 Applicability of weighting schemes and distance measures for classification task comparisons

In this section we will state that many feature weighting approaches do not fulfill the conditions (W1)–(W4). Furthermore, one of the most popular distance measures, the Euclidian distance, cannot be used as a classification task distance measure introduced above. Proofs for all theorems in this section can be found in [25].

Lemma 1. *No feature selection method does fulfill the conditions (W1)–(W4).*

Lemma 2. *Any feature weighting method for which $w(F_{ik})$ is calculated independently of $F_B \setminus F_{ik}$ does not fulfill the conditions (W1)–(W4).*

Lemma 2 essentially covers all feature weighting methods that treat features independently such as information gain [31] or Relief [18]. The next theorem states that the Euclidian distance cannot be used as a distance measure based on feature weights.

Theorem 3. *Euclidean distance does not fulfill the conditions (D1)–(D5).*

The next theorem, Theorem 4, states that the feature weights delivered by a linear Support Vector Machine (SVM) obey the proposed weighting conditions. Support Vector Machines are based on the work of Vapnik in statistical learning theory [41]. They aim to minimize the regularized risk $R_{reg}[f]$ of a learned function f which is the weighted sum of the empirical risk $R_{emp}[f]$ and a complexity term $||w||^2$:

$$R_{reg}[f] = R_{emp}[f] + \lambda||w||^2.$$

The result is a linear decision function $y = \text{sgn}\left(\langle w, x\rangle + b\right)$ with a minimal length of w. The vector w is the normal vector of an optimal hyperplane with a maximal margin to both classes. One of the strengths of SVMs is the use of kernel functions to extend the feature space and allow linear decision boundaries after efficient nonlinear transformations of the input [35]. Since our goal is the extraction of features during preprocessing we can just use the most simple kernel function which is the dot product. In this case the components of the vector w can be interpreted as weights for the features.

Theorem 4. *The feature weight calculation of SVMs with linear kernel function meets the conditions (W1)–(W4).*

In order to calculate the distance of classification tasks based only on a set of base feature weights we still need a distance measure that meets the conditions (D1)–(D5).

Theorem 5. *Manhattan distance does fulfill the conditions (D1)–(D5).*

We conclude that SVM feature weights in combination with Manhattan distance fulfill the necessary constraints for a classification task distance measure based on feature weights.

4.4 Experiments

In this section we evaluate the ability of our approach to speed up the extraction of essential features, while the accuracy is preserved. As mentioned before, finding the optimal feature set is a very demanding task which must be performed for each unknown learning task anew. For our experiments, we can tackle this problem in a simpler way. Since we know all training taxonomies in advance we can extract a large feature set for several prototypical training taxonomies. We then perform a feature selection to determine the best feature set for each learning task. Using only a small subset of base features F_B we regard the additionally selected features as the set of specially extracted features $F \setminus F_B$ for each learning task. Additional experiments for synthetic data can be found in [26].

Again, we used the benchmark dataset containing 39 user made taxonomies. The taxonomies were split into a training set of 28 taxonomies and a test set of 11 taxonomies. For both, a set of flat binary classification tasks was generated by splitting the items at each inner node according to subconcepts. As stated before we selected the set of optimal features for each of the learning tasks using a wrapper approach (forward selection) in combination with a nearest neighbor learner evaluated by 10-fold cross validation. We used classification accuracy as the optimization criterion. The underlying feature set contained 20 'allround' features [27] and 123 specialized features created by the approach described in [24]. We selected 10 base features which performed well on a wide range of learning tasks in order to allow for an efficient comparison. We then generated base feature weights for each learning task in the training set and test set using a SVM [32]. The average accuracy was measured for all learning tasks in the test set. We used the base feature set only, the selected set of optimal features for each learning task, and features recommended by case-based feature construction

	Accuracy	Time	Optimization cycles
base features	0.79	-	-
optimal features	0.92	42s	3970
cbfc (k = 1)	0.85	3s	257
cbfc (k = 3)	0.88	5s	389
cbfc (k = 9)	0.89	8s	678

Table 4: Average accuracy and effort for learning using base features, optimized features and feature transfer (cbfc).

with a varying number of predictors. We also measured the effort of finding a good subset of features in terms of run-time and the number of times the inner evaluation operator for feature selection approach was invoked.

The result is shown in table 4. As expected, using the base features only, leads to the lowest accuracy. Choosing an optimal set of features for each task separately leads to the highest accuracy. Using the features of the k most similar taxonomies with respect to the similarity measure described in this section performs in between. However, this feature transfer efficiently achieves an accuracy close to the optimal one. This empirically supports our hypothesis, that our approach can combine the best of both worlds: very fast learning while a high accuracy is preserved. In real-world applications the described approach performs even better. Applying an automatic feature extraction method instead of merely applying a feature selection is very demanding and might take up to several days while the run-time of our approach basically remains the same.

In addition to the above evaluation, we visualized the notion of related tasks. Figure 6 shows the base feature vectors of all learning tasks after a singular value decomposition to transform the data into two dimensions. On the one hand, we can see the heterogeneity of learning tasks. On the other hand, several learning tasks form groups of tasks with similar base weights. This approves our hypothesis: while in general different learning tasks require different features, some of the learning tasks resemble each other at least to some extend.

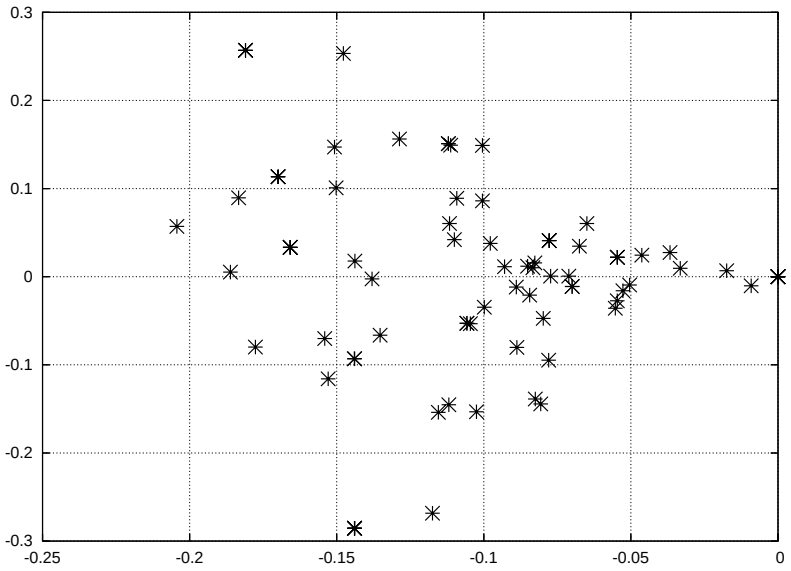

Figure 6: The base feature weights of the audio test cases after a dimensionality reduction on two dimensions.

5 Related Work

There is a clear demand to support users in exploring and managing (private) music collections. Several popular audio management systems offer only limited assistance. Users can usually browse their audio collection using a vendor defined taxonomy only. Manual structuring of audio files is limited to creating playlists. Some tools, as *iTunes* or *amaroK*, offer the creation of intelligent playlists. However, these playlists basically are dynamic database views.

Applying Machine Learning methods to the field of personal music management offers many new opportunities. Typical applications include the classification of music items according to predefined schemes like genres [14, 21], automatic clustering and visualization of music plays [34, 27], recommendations of songs [36], as well as the automatic creation of playlists based on audio similarity and user feedback [29, 23].

A key issue in all these approaches is the representation of the underlying music plays. A large variety of approaches has been proposed for extracting features from audio data [13, 40]. However, it turns out that optimal audio

features strongly depend on the task at hand [30] and the current subset of items [27]. One possibility to cope with this problem is to learn an adapted set of features for each learning task separately [24, 44]. These approaches achieve a high accuracy, but are computationally very demanding and not well suited for real time processing. Furthermore, it is often important to capture cultural aspects not inferable from the audio data. Such aspects have a significant influence on how people structure and perceive their music [1]. Cultural aspects are usually incorporated by co-occurrences of songs in playlists [36] or by textual data [19, 33].

While for supervised learning tasks, at least in principle, optimal features can be evaluated and selected automatically, unsupervised methods are much more challenging. It is well known, that a correct or optimal clustering of items depends strongly on intentions and preferences of the user. This has lead to several approaches that allow for a certain personalization in clustering and visualizing data. A first possibility is to let users directly define, which aspects should be used for structuring [28]. Secondly, users can provide some constraints on the clustering result [6]. These approaches are usually referred to as constrained or semi-supervised clustering. Supervised Clustering is a variant of semi-supervised clustering. Users provide a partial clustering of the items in order to express preferences [10]. Personalization approaches are complementary to our approach since we provide a small set of non-predefined alternatives, by exploiting structures created already by other users.

The latter aspect connects our approach to another area that has recently found increasing attention: clustering with background knowledge. In general, the idea of exploiting (user supplied) background knowledge has shown advantages, e.g., in text clustering [16] or lane finding in GPS data [42]. The methods differ in which kind of knowledge they exploit. In our case, we use a whole set of existing cluster models directly to recommend new clusterings instead of invoking a feature based clustering algorithm. Thus our approach preserves a maximum of structure in the users' taxonomies, which is crucial, as the resulting clusters still contain the original node relation and label information making them intuitively comprehensible and sound.

Distributed collaborative clustering is also related to approaches of multi-view learning and clustering [3, 2]. As in multi-view learning, the results of one learning process (here: a user-made cluster model) is exploited to turn an unsupervised learning task into a supervised one. Connected to this approach is the combination of several weak clusterings to obtain an optimal

one [39, 37]. However, these approaches assume that all cluster models cover the whole set of items. Furthermore, they are not directly applicable to hierarchical structures.

Finally, our approach is loosely related to work in the area of meta learning [38]. Meta learning focuses on the question how interactions between similar learning tasks can be exploited to improve the performance of the individual learners. In this sense our approach can be seen as meta learning approach to clustering and feature construction for audio data.

6 Conclusion

The use of machine learning for structuring music collections is challenging, because two conflicting requirements have to be fulfilled at the same time, namely, real-time processing and local feature extraction. We tackle the problem by exploiting what peers in a network have already prepared manually. We have presented a distributed meta learning approach to feature extraction that compares classification tasks using relevance weights on a common set of base features only. After stating some very basic conditions for such a distance measure, we have shown that a SVM as base feature weighting algorithm and the Manhattan distance fulfill these conditions, while several other popular feature weighting methods and distance measures do not. We have presented experimental results indicating that our method can speed up feature extraction considerably. Hence, our approach is well suited for fast feature extraction from audio data. Features that have been successful in classifying a set of music plays into a certain taxonomy, are applied to the classification into a similar taxonomy.

The local feature extraction is applied in similarity assessments for structuring collections, again, based on peer's taxonomies. Our collaborative clustering can be seen as constrained clustering or clustering using background knowledge. We have proposed a method for distributed collaborative clustering. Regarding the user-made cluster models as the target taxonomies, we could evaluate our results without requiring user studies which are hard to get. Experiments showed that our method clearly outperforms standard clustering. This means that if hand-tailored taxonomies are available, clustering becomes more precise and delivers a higher recall. As the experiments indicate, clustering music data is a particularly hard task. This is due to the large number of features, where only subsets of them are adequate for local

parts of an overall cluster model. Our approach determines the appropriate part for a particular query set of items by finding the (sub-)tree with the highest similarity. Hence, it focuses on more local structures from the very beginning.

All of the clustering approaches mentioned in the last section, are not local, thus they use the same set of features on all hierarchy levels and over the whole collection of data. In contrast, our approach generates a specialized feature set for each inner node making it very flexible. It can handle both different kinds of audio plays and structuring aspects. Due to the alternative solutions users can build their taxonomies from different, arbitrary heterogeneous sources. Thus, we do not rely on a single global similarity measure.

Our approach to collaborative clustering has some connections with methods for ontology matching [9]. The task of ontology matching is to find corresponding concepts in two ontologies. However in our case we only have a query set of items that does not necessarily form a closed concept. Furthermore, our approach has the ability to create new concepts from given ones instead of merely matching them.

The effectiveness of our approach depends on the set of given models. Sequential covering offers a flexible enhancement of the set of conceptual models. This justifies our hope that the system becomes better through its use by several sites.

Acknowledgements

The support of the Deutsche Forschungsgemeinschaft (SFB 475, "Reduction of Complexity for Multivariate Data Structures") is gratefully acknowledged.

References

[1] Stephan Baumann and Oliver Hummel. Using cultural metadata for artist recommendations. In *Proceedings of the International Conference on WEB Delivering of Music*, 2003.

[2] Steffen Bickel and Tobias Scheffer. Multi-view clustering. In *Proceedings of the International Conference on Data Mining*, 2004.

[3] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Annual Conference on Computational Learning Theory*, 1998.

[4] Avrim L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, pages 245–271, 1997.

[5] Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of International Conference on Very Large Data Bases*, pages 426–435. Morgan Kaufmann, 1997.

[6] David Cohn, Rich Caruana, and Andrew McCallum. Semi-supervised clustering with user feedback. Technical Report TR2003-1892, Cornell University, 2000.

[7] M. Dash and H. Liu. Feature selection for classification. *International Journal of Intelligent Data Analysis*, 1(3):131–156, 1997.

[8] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2004.

[9] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. *Handbook on OntologiesOntology matching: A machine learning approach*, pages 385–404. Springer, Berlin, 2004.

[10] T. Finley and T. Joachims. Supervised clustering with support vector machines. In *Proceedings of the International Conference on Machine Learning*, 2005.

[11] Simon Fischer, Ralf Klinkenberg, Ingo Mierswa, and Oliver Ritthoff. Yale: Yet Another Learning Environment – Tutorial. Technical Report CI-136/02, Collaborative Research Center 531, University of Dortmund, Dortmund, Germany, 2002.

[12] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: an efficient clustering algorithm for large databases. In *Proceedings of International Conference on Management of Data*, pages 73–84, 1998.

[13] G. Guo and S. Z. Li. Content-Based Audio Classification and Retrieval by Support Vector Machines. *IEEE Transaction on Neural Networks*, 14(1):209–215, 2003.

[14] F. Guoyon, S. Dixon, E. Pampalk, and G. Widmer. Evaluating rhytmic descriptors for musical genre classification. In *Proceedings of the International AES Conference*, 2004.

[15] Helge Homburg, Ingo Mierswa, Bülent Möller, Katharina Morik, and Michael Wurst. A benchmark dataset for audio classification and clustering. In *Proceedings of the International Conference on Music Information Retrieval*, 2005.

[16] Andreas Hotho, Steffen Staab, and Gerd Stumme. Ontologies improve text document clustering. In *Proceedings of the International Conference on Data Mining*, pages 541–544, 2003.

[17] S. Jones, S. J. Cunningham, and M. Jones. Organizing digital music for use: an examination of personal music collections. In *Proceedings of the International Conference on Music Information Retrieval*, 2004.

[18] K. Kira and I. A. Rendell. The feature selection problem: Traditional methods and a new algoirthm. In *Proceedings of the National Conference on Artificial Intelligence*, pages 129–134. MIT Press, 1992.

[19] P. Knees, E. Pampalk, and G. Widmer. Artist classification with web-based data. In *Proceedings of the International Conference on Music Information Retrieval*, 2004.

[20] D. Koller and M. Sahami. Toward optimal feature selection. In *Proceedings of the International Conference on Machine Learning*, pages 129–134, 1996.

[21] T. Lidy and A. Rauber. Evaluation of feature extractors and psycho-acoustic transformations for music genre classification. In *Proceedings of the International Conference on Music Information Retrieval*, pages 34–41, 2005.

[22] Z. Liu, Y. Wang, and T. Chen. Audio Feature Extraction and Analysis for Scene Segmentation and Classification. *Journal of VLSI Signal Processing System*, 1998.

[23] Beth Logan. Content-based playlist generation: Exploratory experiments. In *Proceedings of the International Symposium on Music Information Retrieval*, 2002.

[24] Ingo Mierswa and Katharina Morik. Automatic feature extraction for classifying audio data. *Machine Learning Journal*, 58:127–149, 2005.

[25] Ingo Mierswa and Michael Wurst. Efficient case based feature construction for heterogeneous learning tasks. In *Proceedings of the European Conference on Machine Learning*, pages 641–648. Springer, 2005.

[26] Ingo Mierswa and Michael Wurst. Efficient feature construction by meta learning – guiding the search in meta hypothesis space. In *Proceedings of the International Conference on Machine Learning Workshop on Meta Learning*, 2005.

[27] F. Mörchen, A. Ultsch, M. Thies, I. Löhken, C. Noecker, M.and Stamm, N. Efthymiou, and M. Kuemmerer. Musicminer: Visualizing perceptual distances of music as topograpical maps. Technical report, Department of Mathematics and Computer Science, University of Marburg, Germany, 2004.

[28] E. Pampalk, S. Dixon, and G. Widmer. Exploring music collections by browsing different views. In *Proceedings of the International Symposium on Music Information Retrieval*, 2003.

[29] E. Pampalk, G. Widmer, and A. Chan. A new approach to hierarchical clustering and structuring of data with self-organizing maps. *Intelligent Data Analysis*, 8(2), 2005.

[30] T. Pohle, E. Pampalk, and G. Widmer. Evaluation of frequently used audio features for classification of music into perceptual categories. In *Proceedings of the International Workshop on Content-Based Multimedia Indexing*, 2005.

[31] R.J. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[32] Stefan Rüping. *mySVM Manual*. University of Dortmund, AI Unit, 2000. http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/.

[33] M. Schedl, P. Knees, and G. Widmer. Discovering and visualizing prototypical artists by web-based co-occurrence analysis. In *Proceedings of the International Conference on Music Information Retrieval*, 2005.

[34] M. Schedl, E. Pampalk, and G. Widmer. Intelligent structuring and exploration of digital music collections. *e & i – Elektrotechnik und Informationstechnik*, 7/8, 2005.

[35] B. Schölkopf and A. J. Smola. *Learning with Kernels – Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[36] Richard Stenzel and Thomas Kamps. Improving content-based similarity measures by training a collaborative model. In *Proceedings of the International Conference on Music Information Retrieval*, 2005.

[37] Alexander Strehl and Joydeep Ghosh. Cluster ensembles – a knowledge reuse framework for combining partitionings. In *Proceedings of the AAAI*, 2002.

[38] S. Thrun and J. O'Sullivan. Discovering structure in multiple learning tasks: The TC algorithm. In *Proceedings of the International Conference on Machine Learning*, 1996.

[39] Alexander P. Topchy, Anil K. Jain, and William F. Punch. Combining multiple weak clusterings. In *Proceedings of the International Conference on Data Mining*, pages 331–338, 2003.

[40] G. Tzanetakis. *Manipulation, Analysis and Retrieval Systems for Audio Signals*. PhD thesis, Computer Science Department, Princeton University, 2002.

[41] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[42] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schroedl. Constrained k-means clustering with background knowledge. In *Proceedings of the International Conference on Machine Learning*, 2001.

[43] T. Zhang and C. Kuo. Content-based Classification and Retrieval of Audio. In *SPIE's 43rd Annual Meeting - Conference on Advanced Signal Processing Algorithms, Architectures, and Implementations VIII*, 1998.

[44] A. Zils and F. Pachet. Automatic extraction of music descriptors from acoustic signals using eds. In *Proceedings of the 116th Convention of the AES*, 2004.