SUPPORTING FOLK-SONG RESEARCH BY AUTOMATIC METRIC LEARNING AND RANKING

Korinna Bade, Andreas Nürnberger, Sebastian Stober
Otto-von-Guericke University Magdeburg
Faculty of Computer Science
{korinna.bade,andreas.nuernberger,
stober}@ovgu.de

Jörg Garbers, Frans Wiering
Utrecht University
Department of Information
and Computing Sciences
{garbers,fransw}@cs.uu.nl

ABSTRACT

In folk song research, appropriate similarity measures can be of great help, e.g. for classification of new tunes. Several measures have been developed so far. However, a particular musicological way of classifying songs is usually not directly reflected by just a single one of these measures. We show how a weighted linear combination of different basic similarity measures can be automatically adapted to a specific retrieval task by learning this metric based on a special type of constraints. Further, we describe how these constraints are derived from information provided by experts. In experiments on a folk song database, we show that the proposed approach outperforms the underlying basic similarity measures and study the effect of different levels of adaptation on the performance of the retrieval system.

1. INTRODUCTION

Folk song researchers detect and document relations between folk songs and their performances. This helps to understand oral transmission. Today, folk song researchers can digitally encode their transcriptions using common music notation editors and use computational methods to detect similarities between songs. However, as there are different ways to detect features in music, there are many different ways to compare songs. Usually, a single computational similarity value will not match directly with the classification criteria that a musicologist applies. Therefore, we choose a weighted linear combination of different basic similarity measures. Depending on the retrieval task at hand, the optimal weighting (with best retrieval performance) of such a complex similarity measure may differ.

In this paper, we describe a metric learning approach that can derive a good weighting in a semi-supervised manner. We apply constraint-based metric learning and formalize the weight adaptation as an optimization problem that is solved by gradient descent. Constraints that guide the adaptation process can be derived from an existing classification of tunes from a collection. We compare different ways of employing the derived similarities to support different browsing and classification tasks in a system that accepts both previous classified and unclassified queries.

The main **contribution** of this paper lies in describing and evaluating a general methodology that allows folk-song researches to automatically generate complex task-specific similarity metrics from basic similarity measures.

2. RELATED WORK

Metric learning has been a topic of interest in general information retrieval for some time, as using a suitable similarity measure is crucial for the performance of many commonly used approaches for clustering, classification or ranking. The general objective is either to get a query closer to the relevant objects (in a classic retrieval scenario) or to refine the decision boundary between relevant and irrelevant objects (in a classification scenario which does not necessarily require a query). The highly subjective nature of perception and the large variety of ways to represent and compare music in many "plausible" ways make it hard to manually define and tweak a metric according to the characteristics of the input data and the specific retrieval task. Consequently, there exist only few approaches for direct manipulation of the metric as described e.g. in [1] and [2]. In contrast to this, our approach allows a semi-supervised metric adaptation. This requires some labeled objects as training data. For the experiments discussed in this paper, such data was already provided. However, if such information is not available a priori, a relevance feedback approach is usually taken where a user is asked to judge on the relevance of some objects.

The idea of incorporating relevance feedback to improve the performance of an information retrieval system goes back to the 1970s [3]. Since, it has been widely applied and further elaborated – primarily in text but also in image retrieval. Recently, in the field of music information retrieval, several approaches using explicit feedback have been presented that adaptively combine the results of different music representation schemes [4], associate different music similarity perception models with users [5], learn to discriminate between similar and dissimilar pieces [6], adapt to the way of querying by taking into account user-specific humming errors [7], or generate user-adaptive play lists [8–11]. Alternatively, the required information may be collected through the analysis of user actions such as the skipping behavior [12] or manual rearrangement of objects on a map through drag & drop [13].

All these approaches are related to the one presented in this paper in that they rely on some form of metric adaptation. Our approach differs from these in two ways. First, it targets a significantly different application scenario. Second, none of the above approaches is based on constrained metric learning, which is applied here, except for the approach presented in [13] that uses similar constraints to guide the clustering of a self-organizing map.

3. OUR APPROACH

The goal of this work is to assist a folk song researcher in classifying new tunes (Section 3.1). Fundamental to this task is the computation of similarities between tunes. Different measures were developed in the past (Section 3.2). However, a particular musicological way of classifying songs is usually not directly reflected by just one of these measures. We show here how a weighted measure derived from a certain classification scheme can be automatically learned (Section 3.3). Based on this measure, we can support the classification of new items by presenting a ranked list (ordered by similarity) of already classified tunes (Section 3.4).

3.1 Expert classification support

Folk song researchers at the Meertens Institute study and classify folk song variants. Besides other means, songs are traditionally classified by assigning them a so called *melody norm*. This classification captures aspects of musical similarity and historical relationships. One tune cannot be part of more than one class.

The WITCHCRAFT project supports researchers by providing a system that enables browsing by musical content. The system's similarity measures operate on symbolic representations of tunes (Humdrum $**kern$ and MIDI format). A query melody is usually specified by clicking on a *search link* besides a database item. The system then ranks database tunes according to a chosen similarity measure.

Two types of ranking lists are supported by switching on/off a filter that is based on tune classification. In unfiltered mode, the *tune-ranking-list* presents all tunes ordered by similarity. This is handy when looking for all variants of a given song. In filtered mode, the *class-ranking-list* presents only the best ranked melody from each class. [1] Therefore, much fewer items are shown. This is handy when classifying a previously unclassified song or when questioning an existing classification.

3.2 Basic tune similarity measures

In this paper, we distinguish between *basic* similarity measures $sim_j(t_1, t_2)$ as introduced in the following and *linear combinations* thereof (Eq. (1)). Note that the *basic* similarity measures in this paper are themselves complex constructions of often more basic musical and mathematical transformations [14, 15]. However, in future work, we plan to also use more basic building blocks. In our experiments, we consider 14 similarity measures. However, the methods proposed in the following work with any set of measures. 11 of the 14 measures are taken from the *Simile* package. [2] These are *rawEd*, *diffEd*, *nGrSumCo*, *nGrUkkon*, *harmCorE*, *rhytFuzz*, *rhytGaus*, *opti1*, *opti3*, *accents_opti1* and *accents_opti2*. Two distance measures are based on the spectra of *Laplacean* and *Adjacency* graphs [16] and one is an unpublished pitch sequence edit distance, implemented by us. All distance measures were transformed to a similarity through $\mathrm{sim} = (1 + \mathrm{dist})^{-1}$.

3.3 Estimating a weighted similarity

Having given a certain number of expert classifications, the question remains whether we can find an optimal weighting of different tune similarity measures to reflect the similarity underlying these expert classifications. In particular, we are interested in the following weighted sum of n similarity measures:

$$\mathrm{sim}_{\boldsymbol{w}}(t_1, t_2) = \sum_{j=1}^{n} w_j \mathrm{sim}_j(t_1, t_2), \qquad (1)$$

with $w_j \geq 0$, and $\sum_{j=1}^{n} w_j = 1$.

The weight vector $\boldsymbol{w}$ can be learned by methods of constrained clustering, which target on learning a metric [17, 18]. In particular, must-link-before (MLB) constraints [18] can be used. Originally, MLB constraints were proposed for hierarchical clustering to describe the hierarchical relation between three different items. The constraint (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. For our problem at hand, we can use a similarity interpretation instead, i.e., items i_x and i_y should be more similar than items i_x and i_z.

Given a certain query tune q, we know from the expert classification, which other tunes t_r belong to the same (relevant) class and which tunes t_i are irrelevant. As the tunes of the same class should be ranked first and, thus, should be more similar, we can build MLB constraints of the following form: (q, t_r, t_i), which implies that

$$\mathrm{sim}(q, t_r) > \mathrm{sim}(q, t_i). \qquad (2)$$

Hence, the goal is to learn a weight vector $\boldsymbol{w}$, with which the fewest of the MLB constraints known for a certain query are violated. This can be achieved with a gradient descent search similar to the work in [18]. During learning, all constraint triples (q, t_r, t_i) are presented to the algorithm several times until convergence is reached. If a constraint is violated by the current similarity measure, the weighting is updated by trying to maximize

$$obj(q, t_r, t_i) = \mathrm{sim}_{\boldsymbol{w}}(q, t_r) - \mathrm{sim}_{\boldsymbol{w}}(q, t_i), \qquad (3)$$

[1] We found that taking the maximum leads to better results than taking the average of all the similarities.

[2] http://doc.gold.ac.uk/isms/mmm/SIMILE_algo_docs_0.3.pdf

which can be directly derived from (2). This leads to the weight update rule of each individual weight w_j

$$w_j = w_j + \eta \Delta w_j, \quad \text{with} \tag{4}$$

$$\Delta w_j = \frac{\partial obj(q, t_r, t_i)}{\partial w_j} = \mathrm{sim}_j(q, t_r) - \mathrm{sim}_j(q, t_i) \tag{5}$$

where η is the learning rate defining the step width of each adaptation step.

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 1. The complete algorithm is summarized in Figure 1.

```
learnWeights(query tune $q$, tunes $T$, expert classification $C$, $\forall k = 1..n$ : similarity $\mathrm{sim}_k$)
  Determine constraints $MLB$ from $T$ and $C$
  Initialize $\boldsymbol{w}$: $\forall j : w_j := 1/n$
  repeat
    for all $(q, t_r, t_i) \in MLB$ do
      if $\mathrm{sim}_{\boldsymbol{w}}(q, t_r) \leq \mathrm{sim}_{\boldsymbol{w}}(q, t_i)$ then
        $\forall j$ : compute $\Delta w_j$
        $\forall j : w_j := \max(0, w_j + \eta \Delta w_j)$
        $sum_{\boldsymbol{w}} = \sum_j w_j$
        $\forall j : w_j := w_j / sum_{\boldsymbol{w}}$
      end if
    end for
  until convergence
  return $\boldsymbol{w}$
```

Figure 1. The weight learning algorithm

This algorithm learns an *individual weighting* $\boldsymbol{w}^q$ based on the set of MLB constraints for a single query q. However, a weighting that works well for several queries would be more useful. In specific, it is interesting to learn *class weightings* $\boldsymbol{w}^{cl(t)}$ that hold for all queries of the same class $cl(t)$ of a tune t and an *overall weighting* $\boldsymbol{w}^a$ that holds for all queries. These can be computed by the same algorithm using the combined constraint sets from all considered rankings.

3.4 Querying with unclassified tunes

The approach from the previous section can learn an optimal similarity measure based on an expert classification. If new tunes are added to a collection, no expert classification is available at first and, hence, no weights optimized for this query are available based on which a ranking list could be build. If there is no perfect global measure that can be applied to all queries, a different strategy can be followed for this query to build the ranking.

This is based on the already known good weightings of all database tunes, which were determined by the method described in the previous section. If we assume that similar songs also have a similar optimal weighted similarity, we can estimate a weighting for the new query tune q by picking the weighting scheme of the closest tune t_{best} in the database. This can be seen as a case-based approach [19] where each tune in the database and its associated weighted similarity correspond to a case and these stored cases are used to decide how to handle the new case, i.e., how to weight concerning the query tune.

However, this does not yet fully solve the problem, because a weighting scheme is already required to find the closest tune t_{best} for a query. A straight forward approach is to use an overall weighting $\boldsymbol{w}^a$. Alternatively, the more specific class weighting $\boldsymbol{w}^{cl(t)}$ or the individual tune weighting $\boldsymbol{w}^t$ associated with each database tune t can be used, because we already know that these similarities are well suited for comparing any tune with t.[3] We will select the case with the largest (local) similarity and use its weighting to finally rank all tunes in the database according to the query tune q: $\boldsymbol{w}_{best} = \arg\max_{\boldsymbol{w}^t} \ \mathrm{sim}_{\boldsymbol{w}^t}(q, t)$.

For ranking, we can also use any of the weightings associated with the closest case t_{best}, i.e., $\boldsymbol{w}^{t_{best}}$, $\boldsymbol{w}^{cl(t_{best})}$ and $\boldsymbol{w}^a$, where the latter is obviously the same for any database tune. In Section 4.4, we use the notation $\boldsymbol{w}_2 \circ \boldsymbol{w}_1$ to indicate that the first step (closest case selection) was performed using a similarity based on $\boldsymbol{w}_1$ and the second step (ranking) is based on $\boldsymbol{w}_2$.

4. EXPERIMENTS

We conducted experiments to study how well the different weighting techniques perform for already classified (Sec. 4.2) and unclassified tunes (Sec. 4.4) with respect to the two different ranking lists (Sec. 4.1). Further, we analyzed the stability of the learned weighting schemes (Sec. 4.3).

4.1 Dataset and measure evaluation method

Our evaluation is done on 360 well understood single melodic strophes (one strophe per recording) described in [20]. The tunes are classified into 26 disjunct classes. For each pair of tunes all 14 basic similarity measures are considered. These $14 \cdot 360^2$ similarity values are precalculated and need not be recomputed in the learning algorithm and in the construction of ranking lists.

As in the application system (Sec. 3.1), our algorithms produce for each (combined) similarity measure and query tune a tune-ranking-list of all database tunes and a class-ranking list. Both are ordered by decreasing computed similarity. All tunes with the same class as the query are marked as being a relevant result. This gives the ground truth for evaluating the ranked lists. As measures, we compute the *average precision* and *average recall* per rank on the tune-ranking-lists for the set of evaluated queries. For evaluation of class-ranking lists, we are interested in the position of the correct class in such a list. We present here the number of misclassifications at rank 1, the *average rank* of the correct class and the *average inverse rank* over all considered queries. The latter average is less sensitive for single extremely bad class retrievals.

[3] Please note that in this case different weightings are used to compute the similarity to different database tunes, which leads to local distortions of the similarity space around each case. While such a locally distorted metric is unsuitable for the computation of the entire ranking, it may still be useful to retrieve only t_{best} as shown in the experiments in Sec. 4.4.

4.2 Querying with classified tunes

In this section we study the retrieval performance of learned weights (cf. Sec. 3.3) and, thus, whether an automatically determined combination of different existing similarity measures performs better than the individual ones. We consider three weightings of different specificity as motivated in Section 3.3: query-specific weighting $\boldsymbol{w}^q$, class-specific weighting $\boldsymbol{w}^{cl(q)}$ and overall weighting $\boldsymbol{w}^a$. The precision/recall curves for these cases are shown in Fig. 3 (left). Additionally, the performance plots of the two best basic similarity measures, $rawEd$ and $opti1$ have been included into the figure for comparison. Table 1 (top) shows the corresponding evaluation of the class-ranking-lists, ordered by best performance, which gives the same order for all three measures, i.e., *average rank* of correct class (smaller is better), *average inverse rank* (larger is better), and the number of *wrong* classifications (inspecting the first rank).

Not surprisingly, using $\boldsymbol{w}^q$ for similarity computation results in the best retrieval performance. It marks the upper bound of what can be achieved with the learning algorithm. Further, it can be observed that $\boldsymbol{w}^{cl(q)}$ indeed performs better than $\boldsymbol{w}^a$. However, if weights get more specific, the danger of overfitting exists. We will discuss this problem in Section 4.4. Nevertheless, this evaluation indicates that there might not be a single perfect overall similarity measure that can be used in general. Instead data/problem specific measures might be needed, which are especially interesting if they can be determined automatically as through our presented method.

It is interesting to see, that the overall weight performs worse than the best basic similarity measure (*rawEd*) in most precision/recall regions (although only slightly) but that *rawEd* performs worse than all shown measures for the class-ranking-lists. This is caused by the convergence behavior of the algorithm, which is not guaranteed to find a global optimum but a local one.

4.3 Stability of the weighting scheme

In order to assess the stability of the individual weighting schemes throughout the different classes, we conducted two experiments. In the first experiment, we analyzed the individual weightings obtained for the classified tunes (cf. Section 4.2) with respect to the classes. The following two measures were computed for each class:

The *average pairwise inner-class similarity* is computed as the average of the similarity of the weightings for all pairs of tunes within the specific class:

$$\overline{\mathrm{sim}}_{inner}(C) = \underset{t_1,t_2\in C, t_1\neq t_2}{\mathrm{average}} \left\{\mathrm{sim}_{cos}\left(\boldsymbol{w}^{t_1}, \boldsymbol{w}^{t_2}\right)\right\}. \quad (6)$$

Analogously, the *average pairwise cross-class similarity* is computed as the average of the similarity of the weightings for all pairs of tunes where one tune belongs to the specific class and the other to a different class:

$$\overline{\mathrm{sim}}_{cross}(C) = \underset{t_1\in C, t_2\notin C}{\mathrm{average}} \left\{\mathrm{sim}_{cos}\left(\boldsymbol{w}^{t_1}, \boldsymbol{w}^{t_2}\right)\right\}. \quad (7)$$

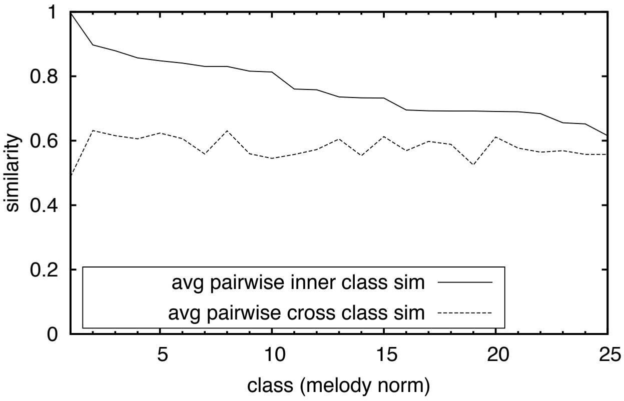

Figure 2. Average pairwise inner- and cross-class similarity of the individual weightings per class (sorted).

For the comparison of two weightings the cosine similarity

$$\mathrm{sim}_{cos}\left(\boldsymbol{w}^{t_1}, \boldsymbol{w}^{t_2}\right) = \frac{\boldsymbol{w}^{t_1}\cdot\boldsymbol{w}^{t_2}}{\|\boldsymbol{w}^{t_1}\|\|\boldsymbol{w}^{t_2}\|} \quad (8)$$

was used. Fig. 2 shows the computed values for all classes (sorted by descending inner-class value for better readability). The inner-class value is always significantly higher than the respective cross-class value. It can be concluded that the individual weightings of tunes belonging to the same class are in general distinct from those belonging to others which explains the usefulness of class weights ($\boldsymbol{w}^{cl(q)}$ in Sec. 4.2). The generally high cross-class values (above 0.5) can be interpreted as an indicator for the existence of a useful overall weighting scheme ($\boldsymbol{w}^a$ in Sec. 4.2).

For the second experiment, we left out one to five relevant tunes selected randomly from a ranking during the learning process for individual weights. The procedure was repeated ten times for each number of excluded tunes. We then compared the $5 \cdot 10$ resulting weighting schemes with the one learned with all available information. To save time, we limited the number of tunes used as queries to two for each of the 26 classes resulting in $2 \cdot 26 \cdot 5 \cdot 10 = 2600$ samples compared.

The number of excluded tunes did not seem to have a large observable effect in our experiment. The different weights learned for the same tune with a differing set of relevant tunes were almost identically with an average similarity of 0.969 ($\sigma = 0.086$). Only a few outliers could be measured, the worst with a minimal similarity of 0.278. From the results we can conclude that the learning algorithm still produces stable results even if almost half of the relevant tunes are removed.

4.4 Querying with unclassifed tunes

In this section we study the retrieval performance for previously unclassified tunes, i.e., for which no previously learned weight $\boldsymbol{w}^q$ or $\boldsymbol{w}^{cl(q)}$ exists. Following the case-based approach described in Sec. 3.4, Fig. 3 (middle and right) shows the respective precision/recall curves. We thereby consider two different real-world situations.

In the first case (Fig. 3; middle) the query tune represents a new tune and is therefore not part of the weight learning. However, the other tunes of the same class are already in the database and therefore used for learning. But of course, the system does not know during ranking which ones these are. In the other case (Fig. 3; right) all tunes from the query tune's class were not used for learning, simulating an entirely new class that shall be added to the database. Thus, no information on this class was available for learning. This is of course an even harder case. For computation of the precision and recall values, all tunes were ranked according to the query tune, including the songs of the unknown class. For both scenarios, we used weights with different specificity in the two steps of the case-based approach (cf. Sec. 3.4).

As expected, the comparison of both diagrams shows that the performance of the learned measures is lower for the harder case of a new class than in the case of a new tune from a known class. The *rawEd* measure can thereby be used as a point of comparison, because it has the same curve in both cases. It showed that the more specific weightings, most notably $\boldsymbol{w}^t \circ \boldsymbol{w}^t$, are much better in the middle graph than in the right one. This is because members of the same tune family can be detected as a case, if they are already in the database. However, specific weightings from other tune families are less fitting. Thus, the approach fails for a new tune family, where no good specific measures are in the database yet. In the middle graph, $\boldsymbol{w}^a$ is best used to establish a case and $\boldsymbol{w}^t$ of that case to finally rank: $\boldsymbol{w}^t \circ \boldsymbol{w}^a$. This approach is also quite good in the right graph, although using only $\boldsymbol{w}^a$ is slightly better. *rawEd* is better at the end of the ranking, while it is worse at the beginning. This is also reflected in the evaluation of class-ranking-lists (Tab. 1). Here, *rawEd* performs worst. With respect to automatic classification, $\boldsymbol{w}^a$ performs best with the fewest errors in the first rank. A comparison of $\boldsymbol{w}^t \circ \boldsymbol{w}^a$ with $\boldsymbol{w}^{cl(t)} \circ \boldsymbol{w}^a$ would be interesting, but the experimental data for $\boldsymbol{w}^{cl(t)} \circ \boldsymbol{w}^a$ is not available, yet.

As a remark it shall be noted that the described methodology of simulating new tunes is very time-consuming because for each considered query the learning has to be redone without the respective information. Therefore, the experiments were done with only 78 query melodies, three melodies from each melody norm. For the development of new similarity measures, the biased evaluation without resampling as used in Section 4.2 can be used to get a rough idea of which measure might be more promising. However, it can never replace a final unbiased evaluation as in this section. Furthermore, the choice of melodies showed a significant impact on the results. Using, e.g., only the reference melodies from [20], the learned measures perform much better in comparison to *rawEd* - also in the most challenging case, while other query tunes are harder to handle. For our evaluations we used the reference melody and two randomly picked other melodies.

5. CONCLUSION

We described an adaptive metric learning approach based on constrained clustering that can be used in folk song research to learn a task-specific similarity measure in form of a weighted linear combination of several basic similarity measures. Individual, class and overall weightings provide different levels for specificity of the adaptation. Experiments on a data set of pre-classified folk songs showed that the combined similarity measures using these weightings can outperform the original basic similarities for ranking and automatic classification.

Table 1. Evaluation of the class-ranking-lists. **Top:** classified tunes (Sec. 4.2). **Middle:** unclassified tunes of a known class (Sec. 4.4). **Bottom:** unclassified tunes of an unknown class (Sec. 4.4).

Measure	Rank	Inverse	1st Wrong
$\boldsymbol{w}^q$	1.042	0.989	6 / 360
$\boldsymbol{w}^{cl(q)}$	1.083	0.985	9 / 360
opti1	1.169	0.975	14 / 360
$\boldsymbol{w}^a$	1.172	0.974	14 / 360
rawEd	1.233	0.967	16 / 360
$\boldsymbol{w}^t \circ \boldsymbol{w}^a$	1.218	0.969	4 / 78
$\boldsymbol{w}^a$	1.231	0.981	2 / 78
$\boldsymbol{w}^t \circ \boldsymbol{w}^t$	1.244	0.957	5 / 78
$\boldsymbol{w}^{cl(t)} \circ \boldsymbol{w}^{cl(t)}$	1.346	0.976	2 / 78
rawEd	1.410	0.946	5 / 78
$\boldsymbol{w}^a$	1.218	0.982	2 / 78
$\boldsymbol{w}^t \circ \boldsymbol{w}^a$	1.244	0.971	3 / 78
$\boldsymbol{w}^t \circ \boldsymbol{w}^t$	1.282	0.942	7 / 78
$\boldsymbol{w}^{cl(t)} \circ \boldsymbol{w}^{cl(t)}$	1.359	0.970	3 / 78

Future experimental work comprises incorporating more basic similarity measures that capture different aspects of the tunes to be classified. Further, the impact of the differing value distributions (within the fixed $[0, 1]$ interval) for the different basic similarities needs to be studied in further experiments as it might cause a bias in the learned weighting schemes.

Future musicological work includes studying clusters of similar weightings. As different weightings represent different metrics, they select different features that separate melody classes. Within a melody norm, several distinct weight clusters suggest the introduction of sub-melody-norms that might be helpful for folk song research. On the other hand, weight clusters shared by different melody norms could be studied to improve the case-based approach. If, e.g., rhythmically ragged melodies generally lead to higher weighted rhythmical similarity measures, then raggedness should be used to select weights instead of rhythmical similarity. For a better support of folk song researchers, the algorithm should be integrated into a graphical user interface. In this context, possible interaction scenarios, e.g., for expert-driven development of new similarity measures, could be examined.

Acknowledgments. This work was supported by the Netherlands Organization for Scientific Research within the WITCHCRAFT project NWO 640-003-501, the German Research Foundation (DFG), and the German National Merit Foundation. Special thanks to contributing developers and data providers.

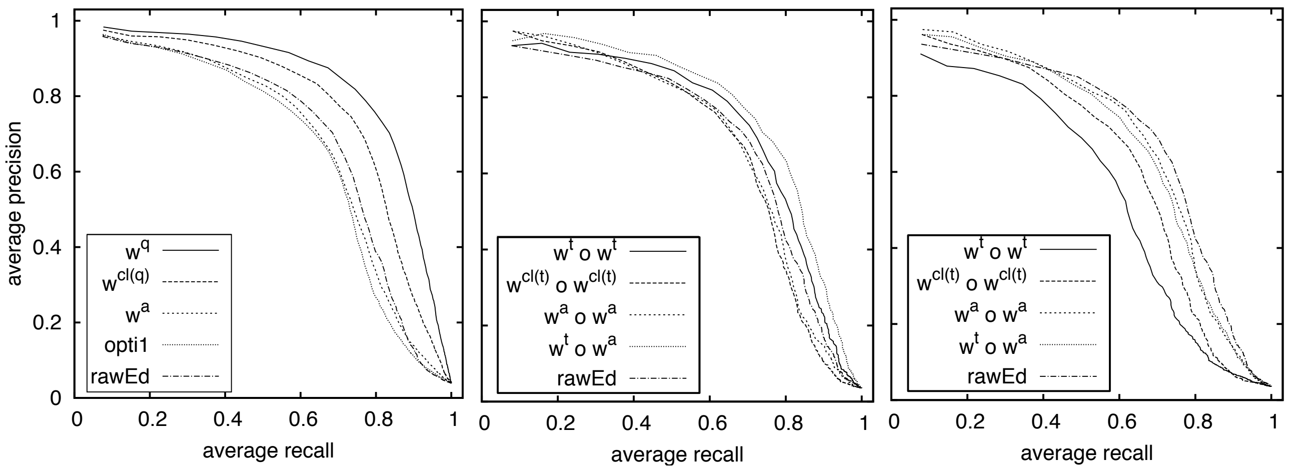

Figure 3. Precison / Recall plots for tune-ranking-lists. **Left:** classified tunes. **Middle:** unclassified tunes of a known class. **Right:** unclassified tunes of an unknown class.

6. REFERENCES

[1] S. Baumann and J. Halloran. An ecological approach to multimodal subjective music similarity perception. In *Proc. of Conf. on Interdisciplinary Musicology*, 2004.

[2] F. Vignoli and S. Pauws. A music retrieval system based on user driven similarity and its evaluation. In *ISMIR*, 2005.

[3] J. J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System - Experiments in Automatic Document Processing*, 1971.

[4] T. Mandl and C. Womser-Hacker. Learning to cope with diversity in music retrieval. In *ISMIR*, 2002.

[5] D. N. Sotiropoulos, A. S. Lampropoulos, and G. A. Tsihrintzis. MUSIPER: a system for modeling music similarity perception based on objective feature subset selection. *User Modeling and User-Adapted Interaction*, 18(4):315–348, 2008.

[6] M. I. Mandel, G. E. Poliner, and D. P. W. Ellis. Support vector machine active learning for music retrieval. *Multimedia Systems*, 12(1):3–13, 2006.

[7] D. Little, D. Raffensperger, and B. Pardo. A query by humming system that learns from experience. In *ISMIR*, 2007.

[8] K. Wolter, C. Bastuck, and D. Gärtner. Adaptive user modeling for content-based music retrieval. In *Proc. of the 6th Int. Workshop on Adaptive Multimedia Retrieval*, 2008.

[9] K. Hoashi, K. Matsumoto, and N. Inoue. Personalization of user profiles for content-based music retrieval based on relevance feedback. In *MULTIMEDIA'03: Proc. of the 11th ACM Int. Conf. on Multimedia*, 2003.

[10] M. Grimaldi and P. Cunningham. Experimenting with music taste prediction by user profiling. In *MIR '04: Proc. of the 6th ACM SIGMM Int. Workshop on Multimedia Information Retrieval*, 2004.

[11] S. Pauws and B. Eggen. PATS: Realization and user evaluation of an automatic playlist generator. In *ISMIR*, 2002.

[12] E. Pampalk, T. Pohle, and G. Widmer. Dynamic playlist generation based on skipping behavior. In *ISMIR*, 2005.

[13] S. Stober and A. Nürnberger. Towards user-adaptive structuring and organization of music collections. In *Proc. of 6th Int. Workshop on Adapt. Multimedia Retr.*, 2008.

[14] D. Müllensiefen and K. Frieler. *The SIMILE algorithms documentation 0.3*, 2006.

[15] D. Müllensiefen and K. Frieler. Cognitive adequacy in the measurement of melodic similarity: Algorithmic vs. human judgments. *Computing in Musicology*, 13, 2004.

[16] A. Pinto, R. H. van Leuken, M. F. Demirci, F. Wiering, and R. C. Veltkamp. Indexing music collections through graph spectra. In *ISMIR*, 2007.

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

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

[19] A. Aamodt and E. Plaza. Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994.

[20] A. Volk, P. Van Kranenburg, J. Garbers, F. Wiering, R. C. Veltkamp, and L. P. Grijp. A manual annotation method for melodic similarity and the study of melody feature sets. In *ISMIR*, 2008.