ANALYZING THE IMPACT OF DATA VECTORIZATION ON DISTANCE RELATIONS

Sebastian Stober & Andreas Nürnberger

Data & Knowledge Engineering Group, Otto-von-Guericke-University Magdeburg, Germany
{sebastian.stober, andreas.nuernberger}@ovgu.de

ABSTRACT

Some popular algorithms used in Music Information Retrieval (MIR) such as Self-Organizing Maps (SOMs) require the objects they process to be represented as vectors, i.e. elements of a vector space. This is a rather severe restriction and if the data does not adhere to it, some means of vectorization is required. As a common practice, the full distance matrix is computed and each row of the matrix interpreted as an artificial feature vector. This paper empirically investigates the impact of this transformation. Further, an alternative approach for vectorization based on Multidimensional Scaling is proposed that is able to better preserve the actual distance relations of the objects which is essential for obtaining a good retrieval performance.

Index Terms— Vectorization, Distance Measures, Multidimensional Scaling, Facets, Aggregation

1. INTRODUCTION

Music information can be described in many different ways: The loudness of a track can be captured in a single (continuous) value. The harmonic key may be a single value, too, but not from a continuous range. Other feature types comprise sets (e.g., instruments, tags), histograms (e.g., chord distribution), intervals (e.g., production period), vectors (e.g., lyrics in the common term frequency vector representation) up to complex descriptions of probability distributions commonly used to describe the timbre in terms of Mel-frequency cepstral coefficients (MFCCs). Moreover, usually more than one feature is used as description.

On the other hand, there are algorithms such as the popular Self-Organizing Maps (SOMs) – or prototype-based clustering approaches in general – that require input data as vectors. Here, the term *vector* is used in its strict mathematical sense, i.e. as an element of a (numerical) *vector space*. For many features, there is no straightforward way of transformation into a flat vector representation without loosing semantics. E.g., writing a covariance matrix as a vector by concatenating the rows completely ignores the feature's semantics when applying vector operations. Sometimes, it may be possible to modify the algorithm such that it can cope with the different data representation as, e.g., described in [1] for Gaussian distributions in SOMs. For the remaining cases, an artificial vectorization step has to be introduced.

Vectorization – in the context of this paper – is a generic pre-processing step that maps each object o of a dataset S onto a vector representation $\vec{x} \in R^n$. This mapping should preserve some characteristic of the data which is particular for the algorithm(s) to be applied afterwards. The focus of this paper lies on the distance relations between the objects which are especially important when using SOMs. Consequently, the approaches covered here require a distance (or dissimilarity) matrix of the dataset to be vectorized as input. They are, however, not confined to any specific feature type and thus generally applicable.

An established vectorization approach that can be considered as common practice, is to simply interpret each row of the distance matrix (containing the pairwise distances of all objects in the dataset) as a feature vector. I.e. the objects are described by the distances amongst each other. This approach has been applied, e.g., for the SOM-based MIR applications Islands of Music [2], nepTune [3] and BeatlesExplorer [4]. The SOMs obtained using this vectorization technique seem satisfactory. However, to the knowledge of the authors, there has not been a formal proof why this vectorization approach works nor an evaluation how well it works so far.

This paper does not aim to give a formal proof either but proposes a methodology for evaluation (Section 2.2). Using a test collection (Section 2.1) similar to the one described for the BeatlesExplorer [4], several experiments motivated by real-world scenarios are conducted (Section 2.3). Furthermore, an alternative vectorization approach based on Multidimensional Scaling (MDS) is described (Section 3.2) and evaluated. MDS has already been applied successfully for vectorization by the SoniXplorer [5] which, however, simply chooses an output dimensionality of $d = 20$ without further analysis of the impact. Such an analysis is provided here with the additional extension to let the MDS automatically choose an appropriate value for the dimensionality parameter depending on a given boundary for the accuracy. Additionally, Section 3.3 describes a vectorization meta-approach suitable for adaptive MIR applications that compute distances as weighted aggregation of several facets such as the BeatlesExplorer [4] and the SoniXplorer [5]. Experimental results are discussed in Section 4. Section 5 concludes this paper.

2. EXPERIMENTAL SETUP

2.1. Test Collection

The collection used for the experiments is similar to the one described in [4] and contains 197 songs from The Beatles. Each song is represented by 20 facets that cover different aspects of music similarity. As defined in [6], a facet refers to a (set of) feature(s) in combination with a respective (facet) distance measure. Distances between two songs are obtained by aggregating the respective facet distances as weighted sum. This way, facet weights are introduced that allow to adapt the importance of each facet according to user preferences. Table 2 contains a list of all facets. The information is extracted (semi-automatically) from Wikipedia[1], LyricWiki[2], Alan W. Pollack's notes on The Beatles[3], manual chord annotations [7] and from the audio recordings using the frameworks CoMIRVA [8] and JAudio [9], A detailed description is given in [4]. In order to avoid an aggregation bias towards facets with large distance values, the distances are normalized such that the mean distance per facet is 1.0.

2.2. Evaluation Measures

The following two straightforward measures are used to assess the quality of a vectorization:

Distance Triples Agreement: This evaluation measure aims to capture how well the distance relations are preserved by the vectorization. To this end, the distance matrix on the original objects $D = (d_{ij})$ is compared with the distance matrix for the vectorized objects $D' = (d'_{ij})$. Both matrices agree with respect to a triple (s, a, b) iff $sign(d_{sa} - d_{sb}) = sign(d'_{sa} - d'_{sb})$, i.e. the vectorized version of the object that was closer to the seed object $o^{(s)}$ is closer to the vector $\vec{x}^{(s)}$ as well which means that the ranking order is maintained by the transformation. The distance triples agreement score of D and D' is the relative number of agreements w.r.t. the total number of possible triples.

Nearest Neighbor Agreement: The retrieval of nearest neighbors plays an important role in many MIR applications such as query-by-example, recommendation and clustering. E.g., for a SOM clustering of vectorized data, it is essential that nearest neighbors are reliably identified because otherwise objects may be assigned to less appropriate locations on the map. Two indicators for the preservation of neighborhoods are the agreement on the closest neighbor and on the ten nearest neighbors (top 10). The closest neighbor agreement is the relative number of objects that have the same nearest neighbor in both, original and vectorized space. Accordingly, the top 10 agreement compares the set of ten nearest neighbors in both spaces (without order taken into account).

[1] http://en.wikipedia.org

[2] http://lyricwiki.org

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

2.3. Test Scenarios

Three test scenarios described in the following sections are considered which are motivated by common applications.

2.3.1. *Vectorizing a Fixed Dataset*

This scenario is by far the most common case in the literature: A dataset is vectorized once and afterwards neither the distances between the objects nor the dataset changes, i.e. no objects are added or removed. Here, the performance of the baseline is most interesting as a poor performance might question the common practice.

For the aggregation of the facet distances, the facets are weighted uniformly. Additionally, analyzing the performance for each single facet gives an impression of how well the vectorization approaches work for different kinds of data.

2.3.2. *Adapting Facet Weights*

What if the facet weights and thus the aggregated distances change after the initial vectorization? Applications like the BeatlesExplorer [4] or the SoniXplorer [5] allow the automatic adaptation of facet weights based on user actions. Changing the facet weights, however, results in modified distances in the original space. One way to propagate this change to the vector space is to compute a new vectorization and all other steps done afterwards such as training a SOM for the vectorized dataset. This can have a severe effect on the visualization even for small weight changes and may confuse or even annoy the user. The SoniXplorer [5] tries to reduce this effect by choosing appropriate initialization values. However, the chain of necessary re-computations is very costly and makes real-time interaction impossible.

An alternative way that avoids the re-computation of the vectorization is proposed in Section 3.3. For its evaluation, a preference-based adaptation approach as motivated in [10] is applied which can be briefly described as follows:

1. Random facet weights are generated that represent the user preferences (which are unknown to the system). Using these weights, the facet distances are aggregated into the user distance matrix $U = (u_{ij})$.

2. Random triples (s, a, b) drawn from U with $u_{sa} < u_{sb}$ serve as training samples for a perceptron learner.

3. The learner uses gradient descent to find a facet weighting that complies with the sampled user preferences.

4. The learned facet weights are applied for aggregation in both, the original and the vector space, and the resulting distance matrices are evaluated against U.

The performance measures are averaged over 100 random user weightings.

2.3.3. Adding New Songs

It might be sufficient for prototypes, demonstrations or retrieval experiments to assume a fixed dataset. However, in the real world this assumption seldom holds. Songs are added and removed from music collections as music taste changes and new interesting music is discovered. In such a case, computing a new vectorization of the changed collection may break existing structures. E.g., a SOM would need to be relearned and the result might look totally different. It would therefore be a welcomed property of a vectorization, if it (to some extend) supported changes of the dataset without severe degradation of the performance. The performance for added objects is of particular interest here as the respective distance information has not been available during the computation of the initial vectorization.

For the experiment, the test collection is randomly split into two parts. The first part is used as the initial dataset for the vectorization. The remaining part is added afterwards and vectorized with respect to the initial dataset. I.e. the vectorization of the new songs does not alter the vector representations of the initial songs. Performance is computed for each possible splitting ratio and averaged over 100 random splits to reduce effects caused by the random assignment.

3. VECTORIZATION APPROACHES

3.1. Baseline

Vectorization by interpreting the rows of the distance matrix as feature vectors is considered as baseline here because it is the common practice in the literature. Using this approach, the vectors can directly be read from the distance matrix without computation. Whilst this is very straightforward, it is not immediately clear how the distance between the resulting vectors should be computed. Therefore, the following three common distance metrics are considered here:

1. Euclidean distance: $d(\vec{x},\vec{y}) = \sqrt{\sum_i (x_i - y_i)^2}$
2. Manhattan distance: $d(\vec{x},\vec{y}) = \sum_i |x_i - y_i|$
3. Cosine distance: $d(\vec{x},\vec{y}) = 1 - cos(\vec{x},\vec{y})$

3.2. Vectorization by Multidimensional Scaling

Given the distances (or dissimilarities) for a set of objects S and a parameter $0 < m \leq |S|$, metric Multidimensional Scaling (MDS) [11] finds an embedding into m-dimensional Euclidean space that maintains the objects' distances as good as possible. I.e. each object $o^{(i)} \in S$ is mapped to an m-dimensional vector $\vec{x}^{(i)} \in R^m$ such that the Euclidean distance between the vectors $\vec{x}^{(i)}$ and $\vec{x}^{(j)}$ is (as close as possible to) the original distance d_{ij} between the corresponding objects $o^{(i)}$ and $o^{(j)}$. Because of this, MDS naturally suggests itself as a method for vectorization.

Here, the following algorithm based on the description in [12, Chapter 15.2] is used to compute the MDS: First, a mean-centered matrix B is constructed from the squared distances by the following two steps:

1. Define matrix $A = (a_{ij})$ with $a_{ij} = -\frac{1}{2} d_{ij}^2$
2. Define matrix $B = (b_{ij})$ with $b_{ij} = a_{ij} - \overline{a}_i - \overline{a}_j + \overline{a}$ for the row mean $\overline{a}_i = \frac{1}{n}\sum_{j=1}^{n} a_{ij}$, the column mean $\overline{a}_j = \frac{1}{n}\sum_{i=1}^{n} a_{ij}$, and the overall mean $\overline{a} = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}$ of A.

The m-dimensional coordinate vectors $\vec{x}^{(i)}$ can then be computed as $\vec{x}^{(i)} = \gamma_i\sqrt{\lambda_i}$ from the m biggest positive eigenvalues λ_i of B and the respective eigenvectors γ_i. Naturally, the parameter m is bounded by the dimensionality of B and the number of positive eigenvalues.

The (relative) difference between the sum of the m used eigenvectors and the trace of the matrix B (i.e. the sum of all its eigenvalues) will in the following be called *residual*. It captures, how much distance information could not be preserved in the output space. The trade-off between the number of vector dimensions and the quality of the vectorization can be controlled directly by providing a threshold r_{max} for the residual instead of having to choose m a priori. The algorithm then automatically selects the smallest m that results in a mapping with a residual $r \leq r_{max}$. The best possible mapping is obtained for $r = 0$ which also results in the highest number of output dimensions.

3.3. Vectorization per Facet

In the second test scenario described in Section 2.3.2, the facet weights and thus the aggregated distances between the objects (in the original space) change. This change has to be propagated somehow to the vector space. Re-computation of the vectorization can be avoided, if the adapted facet weights can also be applied in the vector space without loosing their semantics. This is only possible, if there is a direct correlation of the vector space dimensions and the facets. To this end, the objects have to be vectorized with respect to each single facet. The resulting vectors are then concatenated for each object. This meta-technique can be applied for any basic vectorization approach.

For the computation of the distances between the resulting vectors, two possibilities are considered:

1. The distance measure to be used in the output vector space is applied to the concatenated vectors using the weight of the corresponding facet *for each dimension*. I.e. for the MDS this means to compute the weighted Euclidean distance:

$$d(\vec{x},\vec{y}) = \sqrt{\sum_i w_i (x_i - y_i)^2} \qquad (1)$$

Table 1. Performance comparison of the different vectorization approaches. All values in percent except dim. *Selected for further comparisons.

vectorization approach	dim	triples agreement	nearest neighbors closest	nearest neighbors top 10
baseline (Euclidean distance)	197	76.8	45.7	54.4
baseline (Manhattan distance)	197	72.7	37.6	44.3
baseline (Cosine distance)*	197	78.9	54.8	60.7
baseline vectorized per facet (Eucl.)	3940	84.4	47.2	57.6
baseline vectorized per facet (Man.)	3940	85.9	52.8	63.6
baseline vectorized per facet (Cos.)	3940	82.2	46.2	55.7
MDS	58	96.1	78.2	87.2
MDS vectorized per facet	1051	92.5	71.6	80.7
MDS vec. & aggregated per facet	1051	98.3	94.4	94.1

2. The distance measure to be used in the output vector space is applied to compute the distance *for each facet* individually. These facet distances are then aggregated in a weighted sum using the respective weights:

$$d(\vec{x},\vec{y}) = \sum_f w_f d_f(\vec{x},\vec{y}) \quad (2)$$

where $d_f(\vec{x},\vec{y})$ is the distance of the vectors $\vec{x}$ and $\vec{y}$ taking only into account the dimensions that correspond to facet f.

In the second case which will be referred to as "aggregated per facet", the same aggregation function (i.e. the weighted sum) as for the original object facets is used. Only the specific facet distance measures are substituted by the distance metric of the vector space that now computes each facet distance from the corresponding vector dimensions.

4. RESULTS

4.1. Vectorizing a Fixed Dataset

Table 1 shows the performance of the different vectorization approaches for this typical use-case. Additionally to the evaluation measures, it contains the number of output dimensions. The baseline approach obviously produces vectors with 197 dimensions as this is the size of the dataset. The probably most important observation here is that the baseline vectorization approach does actually work reasonably well with an agreement of more than 75% for the triples and between 50% and 60% for the neighborhoods. The most suitable distance metric in the vector space is – quite surprisingly – the cosine distance which especially does much better preserve the neighborhoods than the Euclidean or Manhattan distance. It is therefore considered as baseline in further comparisons.

Combining the baseline with the meta-technique for per-facet vectorization described in Section 3.3 mainly has an impact on the triples agreement. Surprisingly, the Cosine distance now performs worse whereas the Manhattan distance

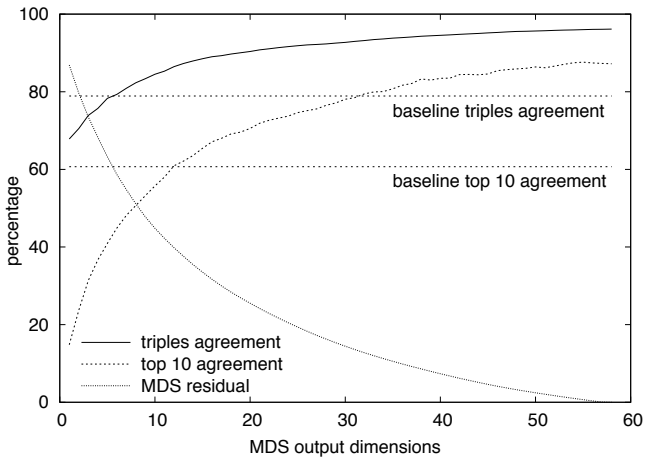

Fig. 1. Performance of the MDS vectorization depending on the number of dimensions of the output space.

appears to be the best choice. However, the small improvement in performance hardly justifies the remarkable increase of vector dimensions which is now the dataset size times the number of facets. Applying Equation 2 for distance computation in the vector space does not lead to an improvement for any of the considered distance metrics (omitted in Table 1).

Using the basic MDS approach increases performance by about 20% (absolute). Again, the agreement is significantly higher for the triples than for the neighborhoods. Most importantly, the number of output dimensions is much less than for the baseline. As Figure 1 illustrates, MDS already outperforms the baseline performance using only 13 output dimensions. The plot also shows that using more than 40 dimensions which roughly corresponds to a residual of 10% does not lead to much performance improvement anymore.

Looking at Table 2 confirms the superior performance of the MDS approach but also reveals that there is a high variance in the number of dimensions required per facet. The maximum number of dimension is needed for the distribution of MFCCs ("audio:mandelellis") which is a very complex feature. The worst performance is obtained for "production:date". This feature is a set of (time) intervals and for distance computation a rather complicated measure is used which does not conform to the triangle inequality and thus is not a metric. For such non-metric cases, special variants of MDS exist as described, e.g., in [12, Chapter 15.3] that could be applied but this is beyond the scope of this paper. Interestingly, the (metric) MDS approach still performs significantly better than the baseline. This could be indicating that the baseline approach should be used only for distance metrics though there is not sufficient evidence to support this hypothesis. Table 2 also shows that there seems to be a problem with text-based features (title and lyrics). However, the reason for this is not clear and requires further investigation.

Table 2. Single-facet MDS vectorization performance. (All values in percent except dim. Absolute improvement over baseline in brackets.)

facet name	dim	triples agreement		nearest neighbors closest		nearest neighbors top 10	
title	143	19.2	(0.9)	51.3	(6.6)	54.0	(10.4)
lyrics:text	110	89.3	(18.6)	55.8	(23.9)	69.9	(34.2)
creators	28	100.0	(29.0)	100.0	(8.1)	100.0	(12.0)
year	7	100.0	(18.9)	100.0	(0.0)	100.0	(0.0)
producer / engineer	26	100.0	(37.4)	100.0	(6.1)	100.0	(14.3)
production:location	54	100.0	(36.5)	100.0	(18.8)	100.0	(28.5)
production:date	60	16.1	(8.4)	36.0	(-3.6)	71.3	(36.5)
instruments	109	99.4	(32.3)	95.4	(36.0)	96.1	(46.6)
musicians:all	55	100.0	(27.5)	100.0	(17.3)	100.0	(29.1)
musicians:lead vox	11	100.0	(25.7)	100.0	(1.5)	100.0	(4.1)
musicians:bg vox	16	100.0	(39.5)	100.0	(4.6)	100.0	(6.8)
musicians:guitars	10	100.0	(25.4)	100.0	(1.5)	100.0	(4.5)
musicians:bass	7	100.0	(3.3)	100.0	(0.5)	100.0	(5.1)
musicians:drums	8	100.0	(3.2)	100.0	(2.0)	100.0	(4.5)
audio:mandelellis	196	100.0	(17.9)	100.0	(7.6)	100.0	(9.7)
audio:fluctuation	101	100.0	(11.3)	100.0	(62.4)	100.0	(44.5)
audio:marsyas07	57	100.0	(23.0)	100.0	(67.0)	100.0	(52.6)
key	14	84.3	(7.4)	99.0	(0.5)	98.6	(5.9)
chords:variety	14	100.0	(7.6)	100.0	(0.0)	100.0	(1.3)
chords:distribution	25	100.0	(28.8)	100.0	(37.1)	100.0	(27.5)

The concatenation of the vectors obtained separately for each facet results in 1051 dimensions – more than 5 times the value for the baseline but significantly less than the one for the baseline vectorized per facet. Compared to the basic MDS approach, there is only a small increase of the triples agreement (which was already close to the maximum anyways) but also a significant improvement of the neighborhood preservation close to maximum. This holds, however, only if Equation 2 is used. In fact, Equation 1 performs worse than the basic MDS approach.

4.2. Adapting Facet Weights

As pointed out in Section 3.3, this is the intended application scenario of the per-facet vectorization approach. Table 3 shows that the approach indeed correctly propagates facet weight changes into the vector space for the MDS vectorization so that a values close to the initial performance (c.f. Table 1) are obtained. Again, distance computation using Equation 2 is superior – especially in preserving the neighborhoods. Very surprisingly, the technique seems not to work for the baseline per-facet vectorization (i.e. with 3940 dimension). The learned weights[4] (as any weighting other than the uniform one) further degrade the performance. A reason for this could lie in the high dimensionality of the vector space for this case but this has to be investigated further.

[4]Note that the weights are learned using the original object representation – i.e. independent of the vectorization – and thus are the same for all vectorization approaches as described in Section 2.3.2. Therefore, an inappropriate weighting cannot be the reason for the poor performance of the baseline.

Table 3. Performance of the per-facet vectorization approaches in a scenario where initial distance facet weights (uniform) are adapted by a learning algorithm according to a user's distance judgments. Mean (and standard deviation in brackets) over 100 random user weightings. Top row: performance before and after adaptation in original space (i.e. expected best achievable value). Bottom rows: performance against uniform weighting (c.f. Table 1) for comparison, and before and after propagation of the weight change.

vectorization approach	comparison	triples agreement		nearest neighbors closest		nearest neighbors top 10	
no vectorization	unadapted	82.3	(3.6)	42.2	(10.0)	55.5	(7.0)
	adapted	100.0	(0.0)	100.0	(0.0)	100.0	(0.0)
baseline per facet (Manhattan dist.)	vs uniform	85.9		52.8		63.6	
	unadapted	77.1	(3.8)	31.9	(8.6)	44.9	(6.6)
	adapted	76.9	(3.9)	31.8	(8.6)	44.7	(6.6)
MDS (Equation 1)	vs uniform	92.5		71.6		80.7	
	unadapted	80.4	(3.9)	38.0	(8.4)	51.8	(6.9)
	adapted	92.5	(0.9)	70.8	(5.8)	80.1	(1.7)
MDS (Equation 2)	vs uniform	98.3		94.4		94.1	
	unadapted	82.2	(3.6)	41.7	(10.5)	55.3	(7.3)
	adapted	98.2	(1.3)	88.1	(8.3)	93.8	(4.4)

4.3. Adding New Songs

As motivated in Section 2.3.3, the distance relations and neighborhoods for the new songs are of interest here. The change of the distance triples agreement with increasing number of new songs (and simultaneously decreasing initial collection size) is shown in Figure 2 (top). The plot can be divided into two sections: Up to a ratio of roughly 4:1 (4 times more new songs than in the initial collection), there is almost no decrease in performance for all vectorization approaches. In this section, the agreements is close to what can be achieved by vectorizing the whole collection (c.f. Section 4.1). For higher ratios, the performance decreases – most significantly for the basic MDS vectorization. The other approaches are hardly affected up to a rather extreme 10:1 ratio.

Similar behavior can be observed for the top 10 agreement in Figure 2 (bottom). Here, only the very left region looks different. This is because the ranked lists considered for the top 10 agreement are very short due to the small number of new songs. Therefore, the performance values in this region cannot be considered significant.

5. CONCLUSIONS

In applications where data vectorization is unavoidable, it is important to be aware of the effect this transformation may have on the characteristics of the data. In this paper, a general methodology that can be applied to arbitrary dataset has been proposed for how to assess changes in distance relations and neighborhoods – two important characteristics in

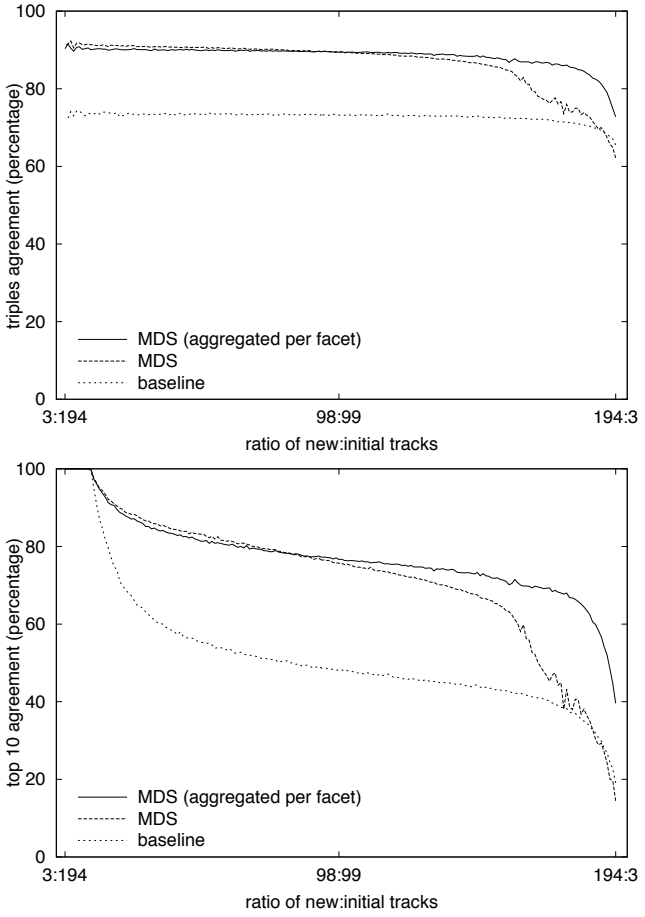

Fig. 2. Performance degradation with increasing portion of new songs added after the vectorization of an initial collection (measured for the new songs). Mean values over 100 random splits for each ratio.

retrieval applications. Using this methodology, the current common practice of vectorizing a dataset as the row vectors of a distance (or dissimilarity) matrix has been empirically evaluated on a multi-facet test collection. The experiments have been motivated by real-world applications: Apart from the common scenario of a fixed dataset, the adaptation of facet weights (and thus changes of distances) and changes of the dataset have been addressed. Furthermore, an alternative approach based on Multidimensional Scaling (MDS) has been proposed which shows significantly better performance while requiring far less vector dimensions.

6. ACKNOWLEDGMENTS

This work was supported in part by the German National Merit Foundation and the German Research Foundation (DFG) under the project AUCOMA.

7. REFERENCES

[1] D. Schnitzer, A. Flexer, G. Widmer, and M. Gasser, "Islands of Gaussians: The Self Organizing Map and Gaussian Music Similarity Features," in *Proc. of 11th Int. Conf. on Music Information Retrieval (ISMIR'10)*, 2010, pp. 327–332.

[2] Elias Pampalk, *Computational Models of Music Similarity and their Application in Music Information Retrieval*, Ph.D. thesis, Vienna University of Technology, Vienna, Austria, March 2006.

[3] Peter Knees, Tim Pohle, Markus Schedl, and Gerhard Widmer, "Exploring Music Collections in Virtual Landscapes," *IEEE MultiMedia*, vol. 14, no. 3, pp. 46–54, 2007.

[4] Sebastian Stober and Andreas Nürnberger, "Towards user-adaptive structuring and organization of music collections," in *Proc. of 6th Int. Workshop on Adaptive Multimedia Retrieval (AMR'08)*, 2008.

[5] Dominik Lübbers and Matthias Jarke, "Adaptive multimodal exploration of music collections," in *Proc. of 10th Int. Conf. on Music Information Retrieval (ISMIR'09)*, 2009, pp. 195–200.

[6] Sebastian Stober and Andreas Nürnberger, "A multi-focus zoomable interface for multi-facet exploration of music collections," in *Proc. of 7th Int. Symposium on Computer Music Modeling and Retrieval (CMMR'10)*, 2010, pp. 339–354.

[7] Christopher Harte, Mark B. Sandler, Samer A. Abdallah, and Emilia Gómez, "Symbolic representation of musical chords: A proposed syntax for text annotations," in *Proc. of 6th Int. Conf. on Music Information Retrieval (ISMIR'05)*, pp. 66–71.

[8] Markus Schedl, "The CoMIRVA Toolkit for Visualizing Music-Related Data," Technical report, Johannes Kepler University Linz, 2006.

[9] Daniel McEnnis, Cory McKay, Ichiro Fujinaga, and Philippe Depalle, "jAudio: An feature extraction library.," in *Proc. of 6th Int. Conf. on Music Information Retrieval (ISMIR'05)*, pp. 600–603.

[10] Weiwei Cheng and Eyke Hüllermeier, "Learning similarity functions from qualitative feedback," in *Proc. of 9th Europ. Conf. on Advances in Case-Based Reasoning (ECCBR '08)*, 2008, pp. 120–134.

[11] J.B. Kruskal and M. Wish, *Multidimensional Scaling*, Sage, 1986.

[12] W. Härdle and L. Simar, *Applied multivariate statistical analysis*, Springer Verlag, 2007.