Localized Alternative Cluster Ensembles for Collaborative Structuring

Michael Wurst, Katharina Morik and Ingo Mierswa

University of Dortmund, Department of Computer Science
Baroperstr. 301, 44221 Dortmund, Germany
{wurst,morik,mierswa}@ls8.cs.uni-dortmund

Abstract. Personal media collections are structured in very different ways by different users. Their support by standard clustering algorithms is not sufficient. First, users have their personal preferences which they hardly can express by a formal objective function. Instead, they might want to select among a set of proposed clusterings. Second, users most often do not want hand-made partial structures be overwritten by an automatic clustering. Third, given clusterings of others should not be ignored but used to enhance the own structure. In contrast to other cluster ensemble methods or distributed clustering, a global model (consensus) is not the aim. Hence, we investigate a new learning task, namely learning localized alternative cluster ensembles, where a set of given clusterings is taken into account and a set of proposed clusterings is delivered. This paper proposes an algorithm for solving the new task together with a method for evaluation.

1 Introduction

Collaborative approaches allow users to share preferences and knowledge without requiring a common semantic or explicit coordination. Data-driven methods as link analysis for web search and collaborative filtering have proven to be successful despite their lack of a clear semantic. Furthermore, not requiring coordination is one of the key factors that led to the fast growth of the Internet, as users can contribute information completely independently of other users.

Recently, new applications emerged under this Web 2.0 paradigm. Systems as flikr or del.icio.us allow users to annotate items with arbritrary chosen tags. Such tags complement global properties, e.g. artist, album, genre, etc. for music collections used by media organizers as iTunes. In contrast to these global properties, many user-assigned tags are *local*, i.e. they represent the personal views of a certain user not aiming at a global structure or semantic.

While users tend to start the organization of their personal collection eagerly, they often end up with a large set of items which are not yet annotated and a structure which is too coarse. A major challenge for machine learning is to exploit such local information in order to enable other users to navigate and structure media collections.

If there are enough annotated items, classification learning can deliver a decision function φ which maps items x of the domain X to a class g in a set of classes G. New items will be classified as soon as they come in and the user has no burden of annotation any more. However, classification does not refine the structure. If there is no structure given yet, clustering is the method to choose. It creates a structure of groups G for the not yet annotated items $S \subseteq X$. Traditional clustering schemes do not take into account the structure which users already have built up. Semi-supervised clustering obeys given groupings [1,2], but it does not refine structures. Non-redundant data clustering creates alternative structures to a set of given ones [3]. Given a structure G for all items in the collection, it creates an alternative structure G' for all items. However, it does not focus on the not yet annotated items S but restructures also the items which were already carefully structured.

Non-redundant clustering is connected 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 [4] or lane finding in GPS data [5]. Although must-link constrained clustering reuse existing clustering, the label information will not be preserved. In addition, these approaches use a feature-based clustering instead of given input clusterings and are hence not applicable to our problem.

We may consider the structuring achieved so far a set of partitionings φ_i, each mapping S to a set of groups G_i. Ensemble clustering then produces a consensus φ which combines these input partitionings [6]. This is almost what we need. However, there are three major drawbacks: first, all input clusterings must be defined at least on S. Second, the consensus model does not take the locality of S into account. Finally, merging several heterogenous user clusterings by a global consensus does not preserve valuable label information.

In many current applications it is important to consider structures of several users who interact in a network, each offering a clustering $\varphi_i : S_i \rightarrow G_i$. A user with the problem of structuring her left-over items S might now exploit the cluster models of other users in order to enhance the own structure. Distributed clustering learns a global model integrating the various local ones [7]. However, this global consensus model again destroys the structure already created by the user and does not focus on the set S of not appropriately structured items.

Whether own partial clusterings or those of other peers in a network are given, the situation is the same: current clustering methods deliver a consensus model overwriting the given ones and do not take into account S. In addition, users might want to select among proposed models which the learner delivers. The practical need of the user in organizing her media collection is not yet covered by existing methods. The situation we are facing is actually a new learning task.

Let X denote the set of all possible items. A function $\varphi : S \rightarrow G$ is a function that maps objects $S \subseteq X$ to a (finite) set G of groups. The set Φ contains all possible functions φ. We denote the domain of a function φ with D_φ. In cases where we have to deal with overlapping and hierarchical groups, we denote the set of groups as 2^G.

Definition 1 (Localized Alternative Cluster Ensembles) *Given a set $S \subseteq X$, a set of input functions $I \subseteq \{\varphi_i : S_i \rightarrow G_i\}$, and a quality function*

$$q : 2^{\Phi} \times 2^{\Phi} \times 2^{S} \rightarrow R \tag{1}$$

with R being partially ordered[1] LOCALIZED ALTERNATIVE CLUSTERING ENSEMBLES *delivers the output functions $O \subseteq \{\varphi_i | \varphi_i : S_i \rightarrow G_i\}$ so that $q(I, O, S)$ is maximized and for each $\varphi_i \in O$ it holds that $S \subseteq D_{\varphi_i}$.*

Note that in contrast to cluster ensembles, the input clusterings can be defined on any subset S_i of X. Since for all $\varphi_i \in O$ it must hold that $S \subseteq D_{\varphi_i}$, all output clusterings must at least cover the items in S.

We present a method solving this task in two steps: a base algorithm (Section 2.1) which is enhanced to become a hierarchical clustering in Section 2.2. The method is well suited for distributed clustering (Section 3) and we present the application from which the work originated (Section 3.1). Based on actual structures of music collections we can evaluate our approach in a way similar to that of evaluating supervised learning tasks (Section 4).

2 An Approach to Localized Alternative Cluster Ensembles

In the following, we describe a clustering method, that is based on the idea of bag of clusterings: deriving a new clustering from existing ones by extending the existing clusterings and combining them such, that each of them covers a subset of objects in S. In order to preserve existing label information but allowing the group mapping for new objects we define the extension of functions φ_i:

Definition 2 (Extended function) *Given a function $\varphi_i : S_i \rightarrow G_i$, the function $\varphi'_i : S'_i \rightarrow G_i$ is an* EXTENDED FUNCTION *for φ_i, if $S_i \subset S'_i$ and $\forall x \in S_i : \varphi_i(x) = \varphi'_i(x)$.*

Extended functions allow us to define a bag of extensions of non-overlapping originally labeled subsets that covers the entire collection:

Definition 3 (Bag of clusterings) *Given a set I of functions. A* BAG OF CLUSTERINGS *is a function*

$$\varphi_i(x) = \begin{cases} \varphi'_{i1}(x), & \text{if } x \in S'_{i1} \\ \vdots & \vdots \\ \varphi'_{ij}(x), & \text{if } x \in S'_{ij} \\ \vdots & \vdots \\ \varphi'_{im}(x), & \text{if } x \in S'_{im} \end{cases} \tag{2}$$

where each φ'_{ij} is an extension of a $\varphi_{ij} \in I$ and $\{S'_{i1}, \ldots, S'_{im}\}$ partitioning S.

[1] For example, $R = \mathbb{R}$ if one is interested in a unique solution.

Since each φ'_{ij} is an extension of an input clustering φ_{ij} on a subset S_{ij}, the label information is preserved. Now, we can define the objective function for our bag of clusterings approach to local alternative clustering ensembles.

Definition 4 (Quality of an output function) *The* QUALITY OF AN INDIVIDUAL OUTPUT FUNCTION *is measured as*

$$q^*(I, \varphi_i, S) = \sum_{x \in S} \max_{x' \in S_{ij}} sim(x, x') \text{ with } j = h_i(x) \tag{3}$$

where sim is a similarity function $sim : X \times X \rightarrow [0, 1]$ *and* h_i *assigns each example to the corresponding function in the bag of clusters* $h_i : S \rightarrow \{1, \ldots, m\}$ *with*

$$h_i(x) = j \Leftrightarrow x \in S'_{ij}. \tag{4}$$

The QUALITY OF A SET OF OUTPUT FUNCTIONS *now becomes*

$$q(I, O, S) = \sum_{\varphi_i \in O} q^*(I, \varphi_i, S). \tag{5}$$

Besides optimizing this quality function, we want to cover the set S with a bag of clusterings that contains as few clusterings as possible.

2.1 The Algorithm

In the following, we present a greedy approach to optimizing the bag of clusterings problem. The main task is to cover S by a bag of clusterings φ. The basic idea of this approach is to employ a sequential covering strategy. In a first step, we search for a function φ_i in I that best fits the set of query objects S. For all objects not sufficiently covered by φ_i, we search for another function in I that fits the remaining points. This process continues until either all objects are sufficiently covered, a maximal number of steps is reached, or there are no input functions left covering the remaining objects. All data points that could not be covered are assigned to the input function φ_j containing the object which is closest to the one to be covered. Alternative clusterings are produced by performing this procedure several times using each input function at most once.

We now have to formalize the notion of a function sufficiently covering an object and a function fitting a set of objects such that the quality function is optimized. When is a data point sufficiently covered by an input function so that it can be removed from the query set S? We define a threshold based criterion for this purpose:

Definition 5 *A function* φ SUFFICIENTLY COVERS *a object* $x \in S$ *(written as* $x \sqsubset_\alpha \varphi$ *), iff* $x \sqsubset_\alpha \varphi :\Leftrightarrow \max_{x' \in Z_\varphi} sim(x, x') > \alpha$.

The set Z_{φ_i} of items is delivered by φ. This threshold allows us to balance the quality of the resulting clustering and the number of input clusters. A small value of α allows a single input function to cover many objects in S. This, on average, reduces the number of input functions needed to cover the whole query set. However, it may also reduce the quality of the result, as the algorithm covers many objects in a greedy manner, which could be covered better using an additional input function.

Turning it the other way around: when do we consider an input function to fit the items in S well? First, it must contain at least one similar object for each object in S. This is essentially what is stated in the quality function q^*. Second, it should cover as few additional objects as possible. This condition follows from the locality demand. Using only the first condition, the algorithm would not distinguish between input functions which span a large part of the data space and those which only span a small local part. This distinction, however, is essential for treating local patterns in the data appropriately. The situation we are facing is similar to that in information retrieval. The target concept S – the ideal response – is approximated by φ delivering a set of items – the retrieval result. If all members of the target concept are covered, the retrieval result has the highest recall. If no items in the retrieval result are not members of S, it has the highest precision. We want to apply precision and recall to characterize how well φ covers S. We can define

$$prec(Z_{\varphi_i}, S) = \frac{1}{|Z_{\varphi_i}|} \sum_{z \in Z_{\varphi_i}} max\, \{sim(x, z) | x \in S\} \tag{6}$$

and

$$rec(Z_{\varphi_i}, S) = \frac{1}{|S|} \sum_{x \in S} max\, \{sim(x, z) | z \in Z_{\varphi_i}\}. \tag{7}$$

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 an input function and a set of objects now becomes a continuous f-measure:

$$q_f^*(Z_{\varphi_i}, S) = \frac{(\beta^2 + 1) rec(Z_{\varphi_i}, S) prec(Z_{\varphi_i}, S)}{\beta^2 rec(Z_{\varphi_i}, S) + prec(Z_{\varphi_i}, S)}. \tag{8}$$

Recall directly optimizes the quality function q^*, precision ensures that the result captures local structures adequately. The fitness $q_f^*(Z_{\varphi_i}, S)$ balances the two criteria.

Deciding whether φ_i fits S or whether an object $x \in S$ is sufficiently covered requires to compute the similarity between an object and a cluster. If the cluster is represented by all of its objects ($Z_{\varphi_i} = S_i$, as usual in single-link agglomerative clustering), this central step becomes inefficient. If the cluster is represented by exactly one point ($|Z_{\varphi_i}| = 1$, a centroid in k-means clustering), the similarity calculation is very efficient, but sets of objects with irregular shape, for instance, cannot be captured adequately. Hence, we adopt the representation by "well scattered points" Z_{φ_i} as representation of φ_i [8], where $1 < |Z_{\varphi_i}| < |S_i|$. These points are selected by stratified sampling according to G.

$O = \emptyset$
$I' = I$
while $(|O| < max_{alt})$ **do**
  $S' = S$
  $B = \emptyset$
  $step = 0$
  while $((S' \neq \emptyset) \wedge (I' \neq \emptyset) \wedge (step < max_{steps}))$ **do**
    $\varphi_i = \arg\max_{\varphi \in J} q_f^* (Z_\varphi, S')$
    $I' = I' \setminus \{\varphi_i\}$
    $B = B \cup \{\varphi_i\}$
    $S' = S' \setminus \{x \in S' | x \sqsubset_\alpha \varphi_i\}$
    $step = step + 1$
  end while
  $O = O \cup \{bag(B, S)\}$
end while

Fig. 1. The sequential covering algorithm finds bag of clusterings in a greedy manner. max_{alt} denotes the maximum number of alternatives in the output, max_{steps} denotes the maximum number of steps that are performed during sequential covering. The function *bag* constructs a bag of clusterings by assigning each object $x \in S$ to the function $\varphi_i \in B$ that contains the object most similar to x.

We can now dare to compute the fitness q_f^* of all $Z_{\varphi_i} \in I$ with respect to a query set S in order to select the best φ_i for our bag of clusterings. The whole algorithm works as depicted in figure 1. We start with the initial set of input functions I and the set S of objects to be clustered. In a first step, we select an input function that maximizes $q_f^*(Z_{\varphi_i}, S)$. φ_i is removed from the set of input functions leading to a set I'. For all objects S' that are not sufficiently covered by φ_i, we select a function from I' with maximal fit to S'. This process is iterated until either all objects are sufficiently covered, a maximal number of steps is reached, or there are no input functions left that could cover the remaining objects. All input functions selected in this process are combined to a bag of clusters, as described above. Each object $x \in S$ is assigned to the input function containing the object being most similar to x. Then, all input functions are extended accordingly, again by nearest-neighbor classification (cf. definition 2). We start this process anew with the complete set S and the reduced set I' of input functions until the maximal number of alternatives is reached.

As each function is represented by a fixed number of representative points, the number of similarity calculations performed by the algorithm is linear in the number of query objects and in the number of input functions, thus $O(|I||S||Z_{\varphi_i}|)$. The same holds for the memory requirements.

2.2 Hierarchical Matching

A severe limitation of the algorithm described so far is, that it can only combine complete input clusterings. In many situations, a combination of partial clusterings or even individual clusters would yield a much better result. This is especially true, if local patterns are to be preserved being captured by maximally specific concepts. Moreover, the algorithm does not yet handle hierarchies. Our motivation for this research was the structuring of media collections. Flat structures are not sufficient with respect to this goal. We cannot use a standard hierarchical clustering algorithm, since we still want to solve the new task of local alternative cluster ensembles. In the following, we extend our approach to the combination of partial hierarchical functions. A hierarchical function maps objects to a hierarchy of groups.

Definition 6 (Group hierarchy) *The set G_i of groups associated with a function φ_i builds a* GROUP HIERARCHY, *iff there is a relation $<$ such that $(g < g') :\Leftrightarrow (\forall x \in S_i : g' \in \varphi_i(x) \Rightarrow g \in \varphi_i(x))$ and $(G_i, <)$ is a tree. The function φ_i is then called a* HIERARCHICAL FUNCTION.

It should be possible to match functions that correspond to only a partial group hierarchy. We formalize this notion by defining a hierarchy on functions, which extends the set of input functions such that it contains all partial functions.

Definition 7 (Function hierarchy) *Two hierarchical functions φ_i and φ_j, are in* DIRECT SUB FUNCTION RELATION *$\varphi_i \prec \varphi_j$, iff $G_i \subset G_j$, $\forall x \in S_i : \varphi_i(x) = \varphi_j(x) \cap G_i$, and $\neg\exists \varphi_i' : G_i \subset G_i' \subset G_j$.*

Let the set I^* be the set of all functions which can be achieved following the direct sub function relation starting from I, thus

$$I^* = \{\varphi_i | \exists \varphi_j \in I : \varphi_i \prec^* \varphi_j\} \tag{9}$$

where $\prec^*$ is the transitive hull of $\prec$. While it would be possible to apply the same algorithm as above to the extended set of input functions I^*, this would be rather inefficient, because the size of I^* can be considerably larger than the one of the original set of input functions I. We therefore propose an algorithm which exploits the function hierarchy and avoids multiple similarity computations. Each function $\varphi_i \in I^*$ is again associated with a set of representative objects Z_{φ_i}. We additionally assume the standard taxonomy semantics:

$$\varphi_i \prec \varphi_j \Rightarrow Z_{\varphi_i} \subseteq Z_{\varphi_j}. \tag{10}$$

Now, the precision can be calculated recursively in the following way:

$$prec(Z_{\varphi_i}, S) = \frac{|Z^*_{\varphi_i}|}{|Z_{\varphi_i}|} prec(Z^*_{\varphi_i}, S) + \sum_{\varphi_j \prec \varphi_i} \frac{|Z_{\varphi_j}|}{|Z_{\varphi_i}|} prec(Z_{\varphi_j}, S) \tag{11}$$

where $Z^*_{\varphi_i} = Z_{\varphi_i} \setminus \bigcup_{\varphi_j \prec \varphi_i} Z_{\varphi_j}$. For recall a similar function can be derived. Note, that neither the number of similarity calculations is greater than in the base version of the algorithm nor are the memory requirements increased.

Moreover, the bottom-up procedure also allows for pruning. We can optimistically estimate the best precision and recall, that can be achieved in function hierarchy using all representative objects Z_e for which the precision is already known. The following holds:

$$prec(Z_{\varphi_i}, S) \leq \frac{|Z_e| prec(Z_e, S) + |Z_{\varphi_i} \setminus Z_e|}{|Z_{\varphi_i}|} \tag{12}$$

with $Z_e \subset Z_{\varphi_i}$. An optimistic estimate for the recall is one. If the optimistic f-measure estimate of the hierarchy's root node is worse than the current best score, this hierarchy does not need to be processed further. This is due to the optimistic score increasing with $|Z_{\varphi_i}|$ and $|Z_{\varphi_i}| > |Z_{\varphi_j}|$ for all sub functions $\varphi_j \prec \varphi_i$. No sub-function of the root can be better than the current best score, if the score of the root is equal or worse than the current best score.

This conversion to hierarchical cluster models concludes our algorithm for Local Alternative Cluster Ensembles (LACE).

3 A Distributed Algorithm

The LACE algorithm is well suited for distributed scenarios. We assume a set of nodes connected over an arbitrary communication network. Each node has one or several functions φ_i together with the sets S_i. If a node A has a set of objects S to be clustered, it queries the other nodes and these respond with a set of functions. The answers of the other nodes form the input functions I. A computes the output O for S. The node B being queried uses its own functions φ_i as input and determines the best fitting φ_i for S and sends this output back to A. The algorithm is the same for each node. Each node executes the algorithm independently of the other nodes.

We introduce three optimizations to this distributed approach. First, given a function hierarchy, each nodes returns exactly one optimal function in the hierarchy. This reduces the communication cost, without affecting the result, because any but the optimal function would not be chosen anyway (see pruning in the last section).

Second, input functions returned by other nodes can be represented more efficiently by only containing the items in the query set, that are sufficiently covered by the corresponding function. Together with the f-measure value q_f^* (equation 8) for the function, this information is sufficient for the querying node in order to perform the algorithm.

In many application areas, we can apply a third optimization. If objects are uniquely identified, such as audio files, films, web resources, etc. they can be represented by these IDs only. In this case, the similarity between two objects is 1, if they have the same ID, and 0 otherwise. A distributed version of our algorithm only needs to query other nodes using a set of IDs. This reduces the communication cost and makes matching even more efficient. Furthermore, such queries are already very well supported by current (p2p) search engines.

In a distributed scenario, network latency and communication cost must be taken into account. If objects are represented by IDs, both are restricted to an additional effort of $O(|S| + |I^*|)$. Thus, the algorithm is still linear in the number of query objects.

3.1 Distributed Media Management

The LACE algorithm is applied within Nemoz[2], a distributed media organization system which focuses on the application of data mining in p2p networks. It supports users in structuring their private media collections by exploiting information from other peers. Each user may create arbitrary, personal classification schemes to organize her media, e.g. music. For instance, some users structure their collection according to mood and situations, others according to genres, etc. Some such structures overlap, e.g., the blues genre may cover similar music as does the melancholic mood.

Nemoz supports the users in structuring their media objects while not forcing them to use the same set of concepts or annotations. If an ad hoc network has been established, peers support each other in structuring. A user who needs to structure a set of media objects S (e.g., refining an over-full node in her taxonomy) invokes the distributed algorithm described above. Then, the system offers a set of alternative clusterings, each combined from peers' response and covering S. The user chooses which of the clusterings she wants to incorporate into her collection's structure. Note, that in this scenario, the enhanced functions from definition 2 become particularly meaningful – she receives recommendations for similar music in addition to her own set S!

4 Experiments

The evaluation of LACE 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 (functions $\varphi_1, ..., \varphi_{39}$) and overall 1886 songs [9][3]. All experiments described in this paper were performed with the machine learning environment YALE [10][4].

The evaluation of LACE is performed by subsequently leaving out one function φ_i of the dataset. Then we apply clustering to reconstruct this taxonomy. Hence, we can evaluate cluster models in a way similar to classification learning. We have a "ground truth" available. A user taxonomy φ is compared with a taxonomy φ' created automatically by clustering as follows. We construct the usual tree distance matrix for the two taxonomies and compare these matrices on all pairs of objects in the set S. For the *absolute distance* criterion, the difference between the tree distance in φ and the one in φ' are summed-up and divided by the number of objects (see Table 1 for illustration).

As second criterion we use the *correlation* between these tree distances. Finally, for each cluster in the left-out taxonomy we search for the best corresponding cluster in the learned taxonomy according to f-measure. The average performance over all user-given clusters is then used as the (*FScore*) evaluation measure [11]. Note, that although we report the FScore, it is not normalized with

[2] Available at http://www.sourceforge.net/projects/nemoz
[3] Available at http://www-ai.cs.uni-dortmund.de/audio.html
[4] Available at http://yale.sf.net.

S	x_1	x_2	...	x_m	sum of differences
x_1	-	φ:1;φ':3			2+
x_2		-		φ:1;φ':2	1+
...			-		
x_m				-	
Total					3+

Table 1. Tree distance matrix indicating for all pairs of items in S how many edges they are away from each other, once concerning the hierarchy of φ and once concerning the hierarchy of φ'. For instance, in φ there is only one edge between x_1 and x_2, but in φ, there are three. The last columns sums-up the differences between the distances in φ and φ' for one item with respect to all other items. The last field gives the total of all differences. Total/m gives the absolute distance of φ and φ'.

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

Table 2. The results for different evaluation measures.

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 taxonomy.

We compare our approach with single-link agglomerative clustering using cosine measure, top down divisive clustering based on recursively applying kernel k-means [12] (TD), and with random clustering. Localized Alternative Cluster Ensembles were applied using cosine similarity as inner similarity measure. TD and random clustering were started five times with different random initializations. We use a set of 20 features which were shown to work well in a wide range of applications [13] as underlying audio features. Since, here, we want to test the new clustering method, we do not investigate different feature sets. The parameter β was set to 1.

Table 2 shows the results. As can be seen, the local alternative cluster ensembles approach LACE performs best. Note however, that absolute distance does not lead to results that are representative for agglomerative clustering as such, because it usually builds-up quite deep hierarchies, while the user constructed hierarchies were rather shallow.

A second experiment inspects the influence of the representation on the accuracy. The results of LACE with different numbers of instances at a node are shown in Table 3. Representing functions by all points performs best. Using a single centroid for representing a subtree leads to inferior results, as we already

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

Table 3. The influence of concept representation (cardinality of $|Z|$).

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 4. The influence of response set cardinality $|O|$.

expected. Well scattered points perform 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 output functions influences the quality of the result. The result should be clearly inferior with a decreasing number. Table 4 shows the result. On one hand, we observe that even with just one model, i.e. $|O| = 1$, LACE still outperforms the other methods with respect to tree distance. 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. Probably, the performance would increase even further for more output clusterings. Although a user still would select the best available clustering from all alternatives – which motivates this form of evaluation – the number of solutions should be rather small and was restricted to 5 in this setting.

5 Conclusion

Structuring media collections is one of the most important tasks for current and future media organization applications. Clustering is a basic technique for this problem. A correct or optimal clustering of items depends strongly on intentions and preferences of the user. An important challenge for new clustering techniques is the question of how to integrate clusterings provided by other users in a way that allows for a certain personalization which reflects the locality of the data and preserves user created clusterings. In contrast to other cluster ensemble methods or distributed clustering, a global model (consensus) is not the aim.

Investigating the practical needs carefully has led to the definition of a new learning task, namely learning localized alternative cluster ensembles, where a set of given clustering is taken into account and a set of proposed clusterings is delivered. We have formalized the learning task and developed a greedy approach solving it. Enhancements for hierarchical structures accomplish the LACE algorithm. It is well suited for distributed settings.

The performance of algorithms solving the localized alternative cluster ensembles task can be measured by a leave-one-structuring-out approach. The proposed algorithm outperforms standard clustering schemes on a real-world data set in the domain of music collections. We also investigated the influence of the number of representative points and the influence of response set cardinality which are important in distributed scenarios.

In our opinion, applications in the Web 2.0 context offer many interesting opportunities for machine learning. LACE is a very promising approach to overcome some of the problems associated with this new kind of applications.

References

1. Cohn, D., Caruana, R., McCallum, A.: Semi-supervised clustering with user feedback. Technical Report TR2003-1892, Cornell University (2000)
2. Finley, T., Joachims, T.: Supervised clustering with support vector machines. In: Proc. of the International Conference on Machine Learning. (2005)
3. Gondek, D., Hofmann, T.: Non-redundant data clustering. In: Proc. of the International Conference on Data Mining. (2004)
4. Hotho, A., Staab, S., Stumme, G.: Ontologies improve text document clustering. In: Proc. of the International Conference on Data Mining. (2003) 541–544
5. Wagstaff, K., Cardie, C., Rogers, S., Schroedl, S.: Constrained k-means clustering with background knowledge. In: Proc. of the International Conference on Machine Learning. (2001)
6. Strehl, A., Ghosh, J.: Cluster ensembles – a knowledge reuse framework for combining partitionings. In: Proc. of AAAI 2002, Edmonton, Canada. (2002)
7. Datta, S., Bhaduri, K., Giannella, C., Wolff, R., Kargupta, H.: Distributed data mining in peer-to-peer networks. IEEE Internet Computing, special issue on distributed data mining (2005)
8. Guha, S., Rastogi, R., Shim, K.: CURE: an efficient clustering algorithm for large databases. In: Proc. of ACM SIGMOD International Conference on Management of Data. (1998) 73–84
9. Homburg, H., Mierswa, I., Möller, B., Morik, K., Wurst, M.: A benchmark dataset for audio classification and clustering. In: Proc. of the International Symposium on Music Information Retrieval. (2005)
10. Fischer, S., Klinkenberg, R., Mierswa, I., Ritthoff, O.: Yale: Yet Another Learning Environment – Tutorial. Technical Report CI-136/02, Collaborative Research Center 531, University of Dortmund, Dortmund, Germany (2002)
11. Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques. In: Proc. of the KDD Workshop on Text Mining. (2000)
12. Dhillon, I.S., Guan, Y., Kulis, B.: Kernel k-means: spectral clustering and normalized cuts. In: Proc. of the conference on Knowledge Discovery and Data Mining. (2004)
13. Moerchen, F., Ultsch, A., Thies, M., Loehken, I., Noecker, M.and Stamm, C., Efthymiou, N., Kuemmerer, M.: Musicminer: Visualizing perceptual distances of music as topograpical maps. Technical report, Dept. of Mathematics and Computer Science, University of Marburg, Germany (2004)