User Oriented Hierarchical Information Organization and Retrieval

Korinna Bade, Marcel Hermkes, and Andreas Nürnberger

Otto-von-Guericke-University, D-39106 Magdeburg, Germany
{kbade,nuernb}@iws.cs.uni-magdeburg.de, marcel.hermkes@googlemail.com

Abstract. In order to organize huge document collections, labeled hierarchical structures are used frequently. Users are most efficient in navigating such hierarchies, if they reflect their personal interests. Thus, we propose in this article an approach that is able to derive a personalized hierarchical structure from a document collection. The approach is based on a semi-supervised hierarchical clustering approach, which is combined with a biased cluster extraction process. Furthermore, we label the clusters for efficient navigation. Besides the algorithms itself, we describe an evaluation of our approach using benchmark datasets.

1 Introduction

With the increasing number of data publicly available also the personal collections of documents have become larger. A useful personal organization of these files is necessary to allow efficient re-finding of information. Hierarchical folder structures have proven to be useful in the past, e.g. in personal file folders or library catalogs. These structures have the advantage that they provide at the same time a (categorized) overview of the collection and direct access to all documents therein. However, users are most efficient in navigating such hierarchies if they reflect their personal interests instead some generally applicable criteria.

The goal of the work presented in the following is to provide the user a tool for building and maintaining such a personal hierarchy. We consider the following scenario. A starting point for the user can be a completely unstructured collection. At this point, the system can provide the user with an initial although unpersonalized structure purely based on standard document similarities. Once the user starts with explicitly filing documents in his own personal structure, either by himself or assisted by the system, this information is used to adapt the structuring of the still unstructured part of the collection towards user specific structuring preferences. Furthermore, these preferences can be applied to other, external collections, which the user is viewing. This allows the user faster access to interesting information therein.

In this paper, we present and evaluate an approach that is capable of extracting such a personal structure, while having different amounts of previously structured data available. The approach consists of three main steps: hierarchical clustering, extraction of clusters from the obtained dendrogram, and labeling. Each step is presented in an own section, also including important related work.

J.N. Kok et al. (Eds.): ECML 2007, LNAI 4701, pp. 518–526, 2007.

2 Personalized Hierarchical Clustering

Our considered task is a two-fold semi-supervised hierarchical learning problem with having unlabeled documents as well as unknown classes. The predominant classes C in the collection are split into the set of known classes C_k and a set of unknown classes C_u. As a consequence, the given labeled documents D_k are only mapped to classes in C_k. The task of the algorithm is to map the unlabeled documents D_u to classes in $C = C_k \bigcup C_u$. This means that the algorithm either derives a mapping to a known class or extracts new classes by grouping similar documents and assigning a class label to this group. Furthermore, we assume hierarchical relations R_H between the classes in form of a tree structure. When structuring a collection into classes, the algorithm should preserve the existing structure R_{Hk} and extract the relations of discovered classes in C_U to each other and to the classes in C_k. Considering our user scenario, C_k and R_{Hk} are defined by the hierarchical filing system of the user. D_k are the documents, which were filed in the past. D_u consists of the documents, which are still unstructured.

Considering related work, semi-supervised clustering is often performed by constraint based clustering. Here, the supervised information is used to generate Must-Link and Cannot-Link constraint sets [9], which influence the clustering. Several approaches exist that either change the underlying similarity space or directly modify the clustering process itself, e.g. [9,6,10,2]. All these approaches search for a flat partitioning of the data, while we want to find a hierarchical structure. In principle, these algorithms could be applied recursively to create a hierarchy. However, partitioning algorithms require the number of clusters as input parameter, which is not known in our scenario and hard to determine automatically. Additionally, it needs to be determined on each hierarchy level. Therefore, we decided to use Hierarchical Agglomerative Clustering (HAC) that directly produces a hierarchical representation of the data by a dendrogram. Furthermore, hierarchical approaches produce more stable and more accurate results, especially if the data of the used collection is naturally hierarchical.

In our approach, labeled data is used to change the underlying similarity space of a HAC algorithm to express personal structuring preferences. We assume that the extracted features are sufficient to describe these preferences. In our current work, we restricted ourselves to content features, i.e. occurring terms in text documents. For each feature f_i, a weight w_i is computed that expresses its influence in the clustering. These weights are integrated in the cosine similarity measure: $sim(fv_1, fv_2, w) = \sum_i w_i \cdot fv_{1,i} \cdot fv_{2,i}$. In [1], we present a method to learn and apply these weights in detail. Its evaluation showed that feature weighting improves the initial clustering towards a user specific structure.

3 Cluster Extraction

The goal of cluster extraction is to compress the dendrogram representation to the most "meaningful" nested clusters, i.e. to the clusters describing classes in $C = C_k \bigcup C_u$. In our setting, "meaningful" is partially defined by the given

labeled data. However, it is usually rare, does not cover all classes and might be erroneous. Therefore, we first develop an unsupervised algorithm, which is then enhanced with labeled data.

An Unsupervised Approach. In published research, there is a common understanding that clusters could be extracted by two basic approaches. The dendrogram is either recursively cut with similarity thresholds or clusters are extracted on a node to node basis (e.g. by looking for significant changes in the merging similarity between a node (i.e. the similarity of its two child clusters) and its parent node). Both algorithm need a threshold as parameter. While the second approach can better handle different densities of sibling clusters, the first approach allows the use of a "global" criterion that helps in the extraction of less obvious clusters by using more obvious sibling clusters. Furthermore, it can also be used to always reduce the dendrogram, even without obvious sub-clusters. Nevertheless, cluster extraction is not widely discussed in the literature. To our knowledge, no work was published that dealt with this problem more thoroughly. Some work was done on extracting clusters from reachability plots produced by density based clustering (see [7,3]). The authors of [7] also show the similarities between reachability plots and dendrograms making it possible to apply their algorithms to dendrograms. However, these algorithms require specific assumptions that do not necessarily hold in our setting. The work in [7] works best with sharp cluster distinctions usually obtained, when data points are only assigned to leaf-clusters, which violates our problem definition. The work in [3] focused on extraction of narrowing sub-clusters. However, their approach requires a very smooth reachability plot, which is not produced in our application.

In this paper, we used a threshold approach. A recursive procedure is applied that starts at the root of the dendrogram and is repeated for each top node of an extracted cluster. A threshold t is computed in each iteration depending on the standard deviation σ of merging similarities in the considered sub-tree. The idea is to skip a top fraction of nodes with merging similarities that are "outstanding" from the others. As reference value the merging similarity of the current top node is used, which is also the minimum merging similarity of the whole sub-tree sim_{min}. t is computed by $sim_{min} + p \cdot \sigma$. While sim_{min} and σ are computed from the dendrogram, p is a parameter to determine the size of the fraction of minimal merging similarities to skip. Further parameters can restrict the cluster extraction, which are useful from the application point of view, i.e. the minimum number of items per cluster (preventing the extraction of too small clusters), the minimum difference in item size between a cluster and its parent cluster (preventing narrowing sub-clusters of being too similar to their parents), a minimum standard deviation (underneath it, merging similarities are supposed to be indistinguishable). Appropriate values for these are highly dependent on personal preference. Their values are not very crucial for the extraction process and adaptations to them can be made during interaction with the collection.

Using Supervision. The labeled data is used locally to make known class extraction more robust, i.e. avoiding the split of such a class. However, one has

to be cautious in doing so, as this "robustness" should not overextend onto unknown classes. Due to this, we use the labeled data in a post-processing step after the unsupervised extraction rather than integrating it directly. First, the extracted clusters are labeled with known classes, if possible, as described in Sec. 4. We then merge sibling clusters labeled equally. As simple merge, we create intermediary clusters for groups of at least two equally labeled siblings. More interesting is what we call a deep merge based on the original dendrogram. Here, the sibling nodes are merged by extracting the common ancestor node from the dendrogram. The idea is that other items that also belong to the common ancestor node, but were not extracted or labeled as such, also belong to the same class. This combines more items of one class. However, it only works correctly, if the initial clustering represents the desired cluster structure appropriately. This can be detected, if integrated clusters are labeled differently, in which case the merge is not performed. However, this can especially not be detected for unknown classes, making it vulnerable for mislabeling.

Additionally and maybe more important is the use of the labeled data for estimating initial parameter values. Especially if the user starts interacting with the collection, he might not know how to set the parameters appropriately. Once he has a first result, it is easier to adapt parameters according to preference. For estimation, we label the clusters in the dendrogram. For each labeled cluster, we use the distance in merging similarity between this cluster and the cluster labeled with the parent class to estimate p. The mean value of all estimates is the final estimate. Minimum cluster size and standard deviation could be set to the maximum value that still allows the extraction of all labeled clusters.

4 Cluster Labeling

Labeling extracted clusters is crucial for the effectiveness of the hierarchy as it guides the user in browsing it. A good label must be capable of summarizing the content of a cluster. At the same time, it must be very short. In a hierarchy, the label must also be able to distinguish a cluster from its sibling clusters. Furthermore, it must show the differences between the cluster and its parent cluster. Although there are different approaches to automatically extract good cluster labels, a label given by the user himself is most descriptive. Therefore, we try to reuse known labels, if possible, before computing user independent labels.

Labeling Clusters as Known Classes. Known classes can be identified with the labeled data. As it is rare and possibly erroneous, we cannot trust every single instance but can also not assume high support or confidence of a labeling decision. In our work, we use two parameters to deal with this problem and constrain the labeling. We require to have a minimum precision for one class in all labeled items of the cluster and a minimum number of items labeled as such. The higher we set the thresholds of these parameters the less errors we make. However, this also means labeling less data. Good parameter values depend on the available amount of labeled data and on the clustering quality.

The clusters are labeled in a recursive procedure starting from the root of the cluster tree. If a label following the defined criteria is found, it is assigned to the cluster. All sub-clusters of it are then restricted to be labeled with sub-labels. This ensures the consistency of the hierarchy of known classes. Furthermore, labels are propagated upwards during cluster merges.

Labeling Clusters of Unknown Classes. The basis for most existing approaches are term statistics. Unfortunately, most related work only considers a flat cluster environment, which makes them not necessarily applicable to a hierarchical structure. [5] dealt with hierarchies by distinguishing three different concepts: terms describing the cluster itself, terms that are more general and thus better describe the parent cluster, and terms that are more specific, describing a child cluster. The distinction between these three concepts is made by predefined thresholds on term frequencies. However, these thresholds are hard to determine. Furthermore, it is questionable, whether one threshold works for different hierarchy levels as the distribution of term frequencies might vary. [4] uses a linear function that combines different statistical features that include hierarchical labeling criteria. In contrast to our work, they try to learn weights for different features by using linear regression on the basis of a set of labeled data. Our approach also uses statistical measures. As [5] and [4], we integrate parent and child clusters to avoid several occurrences of the same label along pathes in the hierarchy. A score of descriptiveness $Des_{t,C}$ is computed for each term t and each cluster C mainly based on the (absolute) document frequencies $df_{t,C}$, i.e. the number of documents that contain t (see (1), (2)). Here, each cluster is handled as containing all documents assigned to it and its child clusters.

$$Des_{t,C} = \log\left(\frac{rank_P(df_{t,P})}{rank_C(df_{t,C})}\right) \cdot (1 - SI_{t,P} + SI_{t,C})/2 \tag{1}$$

$$SI_{t,C} = \begin{cases} 1 & \text{if } Child(C) = \emptyset \\ \left(\sum_{c_i \in Child(C)} -\frac{df_{t,c_i}}{df_{t,C}} \log_2 \frac{df_{t,c_i}}{df_{t,C}}\right) / \log_2 |Child(C)| & \text{else} \end{cases} \tag{2}$$

The first factor measures the boost in document frequency ranking of t in comparison to the parent cluster P (as $rank_C(df_{t,C})$ is the rank of t in an descending order according to the document frequency of t in C, as in [4]). This assures that terms get higher scores that were not already good descriptors for the parent cluster and are therefore too general for the current cluster. The second factor considers information on how the term is distributed in sibling and child nodes, expressed by SI, which is bound to $[0; 1]$. Terms occurring in several child clusters are favored by $SI_{t,C}$, while terms that are also descriptors of sibling clusters are penalized by $1 - SI_{t,P}$. For each cluster, n terms with highest descriptiveness are used as label. Unfortunately, our score cannot completely avoid that a term occurs several times along pathes through the cluster hierarchy (i.e. pathes from the root cluster to all leaf nodes). Therefore, we go through all such pathes in a post-processing step. If we encounter a term in the selected n descriptive labels occurring several times in a path, we remove it from the set of

descriptive labels in all clusters except the one with highest $Des_{t,C}$. All clusters now having less than n terms as label get added new terms by taking the next best descriptive terms from the initially computed list.

5 Evaluation

In this evaluation, the general performance of the algorithms is evaluated using two different datasets of web pages that simulate the problem. The first is the banksearch dataset [8] (see Fig. 1). The second was created by us by downloading parts of the open directory (www.dmoz.org). The properties can be summarized as: hierarchy depth 4, 3 to 16 direct child nodes per inner node, about 50 documents directly in each node, 2119 documents in total. All documents were represented with standard tfidf document vectors.

We evaluated different settings to simulate different user data. For both datasets, we evaluated a setting with 10 labeled documents per class, i.e. a classification scenario (settings (1), (5)). For the banksearch data, we also evaluated settings with unknown classes: (2) *Motor Sport*, (3) *Science*, (4) *Science* and *Sport*. As measure, we used the f-score gained in accordance to the given dataset, which is supposed to be the true user defined class structure that shall be recovered. For its computation in an unlabeled cluster tree, we followed a common approach that selects for each class in the dataset the cluster gaining the highest f-score on it. When evaluating cluster labeling, the f-score of known classes is determined based on all documents labeled as such. The unknown classes are again extracted as best f-score clusters, however only in hierarchy consistent unlabeled parts of the cluster tree.

As we already evaluated the baseline performance of the clustering algorithm in [1], we focus here on evaluating cluster extraction and labeling. The competitiveness of our approach for classification can briefly be shown by comparison with SVM. For the banksearch data, the SVM reaches a mean F-score of 0.6892, while our approach reaches 0.7570 on the dendrogram. For the open directory data, the SVM reaches 0.6198, while our approach reaches 0.6100. Hence, our algorithm has a good baseline performance.

In Tables 1 and 2, we evaluate our cluster extraction methods (CE - unsupervised extraction, SM - simple merge, DM - deep merge) in comparison to the baseline given by the dendrogram (DG). As we consider here only a single cluster per class, this evaluation shows how good the algorithms are in preserving the best cluster. We only varied p for cluster extraction as the other parameters only do pruning of the cluster tree. We set the minimum cluster size and the minimum

• **Finance** (0)	• **Programming** (0)	• **Science** (0)	• **Sport** (100)
◦ Commercial Banks (100)	◦ C/C++ (100)	◦ Astronomy (100)	◦ Soccer (100)
◦ Building Societies (100)	◦ Java (100)	◦ Biology (100)	◦ Motor Sport (100)
◦ Insurance Agencies (100)	◦ Visual Basic (100)		

Fig. 1. Class structure of the banksearch dataset

Table 1. F-Score for different cluster extraction methods using the banksearch data

Setting	DG	$p = 0.1$			$p = 0.05$			$p = 0.03$			$p = 0.01$		
		CE	SM	DM	CE	SM	DM	CE	SM	DM	CE	SM	DM
(1)	0.757	0.693	0.733	0.723	0.735	0.737	0.735	0.718	0.760	0.745	0.754	0.754	0.754
(2)	0.771	0.699	0.727	0.752	0.713	0.724	0.724	0.762	0.762	0.762	0.767	0.767	0.767
(3)	0.734	0.582	0.654	0.667	0.676	0.705	0.709	0.717	0.729	0.731	0.732	0.732	0.732
(4)	0.697	0.542	0.585	0.583	0.575	0.622	0.617	0.641	0.653	0.643	0.694	0.694	0.694

Table 2. F-Score open directory data for CE/SM/DM

Setting	(5)
DG	0.610
$p = 0.2$	0.551/0.581/0.568
$p = 0.1$	0.577/0.587/0.585
$p = 0.01$	0.586/0.590/0.587

Table 3. Estimation p

Setting	p
(1)	0.196
(2)	0.072
(3)	0.047
(4)	0.030
(5)	0.212

Table 4. F-Score after labeling

Setting	SM	DM	SM-l	DM-l
(1)	0.760	0.745	0.696	0.696
(2)	0.762	0.762	0.728	0.728
(3)	0.729	0.731	0.692	0.694
(4)	0.653	0.643	0.624	0.519
(5)	0.587	0.585	0.525	0.521

Table 5. Example labeling for the banksearch data

Class	Five selected terms
Banking	mortgage, savings, payments, debit, income
Commercial Banks	bank, depositor, internet, abbey, advert
Building Societies	society, interest, building, telegraphic, superseeded
Insurance Agencies	insurance, cover, claims, wording, policy

difference in cluster size between parent and child cluster to 10. The minimum standard deviation was set to 0. Increasing p leads in general to broader cluster trees and a fewer number of extracted clusters. A too high value for p therefore will split a "class cluster" into several clusters, causing a decrease in the f-score. There is always a value for p that can (almost) recover the best f-score clusters from the dendrogram, while highly condensing the dendrogram representation, shrinking the number of clusters from over 2000 to about 100 or less for the banksearch data and from over 4000 to 200 or less. Furthermore, it seems that good values for p are quite stable for different data. Its order of magnitude, which is quite low, is in our opinion given by the fact that we cluster high dimensional text data. Both merging methods are useful for getting back lost performance due to splits in the cluster tree with similar results. Although hypothesized differently by us, the deep merge seems not to be better. This suggest that a simple merge, which also requires less computation time, is a sufficient and therefore better choice. Table 3 shows the estimations for p as computed based on the labeled data. Although these values are not perfect, they provide a good initial starting point for the exploration of the cluster tree.

Table 4 evaluates the identification of given classes using a minimum precision of 0.6 and a minimum number of labeled items of 2. We used a fixed p of 0.03 for the banksearch settings and 0.1 for the open directory setting. Both merges are considered. Labeling f-score is always less than the best cluster f-score as the given labeled data is not sufficient to always identify the best clusters. In setting (4), the deep merge performs a lot worse than the simple merge as it overextends the label of the known class *Programming* onto the unknown *Science* class. This suggests that the deep merge might be problematic in the case of unknown classes. Nevertheless, the identification of existing classes works well in general.

Table 5 gives an inside on how the labeling of unknown classes works with a small example. The labeling algorithm was directly applied to the dataset hierarchies to compute class labels. In general, the manually chosen labels are in about 70% of the classes among the five selected terms. The computed terms seem quite descriptive for the classes. Nevertheless, a more thorough evaluation of the labeling method is still necessary.

6 Conclusion

In this paper, we presented an integrated approach that provides a personalized hierarchical cluster structure for a certain collection. The algorithm comprises of several steps, i.e. (1) do personalized HAC, (2) extract clusters unsupervised, (3) label clusters according to known classes, (4) merge clusters, and (5) label still unlabeled clusters. We evaluated each step and showed the validity of our approach. The algorithms presented as solutions to certain steps can also be applied in different settings and are not necessarily restricted to our application.

References

1. Bade, K., Nürnberger, A.: Personalized hierarchical clustering. In: Proceedings of the 2006 IEEE/WIC/ACM Int. Conference on Web Intelligence, pp. 181–187 (2006)
2. Basu, S., Banerjee, A., Mooney, R.: Active semi-supervision for pairwise constrained clustering. In: Proc. of SIAM Int. Conf. on Data Mining, pp. 333–344 (2004)
3. Brecheisen, S., Kriegel, H.P., Kröger, P., Pfeifle, M.: Visually mining through cluster hierarchies. In: Proc. of SIAM Int. Conf. on Data Mining, pp. 400–412 (2004)
4. Callan, J., Treeratpituk, P.: Automatically labeling hierarchical clusters. In: Proceedings of the 2006 International Conference on Digital Government Research. ACM International Conference Proceeding Series, vol. 151, pp. 167–176. ACM Press, New York (2006)
5. Glover, E., Pennock, D., Lawrence, S., Krovetz, R.: Inferring hierarchical descriptions. In: Proceedings of 11^{th} International Conference on Information and Knowledge Management, pp. 507–514 (2002)
6. Kim, H., Lee, S.: An effective document clustering method using user-adaptable distance metrics. In: Proceedings of the 2002 ACM symposium on Applied computing, pp. 16–20. ACM Press, New York (2002)

7. Sander, J., Qin, X., Lu, Z., Niu, N., Kovarsky, A.: Automatic extraction of clusters from hierarchical clustering representations. In: Advances in Knowledge Discovery and Data Mining: 7 th Pacific-Asia Conference (Proc.), pp. 75–87 (2003)
8. Sinka, M., Corne, D.: A large benchmark dataset for web document clustering. In: Soft Computing Systems: Design, Management and Applications, Frontiers in Artificial Intelligence and Applications, vol. 87, pp. 881–890 (2002)
9. Wagstaff, K., Cardie, C., Rogers, S., Schroedl, S.: Constrained k-means clustering with background knowledge. In: Proceedings of 18 th International Conference on Machine Learning, pp. 577–584 (2001)
10. Xing, E., Ng, A., Jordan, M., Russell, S.: Distance metric learning, with application to clustering with side-information. Advances in Neural Information Processing Systems 15, 505–512 (2003)