# Collection Browsing through Automatic Hierarchical Tagging

Korinna Bade and Marcel Hermkes

Otto-von-Guericke-University, D-39106 Magdeburg, Germany
korinna.bade@ovgu.de, marcel.hermkes@googlemail.com

**Abstract.** In order to navigate huge document collections efficiently, tagged hierarchical structures can be used. For users, it is important to correctly interpret tag combinations. In this paper, we propose the usage of tag groups for addressing this issue and an algorithm that is able to extract these automatically for text documents. The approach is based on the diversity of content in a document collection. For evaluation, we use methods from ontology evaluation and showed the validity of our approach on a benchmark dataset.

## 1 How to tag

When searching for information, structured access to data, e.g., as given by web directories or social tagging systems like del.icio.us can be very helpful. The goal of our work is to automatically provide such structure for unstructured collections. For this, we automatically tag text documents based on their content. We do not tag resources individually, but compute the tags collection based. By this, we can directly aim at supporting efficient browsing by adapting presented tags to a dynamically created collection, which, e.g., can be the result set of a query based search. Our approach first structures the collection by a hierarchical clustering algorithm and then tags clusters in the hierarchy. This results in a set of tags combined with hierarchical relations. The cluster tags are then assigned to the documents that belong to this cluster. In this paper, we focus on the extraction of tags once the cluster hierarchy was built. Information on the clustering process can be found in [1].

In todays tagging systems, resources are tagged with one or more single words to describe them. Access to resources is usually provided by tag clouds, which can be used to browse the existing tags. Between tags, no relations are usually assumed (an exception are hierarchical relations from bundle tags). However, a user might use multiple tags for two different reasons. First, he wants to provide synonyms such that more people find his resource. Second, he wants to show that this resource actually belongs to an overlap of several topics. While browsing with a single tag is sufficient for finding a resource of the first type, the second case requires combining more tags in the search. For cluster tags, it is even more important to know how different tags shall be interpreted. Two tags of a cluster could correspond either to the same document or to different documents. This

means that documents in the cluster could either belong to the intersection of two topics or that some documents in the cluster belong more to one topic and the others more to the other topic. As an example consider a cluster tagged with *Banking* and *Programming*. This can either mean that the cluster contains documents about banking software or that the cluster contains documents that deal with *Banking* and others that deal with *Programming*. To help in the interpretation of tags, our approach tries to group tags based on their relevance for documents. A group of tags means that all tags therein belong together in describing a document of this cluster. Such a group, therefore, contains synonyms as well as combined topics. Furthermore, tags in different groups are supposed to relate to different documents in the cluster. In the following, we write such a cluster tag as a set of tag groups, where each tag group is a set of tags. For the example above, we would have either the cluster tag $\{\{Banking, Programming\}\}$ or the cluster tag $\{\{Banking\}, \{Programming\}\}$.

## 2 An algorithm for automatic hierarchical tagging

Tagging is accomplished in three steps, i.e. candidate ranking, grouping, and refinement. In the **candidate ranking** step, terms are weighted for each cluster based on their descriptiveness (i.e. their value in describing the cluster) to identify the best tags. A good tag should not only describe the cluster but also distinguish a cluster from others. In a hierarchy, a tag must be able to distinguish a cluster from its sibling clusters as well as show the differences between the cluster and its parent cluster. These ideas are integrated in our own descriptive score $DS_w$ (see also [4] for related ideas). This score is compared to pure document frequency $df$ and modified information gain $IG_{mod}$ [3]. In specific, the descriptiveness $DS_w$ of a term $t$ in node $n$ is computed by

$$DS_w(t,n) = \log_2\left(\frac{rank_{df}(t,n_p)}{rank_{df}(t,n)}\right) \cdot \frac{1 - SI(t,n) + SI(t,n_p)}{2} \cdot \left(\frac{df_{t,n}}{|n|}\right)^w$$

$$SI(t,n) = \begin{cases} 1 & \text{if } ch(n) = \emptyset \\ \left(\sum_{n_c \in ch(n)} \frac{df_{t,n_c}}{df_{t,n}} \log_2 \frac{df_{t,n_c} \cdot |n|}{df_{t,n} \cdot |n_c|}\right) / \log_2 \frac{|n|}{min_{n_c \in ch(n)} |n_c|} & \text{else} \end{cases}$$

with $rank_{df}(t,n)$ being the rank of $t$ in $n$ if terms are ordered by their document frequency in $n$, $df_{t,n}$ the document frequency of $t$ in $n$, $n_p$ the parent node of $n$, and $ch(n)$ the set of child nodes of $n$. The score combines three factors. The first measures the boost of document frequency ranking in comparison to the parent. This assures that terms get higher scores that were not already good descriptors for the parent 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. $SI$ is based on the KL-Divergence between the distribution of document frequency and the distribution of node size, normalized to stay in the interval $[0;1]$. This means that $SI$ becomes zero, if $t$ is distributed in the child nodes with the same distribution as the documents, i.e. if $\frac{df_{t,n_c}}{df_{t,n}} = \frac{|n_c|}{|n|}$ for all child nodes.

On the other hand, $SI$ reaches the maximum of 1, if $t$ occurs only in the smallest child node. Therefore, the second factor favors terms that occur in several child clusters and penalizes terms that could be also descriptors in sibling nodes. The last factor considers the document frequency as a relatively high frequency is necessary however not sufficient for a good term. How strong the influence of the frequency should be on the final score is controlled by $w$. Our experiments showed that 0.33 is a good value for $w$ (at least for the considered dataset).

In the **grouping** step, the ranked term list is handled sequentially to create tag groups. The first term forms the first tag group. For every following term, it is decided whether it forms a new tag group or belongs to an existing one. A tag group is hereby represented as a coverage vector over the documents in the collection. A document is covered by a term, if the term occurs in it. The coverage of a tag group is a summation of the individual term coverages, whereby the impact of each term is weighted by its rank in the tag group (exponentially decreasing by $e^{-0.5 \cdot (rank(t)-1)}$). Similarity between a term and a tag group (or two tag groups) is computed by the Dice coefficient between the coverage vectors ($\text{sim}(x, y) = \frac{2 \cdot x \cdot y}{||x|| + ||y||}$). A term is merged to the tag group with highest similarity, if it is above a threshold. Once all terms have been used, tag groups are merged as long as similarity between two tag groups is still above the threshold. From the remaining tag groups, all are removed that are a specialization from another tag group (which is determined by $\text{incl}(x, y) = (\sum_i \min(x_i, y_i)) / \sum_i y_i$).

In the **refinement** step, more specific tag groups from deeper hierarchy levels are propagated up in the hierarchy, if the coverage of non-leaf cluster tags is not high enough. Tag groups from child nodes are added to the parent tag, if they sufficiently increase the cluster coverage. Tag groups with the highest increase in coverage are added first.

## 3 Evaluation and Conclusion

We evaluated our approach with the banksearch dataset [5]. We used three different hierarchies, the original one, a binary version of the original one, and a noisy one, in which groups of documents are moved to other clusters. While the first one requires the extraction of a single tag group per class, the other two include multiple tag groups. Our evaluation measures are borrowed and adapted from gold standard evaluation in ontology learning [2]. We present in Table 1 the f-score combining average precision and recall. For each class, precision and recall are computed between the learned tag groups $G_l$ and the reference tag groups $G_r$ by $precision(G_l, G_r) = |G_l|^{-1} \sum_{g_l \in G_l} \max_{g_r \in G_r} sim(g_l, g_r)$ and $recall(G_l, G_r) = |G_r|^{-1} \sum_{g_r \in G_r} \max_{g_l \in G_l} sim(g_l, g_r)$. We use three similarity measures, which are term-based (tb), rank-based (rb), and document-based (db). $sim_{tb}$ measures whether exactly the same terms were chosen in the 5 highest ranked terms, while $sim_{rb}$ takes into account the actual ranking in the tag group. $sim_{db}$ compares the covered documents of two tag groups while ignoring the actual terms. These measures allow to evaluate different granularities, i.e. whether the right documents were assigned to the same tag group and whether

**Table 1.** Results with three f-score measures on three datasets

| Setting | RM | Original | | | Noise | | | Binary | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | tb | rb | db | tb | rb | db | tb | rb | db |
| (1) | *df* | 0.5000 | 0.5377 | 0.9794 | 0.4905 | 0.4751 | 0.8923 | 0.3745 | 0.4593 | 0.9464 |
| | $IG_{mod}$ | 0.8214 | 0.6920 | 0.9467 | 0.7757 | 0.5992 | 0.8672 | 0.7249 | 0.7404 | 0.9204 |
| | $DS_{0.33}$ | 0.7857 | 0.8003 | 0.9530 | 0.7035 | 0.6458 | 0.8684 | 0.6832 | 0.7556 | 0.9226 |
| (2) | $IG_{mod}$ | 0.7775 | 0.6561 | 0.9316 | 0.7959 | 0.5953 | 0.8873 | 0.7233 | 0.7384 | 0.9099 |
| | $DS_{0.33}$ | 0.7932 | 0.7962 | 0.9340 | 0.7762 | 0.7240 | 0.9043 | 0.6658 | 0.7791 | 0.8912 |

correct tags could be extracted. $sim_{tb}$ and $sim_{rb}$ are computed as follows while $sim_{db}$ is the Dice coefficient (see Section 2):

$$sim_{tb}(g_l, g_r) = \frac{\sum_{t \in g_r} \left\{ \begin{array}{l} 1 \text{ if } t \in g_l \\ 0 \text{ else} \end{array} \right\}}{|g_r|} \quad sim_{rb}(g_l, g_r) = \frac{\sum_{t \in g_r} \left\{ \begin{array}{ll} \frac{1}{rank(t,g_l)} & \text{if } t \in g_l \\ 0 & \text{else} \end{array} \right\}}{1/1 + \cdots + 1/|g_r|}$$

Furthermore, we compared two settings, (1) the standard approach of using a single tag group and (2) multiple tag groups. Table 1 only shows results for the best parameter setup found for each measure and setting. Comparing the three ranking measures, it can be seen that all three of them group the right documents together (db measure). However, *df* fails to rank the good terms high. While $IG_{mod}$ is better in the first 5 terms (tb measure), our $DS_w$ usually ranks the important terms higher (rb measure). Furthermore, it can be seen that performance drops in setting (1), if the clusters naturally consist of more than one tag group. Our group method is capable of increasing the performance for these datasets, especially in combination with $DS_w$.

Concluding the paper, we want to point out that we propose in this paper to improve the effectiveness of tagging by integrating relations between tags in form of tag groups. We developed a method that is capable of extracting such tag groups automatically and evaluated it with a dataset. In future work, we want to modify the descriptive score to better reflect multiple tag groups.

## References


1. Bade, K., Hermkes, M., Nürnberger, A.: User oriented hierarchical information organization and retrieval. In: Machine Learning: ECML 2007. (2007) 518–526
2. Dellschaft, K., Staab, S.: On how to perform a gold standard based evaluation of ontology learning. In: Proc. of 5th Int. Semantic Web Conference. (2006) 228–241
3. Geraci, F., Pellegrini, M., Margini, M., Sebastiani, F.: Cluster generation and cluster labeling for web snippets. In: Proc. of the 13th Symposium on String Processing and Information Retrieval. (2006) 25–36
4. Glover, E., Pennock, D., Lawrence, S.: Inferring hierarchical descriptions. In: Proc. of 11th Int. Conference on Information and Knowledge Management. (2002) 507–514
5. Sinka, M., Corne, D.: A large benchmark dataset for web document clustering. In: Soft Computing Systems: Design, Management and Applications, Vol. 87 of Frontiers in Artificial Intelligence and Applications. (2002) 881–890