Improving the Navigability of Tagging Systems with Hierarchically Constructed Resource Lists and Tag Trails*

Christoph Trattner

Knowledge Management Institute

and Institute for Information Systems and Computer Media

Graz University of Technology, Austria

E-mail: ctrattner@iicm.edu

Abstract. *Recent research has shown that the navigability of tagging systems leaves much to be desired. In general, it was observed that tagging systems are not navigable if the resource lists of the tagging system are limited to a certain factor k. Hence, in this paper a novel resource list generation approach is introduced that addresses this issue. The proposed approach is based on a hierarchical network model. The paper shows through a number of experiments based on a tagging dataset from a large online encyclopedia system called Austria-Forum, that the new algorithm is able to create tag network structures that are navigable in a efficient manner. Contrary to previous work, the method featured in this paper is completely generic, i.e. the introduced resource list generation approach could be used to improve the navigability of any tagging system. This work is relevant for researchers interested in navigability of emergent hypertext structures and for engineers seeking to improve the navigability of tagging systems.*

Keywords. *Tagging systems, tag navigation, tags, tag clouds, search, navigation*

1. Introduction

With the emergence of modern Web 2.0 hypertext systems such as Flickr, Delicious, CiteULike or LastFM, tagging systems have emerged as an interesting alternative to traditional forms of hypertext navigation and browsing. Tagging systems allow the user to use a free-form vocabulary to annotate resources with the so-called tags [13, 23]. This is done either for semantic reasons (for example, to enrich information items with additional meta data), conversational reasons (for example, for social signaling) [3] or for organizational reasons (for example, to categorize information) [21]. Regardless of why people tag [26, 29, 28], tags are typically visualized as the so-called tag clouds [3]. Basically, a tag cloud is a selection of tags related to a particular resource. Upon clicking on a tag in the tag cloud, a list of resources related to the tag is presented to the user. Thus, in addition to traditional browsing (through a hierarchal taxonomy) and searching (by entering search terms), tags, respectively tag clouds, provide users with a third orthogonal form of navigation within a collection of resources.

In previous work [15, 16, 7], it was observed that the navigability of tagging systems leaves much to be desired. In particular, in [15, 16] we found that the most common resource list generation approach

*This article is partly based on the work "Enhancing the Navigability of Social Tagging Systems with Tag Taxonomies" published in the Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies, ACM, 2011.

used these days in tagging systems generates network structures which are *per se* unnavigable [15, 16]. The issue is this: Limiting the resource list to a certain factor k, due to interface space restrictions, fragments the bipartite tag network of a tagging system into large isolated network clusters. This renders the network unnavigable from a network-theoretical point of view. However, in [15, 16] we suggested an approach to overcome this issue by applying a simple greedy resource generation strategy. The "trick" is to select, for every click on a particular tag in the tag cloud, the k related resources at random. In common tag cloud algorithms, for every tag click the same result list is generated. Since different resources are selected, this leads to the effect that the tag network becomes connected (even for small values of k) and in theory navigable again. However, as we have shown in [15], this simple strategy does not lead to tag networks which are "good" or even "efficiently" navigable. Therefore, we have investigated in our recent work more sophisticated strategies to generate a k-limited resource list for a particular tag in the tagging system. In [32] we have shown that it is possible, at least in theory, if we apply a hierarchical network model [20] to select the k resources for the resource list. The idea is to place the resources in the collection within a hierarchical taxonomy and to use this taxonomy to generate a probability function to select the k resources in the resource list [32].

In [14] we introduced a hierarchical decentralized search approach. In [32] we used the searcher to simulate a user navigating a tagging system. In short, simulations were able to demonstrate that the hierarchical resource list generation approach generates tag networks which are significantly more navigable than tag networks generated by the most popular resource list generation approach – the reverse chronological sorting resource list generation algorithm [32]. However, these results can only be approximate, as long as there is no research into how people really navigate within a tagging system. Hence, a formal experiment was conducted to validate these results empirically.

In [31] we presented results of a formal experiment. The experiment confirmed the results from the simulations and showed that the hierarchical resource list generation approach creates network structures which are significantly more navigable than networks relying on the most commonly used resource list generation algorithm. Although the experiment was successful, it also confirmed one important limitation of the approach. In particular, the experiment proved the theoretical assumption that the algorithm performs poorly if the underlaying resource taxonomy has high branching factors [20]. For the experiment in [31], a more or less high branched resource taxonomy from a tagging system called Austria-Forum [33] was used. Hence, in this paper we present an enhanced version of the algorithm. Contrary to the approach in [32], the method introduced in this work is able to generate a fixed branched resource taxonomy and corresponding "resource trails" autonomously, i.e. it is independent of any given resource taxonomy. Moreover, the presented approach is generic and can therefore be used to improve the navigability of *any* given tagging system.

The paper is structured as follows: In Section 2 the hierarchical resource list generation algorithm is presented. In Section 4 the approach is evaluated and in Section 5 the approach is discussed. Finally in Section 6, the conclusions are presented.

2. Hierarchical Resource List Construction

The hierarchical resource list generation algorithm is a novel approach for resource list generation in a tagging system [32]. To put it simply, the the approach places the resources into a hierarchical taxonomy and reuses the hierarchy to generate

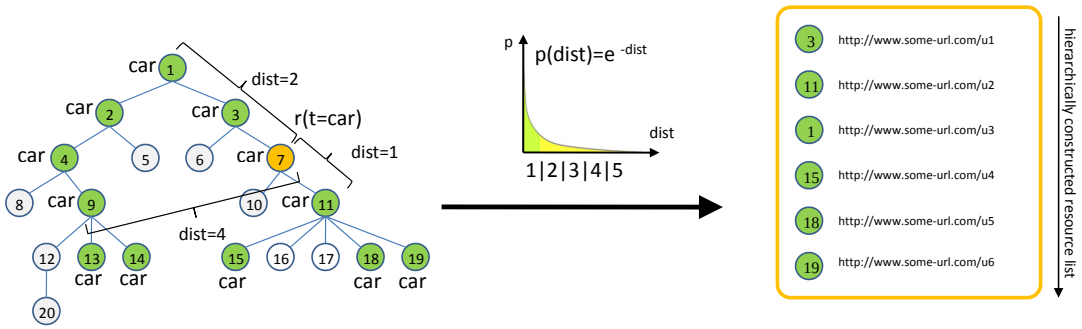


Figure 1: Sample resource taxonomy and corresponding hierarchically constructed resource list for tag "car". The green nodes are the resources in the taxonomy that have the tag "car" applied. In the middle of the Figure the resulting probability function is presented and on the right side the generated resource list is shown.

a probability function to select the resources in the tagging system. If the taxonomy provides a constant branching factor b, the emerging tag network is efficiently navigable. The idea for this algorithm was originally derived based upon work by J. Kleinberg [20] who has investigated structural clues of small world networks. Kleinberg showed [20] that if the nodes of a network can be organized into a hierarchy with a constant branching factor b, then such a hierarchy provides a probability distribution for connecting the nodes in the network to generate a network that is then efficiently navigable.

In detail, the algorithm works as follows: For each click on a tag $t(r)$, where r is a resource in the tagging system, the algorithm returns a k-limited resource where the resources $r(t(r))_i$ in the list are selected randomly according to a probability function p that is calculated from a given resource taxonomy T. p is calculated as

$$e^{-dist(r(t(r)),r(t(r))_i)} \quad (1)$$

The distance $dist(r(t),r(t)_i)$ is calculated as

$$h(r(t))+h(r(t)_i)-2h(r(t),r(t)_i) \quad (2)$$

where $h(r(t))$, $h(r(t)_i)$ are the heights of $r(t)$ and $r(t)_i$ in a given resource taxonomy T and where $h(r(t),r(t)_i)$ is the height of the least common ancestor of $r(t)$ and $r(t)_i$ in the resource taxonomy T [32] (see Algorithm 1).

In Figure 1 an illustrative example of a resource taxonomy and the corresponding hierarchically constructed resource list for the tag "car" is given. Note that the orange node in Figure 1 represents the resource that is currently viewed by the user. The green nodes are the resources in the taxonomy that have the tag "car" applied. The resulting probability function is presented in the middle of Figure 1 and the generated resource list is shown on the right side.

2.1. Resource Taxonomy Generation Algorithm

To overcome the issue of a given resource taxonomy Algorithm 1 has been extended to a generate a fixed branched resource taxonomy autonomously. In related work, [17] Heymann et al. (see also [5]) describe an algorithm to generate a tag taxonomy from tagging data. The input for the algorithm is the so-called tag similarity graph, i.e. an unweighted graph where each tag is a node in the graph, and two nodes are linked to each other if their similarity is above a predefined similarity threshold. In the simplest case, the threshold is defined by tag overlap, i.e. tags need to share at least one resource to be linked with each other.

Algorithm 1 Hierarchical Resource List Generation Algorithm

INPUT: tag t, resource r, max. resource list size k, resource taxonomy T
OUTPUT: resource list RS
$R(t) \leftarrow$ get all resources $r(t) \backslash r$
$D \leftarrow$ new $HashMap$[new $Array$[]]
for each $r(t)_i \in R(t)$ **do**
  $dist \leftarrow h(r) + h(r(t)_i) - 2h(r, r(t)_i) - 1$
  /* $h(r), h(r(t)_i)$ are the heights of the resource nodes $r, r(t)_i$ in T, $h(r, r(t)_i)$ is the height of the least common ancestor of $r, r(t)_i$ in T */
  $D[dist].add(r(t)_i)$
end for
$j \leftarrow 0$
$RS \leftarrow$ new $Array$[]
while $sizeof(RS) < k$ && $sizeof(RS) < sizeof(D)$ **do**
  $RS[j] \leftarrow D[p_{exp}, p_{uni}]$
  /* p_{exp} is a random number with exponential distribution in the interval $0 \leq x < sizeof(D)$, p_{uni} is a random number with uniform distribution in the interval $0 \leq x < sizeof(D[p_{exp}])$ */
end while
sort RS by $dist$ in descending order
return RS

The second prerequisite for the algorithm is the ranking of nodes in a descending order according to how central the tags are in the tag similarity graph. In particular, this ranking produces a generality order where the most general tags from a dataset are highly ranked. The algorithm starts with the most general tag as the root node of the tree. The algorithm then proceeds by iterating through the generality list. For each tag in the the tree it adds the current processed tag as a child to its most similar tag. [14]

In this work, a similar algorithmic approach is developed. Contrary to the algorithm of Heymann et al. the algorithm is able to generate a fixed branched taxonomy without defining a predefined similarity threshold. In Algorithm 2 the actual algorithm is presented. In words, the algorithm works as follows (see also [35]):

The algorithm takes a tag dataset and the desired taxonomy branching factor as input parameters. Since the algorithm should generate a resource taxonomy with the most general resource of the tagging system as root node and related and less general resources as children, the algorithm calculates in the first step degree centrality for all resource of the supplied tagging dataset and stores the centrality-resource pairs into a map C. Degree centrality was chosen since, on the one hand, it is computed fast, and on the other hand, it was observed in previous research [6] that degree centrality in tagging systems is highly correlated to sophisticated centrality measures such as closeness or betweenness centrality. In the next step, the algorithm sorts the resources in C according to their centrality values in descending order.

Subsequently, the algorithm takes the first element of C (i.e. the most general resource) and sets that resource as the root node of the resource taxonomy. Thereafter, the algorithm starts iterating trough the elements (resources) already present in resource taxonomy. For each resource in the resource taxonomy the algorithm calculates then the most similar resources (see $getMoreLikeThis$ in Algorithm 2). Our algorithm calculates cosine similarity between all co-occurring resources taking also the $tf \cdot idf$ values of the tag concepts into account. Additionally, the function returns only resources that are not already part of the constructed resource taxonomy. The results of this function are stored into a map SIM, with resources as key values and with the provided similarity values as corresponding map values. To account for resource generality we multiply resource similarity values with their corresponding centrality values. The final scores are normalized to fall into the range of $[0...1]$. After that, the resources in SIM are sorted by the scores in descending order. This procedure ensures that the resources in SIM are not only similar to the

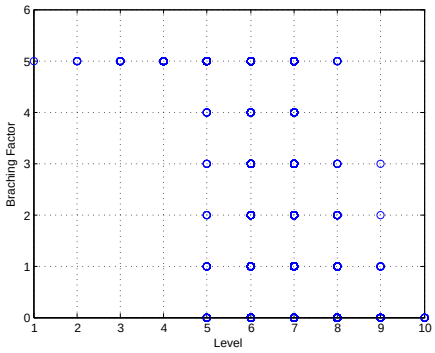


Figure 2: Branching factor distribution for a resource taxonomy with maximum branching $b = 5$ generated from a tagging dataset in Austria-Forum [33].

currently processed resource but also sorted by their generality values. Thereafter the algorithm appends a maximum of b resources to the currently processed resource. The algorithm stops, if no more similar resources can be found.

Note, due the fixed branching factor b the algorithm does not guarantee that all resources of the tagging dataset are contained in the resulting resource taxonomy. However, as it will be shown in Table 1 the probability that one or even more resources are missing is relatively small due to the high number of existing links between the resources of the resource-to-resource network of a given tag dataset. On the other hand, in a tag taxonomy the probability that one concept is missing is significantly higher. The reason for this behavior is the fact that the tag-to-tag network of a tagging system is typically substantially less connected.

Figure 2 shows the branching factor distribution for a tag-resource taxonomy with branching $b = 5$ generated from the Austria-Forum tag dataset. For branching factor $b = 5$ the algorithm does not generate a complete $b-tree$ (from levels 1 to 4 the resulting tree is complete, for levels > 4 the tree is not complete). The reason for this behavior is the fact that in tag networks there are resources which are just connected

Algorithm 2 Resource Taxonomy Generation Algorithm

INPUT: Tag Dataset D, Branching Factor b
OUTPUT: Resource Taxonomy T
$C \leftarrow$ new $HashMap[]$
$T \leftarrow$ new $Tree[]$
for each $r_i \in F$ **do**
  $C[r_i] \leftarrow$ calculate degree centrality
end for
$sortByValues(C)$
/*sort C by values in descending order*/
$T[0] \leftarrow C[0]$
$SIM \leftarrow$ new $HashMap$
for i = 0; i $< sizeof(T)$; i++ **do**
  /*get all similar resources of $T[i]$ and store the resources as key values and the similarity values into SIM*/
  $SIM \leftarrow getMoreLikeThis(T[i])$
  for each $r_i \in SIM$ **do**
    $T[r_i] \leftarrow T[r_i] \cdot C[r_i]$
  end for
  /*sort the resources in SIM by values in descending order*/
  $sortByValues(SIM)$
  for j = 0; j $< sizeof(SIM)$ and $j < b$; j++ **do**
    $T[i].append(SIM[i])$
  end for
end for
return T

to a few resources, i.e if the branching factor b is beneath this threshold the resulting taxonomy becomes incomplete.

2.2. Resource Taxonomy Labeling Algorithm

In order to give the user information about how the resources are structured in the tagging system, tag/title trails are attached as additional information for each resource of the tagging system (see Figure 2). In an experiment [31] conducted recently, resource trails were attached to the resources in the result lists of the tagging system. In other words, in [31] we observed that all 24 participants of the experiment were using

Algorithm 3 Resource Taxonomy Labeling Algorithm

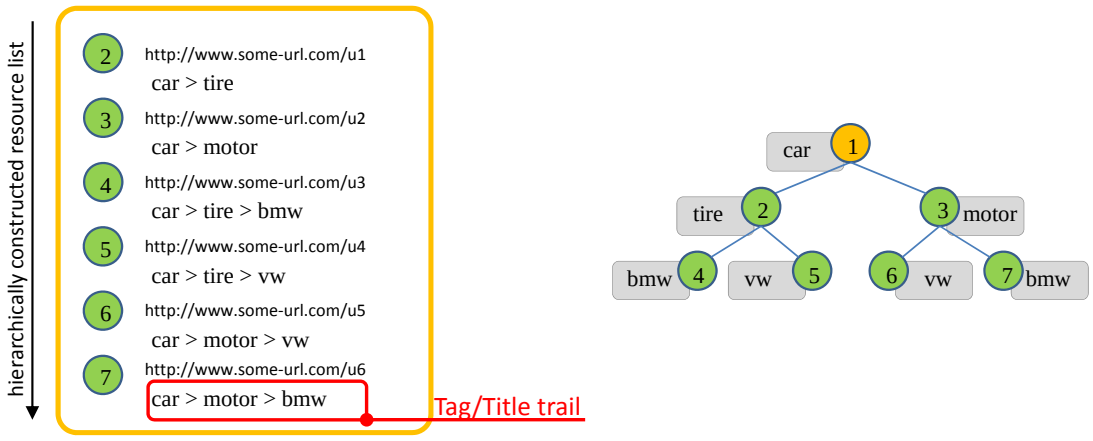
```
INPUT: Resource Taxonomy T, Tag Dataset D
OUTPUT: Tag-resource Taxonomy
COTAGS ← new HashMap[newArray[]]
for i = 0; i < sizeof(T); i++ do
    Ts ← getTags(T[i], D)
    for j = 0; j < sizeof(Ts); j++ do
        cotags ← getCoocTags(Ts[j], D)
        sort(cotags)
        remove all tags from cotags that are not
        contained in T[i]
        COTAGS[T[i]].add(cotags)
    end for
end for
trails ← new HashSet[]
for each r_i ∈ T do
    /*T is traversed in left-order*/
    pl ← getParentLabels(r_i)
    for each l_j ∈ COTAGS[r_i] do
        if !pl.contains(l_j) then
            if !(trails.contains(pl.toString() +
            l_j)) then
                T[r_i].applyLabel(pl)
                trails.add(pl.toString() + l_j)
            end if
        end if
        if T[r_i] has no label then
            T[r_i].applyLabel(getTitle(r_i))
        end if
    end for
end for
return T
```

resource trail information for orientation rather than tag information to navigate the tagging system.

However, since resource trails extracted from a resource taxonomy would be impossible for humans to read, a labeling algorithm is introduced to make the resource taxonomy readable by humans. The basic idea of the algorithm is it to use tag and title information to label a particular resource in the resource taxonomy and to use the resulting taxonomy to generate tag/title trails which are attached to the resources in the resource list (see Figure 3), i.e. we attempt to contextualize the resources.

In general it is a labeling algorithm taking a given resource taxonomy and a tagging dataset as input parameters. Tag information is used as label data. The algorithm tries to apply labels to the given resource taxonomy in such a way, that they are uniquely distinguishable and the most descriptive for the given resource. The candidate tags are thereby ranked by the method of tag co-occurrence. However, since it can happen that resources in the resource taxonomy have the same tags in their parent tag trail, due to the lack of available tags in the tagging system, additional meta-data is taken into account. We use title information of the resources as an additional way for differentiation. In words the algorithm works as follows (see also [35]):

In the first step the algorithm calculates, for each resource in the resource taxonomy a list of co-occurring tags of all resource tags and stores this list sorted in descending order into a map. After that, the algorithm traverses the resource taxonomy in left-order. In this loop the actual labeling procedure is performed. In detail, the labeling process looks as follows: For each resource in the resource taxonomy the corresponding co-occurrence vector is consulted and the first label, i.e. the most frequent tag, is tried to be applied to the currently processed resource. If the currently used candidate tag is already part of the tag trail of the currently processed resource (see variable *trails* in Algorithm 2) the next element, i.e. the next frequent tag label is chosen as candidate tag. If no uniquely distinguishable tag trail can be constructed, i.e. the candidate tag label from the co-occurrence vector is already present in the tag trail of the resource additional meta data is considered. We use title information of the currently processed resource for this purpose. Note, since tag and title information can be identical the proposed method is not completely free of collisions. However, to fix this issue one can



(a) Sample of a hierarchically constructed resource list with attached tag/title trails

(b) Sample resource taxonomy with attached tag/title labels

Figure 3: Sample of a hierarchically constructed resource list with attached tag/title trails (on the left) and corresponding resource taxonomy with applied tag/title labels (on the right). Note, compared to a pure tag taxonomy (see [14] for instance), in a labeled resource taxonomy terms can occur more than once. The orange node in the resource taxonomy (again) denotes the currently viewed resource by the user.

Name	b	n	TTL_{max}	TTL_{mean}
Res2	2	19,430	17	12.45
Res5	5	19,430	10	5.93
Res10	10	19,430	8	4.44

b = branching factor, n = number of nodes in the resource taxonomy, TTL_{max} = maximum Tag/Title Trail Length, TTL_{mean} = mean Tag/Title Trail Length

Table 1: Tag/title Lengths for different branching factors. As expected the the resource taxonomy with the smallest branching factor $b = 2$ generates the longest trails RTL_{max} = 17 and the resource taxonomy with highest branching factor $b = 10$ the shortest RTL_{max} = 8.

include additional meta data information or other methods to generate a unique label such as appending an iterative number for each label that occurs more than once. The algorithm stops if all resources of the given resource taxonomy are labeled.

3. Dataset

The described experiments in this paper are based on the tag dataset from a system called the Austria-Forum [4, 33]. Basically, the Austria-Forum is a large online encyclopedia similar to Wikipedia providing the user with approximately 180,000 resources related to Austria. In contrast to Wikipedia, Austria-Forum structures articles into a taxonomy and provides an integrated tagging system [33, 34, 30], which allows users to assign tags to resources and to navigate to related resources via tag clouds. As of October 16, 2010 the Austria-Forum tag dataset contains 97,908 tag assignments, 13,314 tags, and 19,430 resources.

4. Experiments

In order to evaluate the proposed hierarchical resource list generation approach five different experiments were conducted.

In this section, the experiments are described and results are presented.

4.1. Measuring Average and Maximum Tag/Title Trail Lengths

In the first experiment, we investigated the average and the maximum resource taxonomy depths for different branching factors b. Since the resulting resource taxonomies generated by Algorithm 2 are not complete neither the average nor the maximum depth could not be estimated by formulas. Hence, these values were conducted empirically through an experiment. For the experiment, three different resource taxonomies named $Res2$, $Res5$ and $Res10$ with three different branching factors $b = 2,5$ and 10 were generated. In order to compare the resulting taxonomies against a golden standard taxonomy the DMOZ Open Directory Project [10] (ODP) taxonomy was consulted. This experiment was conducted to determine whether or not the resulting tag/title trails will be in length usable for humans (see also [18, 36, 24]).

As shown in Table 1 and as expected, the resource taxonomy with the smallest branching factor $b = 2$ generates the longest tag/title trails TTL_{max} = 17 and the resource taxonomy with highest branching factor $b = 10$ generates the shortest TTL_{max} = 8. For $b = 5$ the maximum tag/title trail length is TTL_{max} = 10. On average when branching factor $b = 2$, the trail length is $TTL_{mean} = 12.45$. For $b = 5$, the mean trail length is $TTL_{mean} = 5.93$ and for $b = 10$ the mean trail length is $TTL_{mean} = 4.44$. The ODP Taxonomy has a mean depth of 6.86 [2]. The maximum depth is 13 [2]. Hence, compared to the ODP taxonomy, the resource taxonomy with branching factor $b = 5$ maps more closely to a human crafted taxonomy. However, the resource taxonomy with branching factor $b = 10$ is most usable [18, 36, 24] since it generates the shortest trails of all taxonomies. The resource taxonomy with branching factor $b = 2$ is, in our opinion, not usable.

4.2. Measuring Labeling Collision Rate

In the second experiment, we measured the number of "collisions" which occurs when labeling a given resource taxonomy with different branching factors b (with the labeling Algorithm introduced in Section 2). As explained, the labeling algorithm is not 100% collision free. Hence, the experiment should reveal how many collisions occure in general if labeling a resource taxonomy with a given branching b. For this experiment the three resource taxonomies from the prior experiment were used. To show the potential of using tag and title information together to label a given resource taxonomy we also conducted an experiment for which we only used tag information to label the three given resource taxonomies. As shown in Table 2, for branching factor $b = 2$ the collision rate is $CR_{tt} = 0.1\%$, for $b = 5$ the collsion rate is $CR_{tt} = 0.2\%$ and for $b = 10$ the collision rate is $CR_{tt} = 0.2\%$. The collsion rate is significantly increased if taking only tag information into account. For $b = 2$ the collsion rate is $CR_t = 8.5\%$, for $b = 5$ the collision rate is $CR_t = 12.9\%$ and for $b = 10$ the collision rate is $CR_t = 15.9\%$. All in all, one can observe that the higher the branching factor, the higher the collision rate.

4.3. Measuring Semantic Structure of the Labeled Resource Taxonomy

The third experiment investigated the quality of the semantic structure of the three labeled resource taxonomies from the previous experiment. For that purpose, two different semantic measures were consulted – the Taxonomic F-Measure [8] and the Taxonomic Overlap [22]. Both measures assess the quality of a given taxonomy against a gold standard over common concepts. The

Name	b	n	CR_{tt} (%)	CR_t (%)
Res2	2	19,430	0.1%	8.5%
Res5	5	19,430	0.2%	12.9%
Res10	10	19,430	0.2%	15.9%

b = branching factor, n = number of nodes in the resource taxonomy, CR_{tt} = Collision Rate with tag/title information, CR_t = Collision Rate with tag information only

Table 2: Collision Rates for different resource taxonomies with different branching factor b. As shown, the higher the branching factor the higher the collision rate. The collision rate is drastically increased if taking only tag information for labeling into account.

Taxonomic F-measure (=TF) is defined as the harmonic mean of the Taxonomic Recall and Precision (see [8] for more details). The Taxonomic Overlap (=TO) is defined as $TO = TF/(2-TF)$ [8]. As a gold standard for this experiment the Germanet [12] ontology was used (the Austria-Forum tag dataset contains only German tags). Since the idea of a labeled resource taxonomy with attached tag/title labels could be best compared with a tag taxonomy we compared the approach with four different famous tag taxonomies generated by the following popular tag taxonomy induction algorithms: Hierarchical K-Means [9], Affinity Propagation [11, 27], Heymann [17] and Deg/Cooc [14, 5]. In the experiment, TF and TO values were measured for all seven taxonomies generated and benchmarked against each other. The goal of the experiment was to determine if the labeling algorithm generates semantic structures which are more useful than the structures generated by the popular tag taxonomy induction algorithms such as Hierarchical K-Means, Affinity Propagation, Heymann or Deg/Cooc [14]. Note, since a label in a resource taxonomy (compared to tag taxonomy) can occur more than once, the average semantic cotopy values (see [8]) for calculating TF and TO values for the resource taxonomies $Res2$, $Res5$ and $Res10$ were calculated, i.e. for each label we measured the semantic cotopy values and averaged them when there was more than one label present in the taxonomy (see [8] for more details).

In Figure 4, results of the semantic evaluation of the experiment are presented. As shown, the resource taxonomy with a branching factor $b = 2$ generates the lowest $TF = 19\%$ and $TO = 11\%$ values of all measured taxonomies. However, the semantic values of the resource taxonomies increase with the higher branching factor. For $b = 5$, $TF = 29\%$ and $TO = 17\%$. For $b = 10$, $TF = 34\%$ and $TO = 20\%$. As shown in Figure 4, the resource taxonomy with braching factor $b = 10$ generates higher TF and TO values than a tag taxonomy based on the tag taxonomy induction approach such as Affinity Propagation ($TF = 30\%$, $TO = 18\%$) and Heymann ($TF = 25\%$, $TO = 14\%$). However it performs worse than tag taxonomy based on an induction algorithm such as Deg/Cooc ($TF = 39\%$, $TO = 24\%$) and Hierarchical K-Means ($TF = 35\%$, $TO = 21\%$). The resource taxonomy with a branching factor $b = 5$ performs better than the Heymann tag taxonomy, but worse then all other. All in all, one could say that resource taxonomies with branching factors between $b = 5-10$ perform on average as good as "pur" tag taxonomies based on a Affintity Propagation tag taxonomy induction algorithm. Or, in other words, the resulting tag/title trails of a resource taxonomy with braching factors between $b = 5-10$ are in average as good as the tag/title trails of the tag taxonomy generated by the Affinity Propagation tag taxonomy induction algorithm.

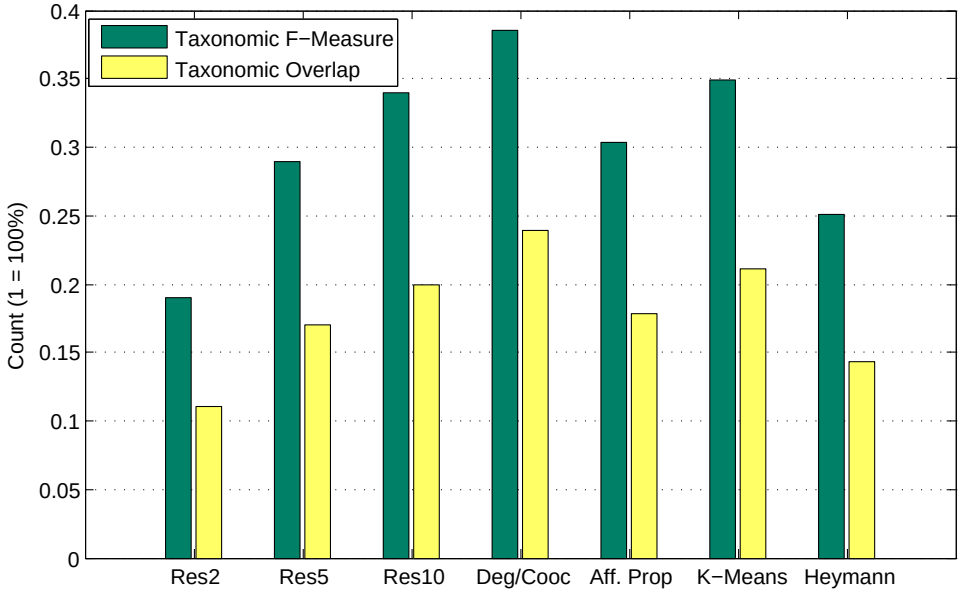


Figure 4: Results of the semantic evaluation of the generated resource taxonomy with applied tag/title labels using the Austria-Forum tag dataset. As shown in the resource taxonomy, *Res*5 generates the lowest TF and TO values of all measured taxonomies. However, the TF and TO values increase with the higher branching factor b. For $b = 10$ the semantic structure of the labeled resource taxonomy *Res*10 is nearly as good as the $K-Means$ tag taxonomy.

4.4. Measuring Navigability

In order to evaluate the navigability of the tag networks resulting from the proposed hierarchical resource list generation approach, three different types of tag networks were generated. They all varied in how the the resource lists were calculated. In the following list, we describe the tag networks as they were generated and used for our further experiments:

Network *CHRON***:** This is the type of tag network as typically found in tagging systems such as Delicious, CiteULike or Flickr. The tag network relies on a resource list generation algorithm that returns for each click on a particular tag t in the tagging dataset a k-limited resource list that is sorted in reverse chronological order. Contrary to the hierarchical resource list generation approach, the resource lists are statically calculated, i.e. no mater what resource in the tagging system is currently viewed by the user it always the same resource list for a particular tag t calculated [32, 15, 16].

Network *RAND***:** This type of tag network relies on the resource list generation algorithm that returns for a particular tag t a different and randomly sorted k-limited resource list. Contrary to the chronological approach, the resource lists are *not* statically calculated, i.e. for each click on a tag $t(r)$, a different resource list is generated.

Network *HIERx***:** This type of tag network relies on the hierarchical resource list generation approach as introduced in Section 2. For our experiments in this paper three separate tag networks of this type were generated. They all vary in the way in which resource taxonomies were used to generate the resulting tag networks. As input resource taxonomies the three resource taxonomies *Res*2, *Res*5 and *Res*10 from Section 4.1. were chosen. The resulting networks are called Network *HIER*2, *HIER*5 and *HIER*10.

In order to determine whether the generated tag networks are navigable, network

Name	k	Nodes	Links	ED	LSCC
$CHRON_{10}$	10	19,430	660,457	4.22	**0.77**
$RAND_{10}$	10	19,430	678,623	4.00	0.99
$HIER2_{10}$	10	19,430	619,641	4.29	0.99
$HIER5_{10}$	10	19,430	622,554	3.99	0.99
$HIER10_{10}$	10	19,430	625,512	4.30	0.99
$HIER5_{10}$	10	19,430	622,554	3.99	0.99
$CHRON_{50}$	50	19,430	2,156,133	3.95	**0.90**
$RAND_{50}$	50	19,430	2,191,483	3.87	0.99
$HIER2_{50}$	50	19,430	2,086,978	4.05	0.99
$HIER5_{50}$	50	19,430	2,093,926	3.90	0.99
$HIER10_{50}$	50	19,430	2,097,897	3.86	0.99

LSCC = Largest Strongly Connected Component, ED = Effective Diameter

Table 3: Tag network statistics. According to Kleinberg [19, 20] networks *RAND* and *HIERx* are navigable networks. Network *CHRON* is unnavigable [19, 20], i.e. not (nearly) all nodes of the network are contained in the giant component of the network.

properties such as the size of the largest strongly-connected component (*LSCC*) and the effective diameter (*ED*) were calculated. From a network-theoretic perspective, Kleinberg [20] showed that a navigable network can be formally defined as a network with a low diameter [25] bounded by $log(n)$, where n is the number of nodes in the network, and an existing giant component, i.e. a strongly connected component containing almost all of the nodes. For that experiment the maximum resource list size k was also varied to $k = 10$ and $k = 50$. This was done to observe whether or not different values of k influence the navigability of the different tag networks. The overall goal of this experiment was to determine whether or not the tagging system relaying on a hierarchical resource list generation algorithm produces tag networks which are more navigable than tag networks generated by a chronological or random resource list generation approach.

In Table 3, the network statistics of all four tag networks are shown. According to Kleinberg [19, 20] networks *RAND* and *HIER* are navigable networks. In network $CHRON_{10}$, 23% and in network $CHRON_{50}$, 10% of all resources are not within the giant component of the tag network. Hence network *CHRON* is unnavigable [19, 20]. These results go along with the observations made in [15, 16].

4.5. Measuring Efficiency

In the concluding experiment, we measured the efficient navigability of the tag network with a hierarchical decentralized searcher [1] as introduced in [14]. As defined by Kleinberg, an efficiently navigable network is a network for which a decentralized searcher exists that is able to navigate to all nodes of the network in $log(n)$ or at least in sub-linear to n time, where n are the number of nodes in the network. In [14] we have introduced a searcher that is able to search a tagging system in $log(n)$. However, contrary to the searcher in [14], the searcher in this work uses as background knowledge the resource taxonomy which was utilized to generate the tag network (see Algorithm 4). Additionally, the searcher in this work is able to walk along a directed tag network. In [14], it was limited to a bipartite tag network. In Algorithm 4, the pseudo code of the implemented searcher is presented. Note that the searcher is using as input parameters a directed resource-resource tag network, a

resource taxonomy, a start and target node and a maximum number of hops parameter that defines how many resources the searcher should at maximum visit before giving up. For an input taxonomy the searcher is taking the corresponding resource taxonomy, i.e. for Network $HIER2$ resource taxonomy $RES2$ is taken, for Network $HIER5$ resource taxonomy $RES5$ is taken, for Network $HIER10$ resource taxonomy $RES10$ is taken and for network $RAND$ a random resource taxonomy was generated.

In order to acquire statistically significant results, 100,000 random searches (with a maximum of 10 hops) for each of the networks were performed. The start and target nodes were selected uniform at random. For the experiment only resource pairs were considered for which a path was present in the network. If the target node could not be found in at least 10 hops or the searcher was caught in a cycle (we did not recover the searcher in that case) this was counted as an error. It is important to note that both searcher were given the exact same start and target nodes for all four networks.

In Figure 4 we present the success rate plots of the hierarchical decentralized searcher for tag networks $RAND$ and $HIERx$ and different values of k. As shown, the hierarchically constructed tag networks outperform the random networks significantly. As also shown in Figure 4, tag network $HIER2$ is most navigable. Regardless of which branching factor, the searcher is able to find nearly 100% of all nodes in this network. According to Kleinberg's definition [19, 20] tag networks $HIER2$ and $HIER5$ are also efficiently navigable network.

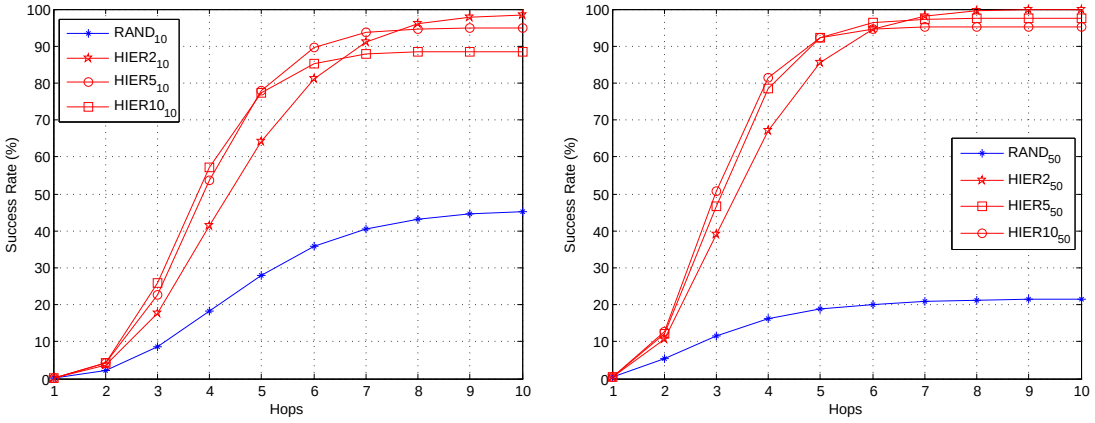
Algorithm 4 Hierarchical Decentralized Searcher [14]

INPUT: resource resource graph R, resource taxonomy T, start node v, target node w, max hops $hops_{max}$
$hops \leftarrow 0$
while $v! = w$ **do**
  if $++hops >= hops_{max}$ **then**
    break
  end if
  $R(v) \leftarrow$ get all resources from $v \in R$
  $dist_{min} \leftarrow \infty$
  for each $r_i \in R(v)$ **do**
    $dist \leftarrow h(r_i,T) + h(v,T) - 2h(r_i,v,T) - 1$
    if $dist < dist_{min}$ **then**
      $dist_{min} \leftarrow dist$
      $v \leftarrow r_i$
    end if
  end for
end while

5. Discussion

However, even if the experiments showed that the proposed resource list generation approach is able to generate tag networks that are efficiently navigable, the experiments also revealed that the proposed approach has also limitations. In particular, the experiments revealed that there are limitations regarding the generated resource taxonomy. For instance, in the first experiment it was observed that it is not useful to generate a resource taxonomy with small branching factors since the resulting tag/title trails could grow too long to be usable. In the second experiment it was shown, that a resource taxonomy with higher branching factors led to tag/title trails that have a higher probability of not being unique distinguishable anymore. In the third experiment, it was shown that a resource taxonomy with low branching factors would lead to tag/title trails which are semantically less useful that the trails of a labeled resource taxonomy with higher branching factors. In experiment five, it was shown that a smaller branched resource taxonomy will lead to tag networks that are more navigable.

However, the experiments also showed that, there is distinctive branching factor b for which a *good* trade-off between usability, semantics and navigability is given. Our experiments showed that for the Austria-



(a) $k = 10$ (b) $k = 50$

Figure 5: Success rates of the hierarchical decentralized for tag networks RAND and HIERx and different values of k. As shown,the hierarchically constructed tag networks outperform the random networks most. As also shown in Figure 4, tag network *HIER*2 is most navigable. Regardless of which branching factor, the searcher is able to find nearly 100% of all nodes in this network. According to Kleinberg's definition [19, 20] tag networks HIER2 and HIER5 are also efficiently navigable network.

Forum tag dataset the optimal branching factor would be approximately $b = 5$ in order to produce resource lists containing usable tag trails and to generate efficiently navigable tag networks.

6. Conclusions

In this paper, a novel approach for resource list generation for tagging systems was presented. In particular, the paper presented a resource list generation approach that is based on a hierarchical network model. A number of experiments showed that the approach is able to generate tag network structures which are efficiently navigable. Contrary to previous work, the proposed approach is completely generic, i.e. the introduced hierarchical resource list generation approach could be used to improve the navigability of *any* tagging system.

7. Acknowledgments

We would like to thank Prof. Denis Helic, Mr. Christian Körner, Mr. Markus Muhr, Prof. Keith Andrews and Prof. Hermann Maurer for many fruitful discussions during this work. This work is funded by - *BMVIT - the Federal Ministry for Transport, Innovation and Technology*, program line *Forschung, Innovation und Technologie für Informationstechnologie*, project *NAVTAG – Improving the navigability of tagging systems*.

References

[1] L. Adamic and E. Adar. How to search a social network. *Social Networks*, 27(3):187 – 203, 2005.

[2] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, and K. Tolle. The ics-forth rdfsuite: Managing voluminous rdf description bases. In *SemWeb*, 2001.

[3] M. Ames and M. Naaman. Why we tag: Motivations for annotation in mobile and online media. In *Proc. SIGCHI Conference on Human Factors in Computing Systems (CHI 2007)*, CHI'07, pages 971–980, USA, NY, 2007. ACM.

[4] Austria-Forum. Das ï¿œsterreichische Wissensnetz. http://wwww.austria-lexikon.at, 2011. [Online; accessed 2011-03-01].

[5] D. Benz, A. Hotho, and G. Stumme. Semantics made by you and me: Self-emerging ontologies can capture the diversity of shared knowledge. In *Proc. of the 2nd Web Science Conference (WebSci10)*, Raleigh, NC, USA, 2010. Web Science Trust.

[6] D. Benz, C. Körner, A. Hotho, G. Stumme, and M. Strohmaier. One tag to bind them all : Measuring term abstractness in social metadata. In *Proceedings of the 8th Extended Semantic Web Conference (ESWC 2011)*, Heraklion, Crete, May 2011.

[7] E. H. Chi and T. Mytkowicz. Understanding the efficiency of social tagging systems using information theory. In *HT '08: Proceedings of the nineteenth ACM conference on Hypertext and hypermedia*, pages 81–88, New York, NY, USA, 2008. ACM.

[8] K. Dellschaft and S. Staab. On how to perform a gold standard based evaluation of ontology learning. In *Proceedings of ISWC-2006 International Semantic Web Conference*, Athens, GA, USA, November 2006. Springer, LNCS.

[9] I. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. In R. Grossman, C. Kamath, and R. Naburu, editors, *Data Mining for Scientific and Engineering Applications*. Kluwer Academic Publishers, Heidelberg, 2001.

[10] DMOZ. Open Directory Project. http://www.dmoz.org/, 2011. [Online; accessed 2011-03-10].

[11] B. J. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, January 2007.

[12] Germanet. A Lexical-Semantic Net for German. http://www.sfs.uni-tuebingen.de/GermaNet/, 2011. [Online; accessed 2011-03-10].

[13] T. Hammond, T. Hannay, B. Lund, and J. Scott. Social bookmarking tools (i): A general review. *D-Lib Magazine*, 11(4), 2005.

[14] D. Helic, M. Strohmaier, C. Trattner, M. Muhr, and K. Lermann. Pragmatic evaluation of folksonomies. In *Proc. of the 21st International World Wide Web conference, WWW '11*, New York, NY, USA, 2011. ACM.

[15] D. Helic, C. Trattner, M. Strohmaier, and K. Andrews. On the navigability of social tagging systems. In *Proc. of 2010 IEEE International Conference on Social Computing*, pages 161–168, Los Alamitos, CA, USA, 2010. IEEE Computer Society.

[16] D. Helic, C. Trattner, M. Strohmaier, and K. Andrews. Are tag clouds useful for navigation? a network-theoretic analysis. *International Journal of Social Computing and Cyber-Physical Systems*, 2011.

[17] P. Heymann and H. Garcia-Molina. Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical Report 2006-10, Stanford InfoLab, April 2006.

[18] J. I. Kiger. The depth/breadth trade-off in the design of menu-driven user interfaces. *Int. J. Man-Mach. Stud.*, 20:201–213, March 1984.

[19] J. Kleinberg. The small-world phenomenon: an algorithm perspective. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, STOC '00, pages 163–170, New York, NY, USA, 2000. ACM.

[20] J. M. Kleinberg. Small-world phenomena and the dynamics of information. In *Advances in Neural Information Processing Systems (NIPS) 14*, page 2001, Cambridge, MA, USA, 2001. MIT Press.

[21] C. Körner, R. Kern, H.-P. Grahsl, and M. Strohmaier. Of categorizers and describers: an evaluation of quantitative measures for tagging motivation. In *Proceedings of the 21st ACM conference on Hypertext and hypermedia*, HT '10, pages 157–166, New York, NY, USA, 2010. ACM.

[22] A. Maedche and S. Staab. Measuring similarity between ontologies. In *Proc. Of the European Conference on Knowledge Acquisition and Management - EKAW-2002. Madrid, Spain, October 1-4, 2002*, volume 2473 of *LNCS/LNAI*, Heidelberg, 2002. Springer.

[23] C. Marlow, M. Naaman, D. Boyd, and M. Davis. Ht06, tagging paper, taxonomy, flickr, academic article, to read. In *Proc. Seventeenth Conference on Hypertext and Hypermedia (Hypertext 2006)*, HT'06, pages 31–40, USA, NY, 2006. ACM.

[24] D. Miller. The depth/breadth tradeoff in hierarchical computer menus. In *Proceedings of the Human Factors Society*, 1981.

[25] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

[26] O. Nov and C. Ye. Why do people tag?: motivations for photo tagging. *Commun. ACM*, 53:128–131, July 2010.

[27] A. Plangprasopchok, K. Lerman, and L. Getoor. From saplings to a tree: Integrating structured metadata via relational affinity propagation. In *Proceedings of the AAAI workshop on Statistical Relational AI*, Menlo Park, CA, USA, July 2010. AAAI.

[28] M. Strohmaier. Purpose tagging - capturing user intent to assist goal-oriented social search. In *SSM'08 Workshop on Search in Social Media, in conjunction with CIKM'08*, USA, Napa Valley, 2008.

[29] M. Strohmaier, C. Koerner, and R. Kern. Why do users tag? detecting users' motivation for tagging in social tagging systems. In *4th International AAAI Conference on Weblogs and Social Media (ICWSM2010)*, USA, Washington, DC, 2010.

[30] C. Trattner. Querycloud: Automatically linking related documents via search query (tag) clouds. In *Proceedings of IADIS International Conference WWW/Internet 2010*, ICWI'10. IADIS, 2010.

[31] C. Trattner. Improving tag cloud navigability: A comparative study. In *Proc. of the 33th International Conference on Information Technology Interfaces*, Cavtat, Dubrovnik, Croatia, 2011. IEEE.

[32] C. Trattner, H. Denis, and M. Strohmaier. On the construction of efficiently navigable tag clouds using knowledge from structured web content. *Journal of Universal Computer Science*, 2011.

[33] C. Trattner, I. Hasani-Mavriqi, D. Helic, and H. Leitner. The austrian way of wiki(pedia)!: development of a structured wiki-based encyclopedia within a local austrian context. In *Proceedings of the 6th International Symposium on Wikis and Open Collaboration*, WikiSym '10, pages 1–10, New York, NY, USA, 2010. ACM.

[34] C. Trattner and D. Helic. Linking related documents: combining tag clouds and search queries. In *Proceedings of the 10th international conference on Web engineering*, ICWE'10, pages 486–489, Berlin, Heidelberg, 2010. Springer-Verlag.

[35] C. Trattner, C. Körner, and D. Helic. Enhancing the navigability of social tagging systems with tag taxonomies. In *Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies*, i-KNOW '11, pages 18:1–18:8, New York, NY, USA, 2011. ACM.

[36] P. Zaphiris. Depth vs breadth in the arrangement of web links. In *Proceedings of the 44th Annual Meeting of the Human Factors and Ergonomics Society*, pages 139–144, Ontario, Canada, 2000.