Enhancing Taxonomies by Providing Many Paths*

Xiaoguang Qi, Dawei Yin, Zhenzhen Xue, Brian D. Davison
Department of Computer Science & Engineering
Lehigh University
Bethlehem, PA 18015 USA
{xiq204,day207,zhx208,davison}@cse.lehigh.edu

October 2010

Abstract

A taxonomy organizes concepts or topics in a hierarchical structure and can be created manually or via automated systems. One major drawback of taxonomies is that they require users to have the same view of the topics as the taxonomy creator. That is, when a user follows a top-down path to find the specific topic of her interest, she has to make choices along the constrained sequence that is present in the hierarchy. As a result, users who do not share that mental taxonomy are likely to have additional difficulties in finding the desired topic. Although this problem can be reduced by remedies like cross-topic links, such approaches break the hierarchical structure and greatly increase human editing cost. Faceted search/browsing has also been proposed to address this problem; however, identifying facets in large scale datasets is a significant challenge.

In this paper, we propose a new approach to taxonomy expansion which is able to provide more flexible views. Based on an existing taxonomy, our algorithm finds possible alternative paths and generates a new, expanded taxonomy with flexibility in user browsing choices. In experiments on the dmoz Open Directory Project, the rebuilt taxonomies show favorable characteristics (more alternative paths and shorter paths to information). User studies show that our expanded taxonomies are preferred compared to the original.

1 Introduction

A taxonomy organizes concepts or topics in a hierarchical structure. The dmoz Open Directory Project[1] and the Yahoo! Directory[2] are two well known examples which, with the help of many human editors, organize what are considered to be good quality web pages into topical taxonomies. Figure 1 illustrates what dmoz ODP looks like. Besides being created and maintained manually, taxonomies can also be created by automated systems. Such systems often group objects together based on their syntactic similarities or distance in the feature space. Automatic taxonomy generation can reduce the enormous amount of human effort and thus speed up the process. However, automatically generated taxonomies do not always match a human's preexisting mental image about the "world"; and therefore are not always easy for a human to understand.

A taxonomy (or hierarchy) is usually constructed based on supertopic-subtopic, or parent-child relationships. At any topic in the hierarchy, following a branch to get to a topic at a lower level means adding another constraint to its parent topic. When a user seeks information in a taxonomy, she usually starts from the root ("everything"), and narrows down the topic by choosing one child under the current topic until she finds what she needs. Unlike keyword search, seeking information in a taxonomy does not require the user to formulate her information need into search keywords. The names of topics (and sometimes descriptions, too) serve as guidance for the user.

*Technical Report LU-CSE-10-005, Dept. of Computer Science and Engineering, Lehigh University, Bethlehem, PA, 18015. A summary of this report is published as a short paper [18].

[1]http://www.dmoz.org/

[2]http://dir.yahoo.com/

One major drawback of taxonomies is that they require users to have the same view of the topics as the taxonomy creator. That is, when a user follows a top-down path to find the specific topic of her interest, she has to make choices along the constrained sequence that is present in the hierarchy. As a result, users who do not share that mental taxonomy are likely to have additional difficulties in finding the desired topic. For example, in one taxonomy, information about Emacs, an open source text editor, can be organized under /Software/OpenSource/Editors/Emacs. Such a taxonomy will not be helpful for a user who looks for information about Emacs but does not know that Emacs is an open source package. This problem can be somewhat reduced by remedies like cross-topic links (as used in ODP). However, after adding such links, the target nodes of the links will have more than one parent, making the taxonomy no longer a tree. Under this approach, nodes are not replicated—links are just added to the graph. Logically, such links are only appropriate when the alternative path also applies to all descendants of the topic. Furthermore, such links bring dependencies among topics and increase editing cost: editing one topic may result in changes in its linked topics and then the linked topics of those topics.

In this paper, we propose a new approach to taxonomy expansion which is able to provide more flexible views. Based on an existing taxonomy, our algorithm finds possible alternative paths and generates a new, expanded taxonomy with flexibility in user browsing choices. In our experiments on the dmoz Open Directory Project and a social bookmarking dataset, the rebuilt taxonomies show favorable characteristics (more alternative paths and shorter paths to information). Our algorithm can be used to expand current web hierarchies such like those provided by ODP and Yahoo!.

Our main contributions include:

- an analysis and formulation of existing problems of hypernym/hyponym taxonomies; and
- a new approach to generate flexible taxonomies that is able to work on existing taxonomies.

The rest of this paper is organized as follows. Related approaches in taxonomy generation are reviewed in Section 2. We motivate our work and define the problem we aim to tackle in Section 3. In Section 4, we describe our approach in detail. Experimental results are shown in Section 5. We conclude with a discussion in Section 6.

2 Related Work

We first review existing approaches related to the general problem of assisting data browsing; we then discuss approaches for taxonomy generation.

2.1 Approaches to assist data browsing

How to organize data and present it to users in a meaningful way has been a difficult problem. We briefly review three main-stream methods for dealing with the issue: taxonomies, faceted search/browsing, and tagging systems.

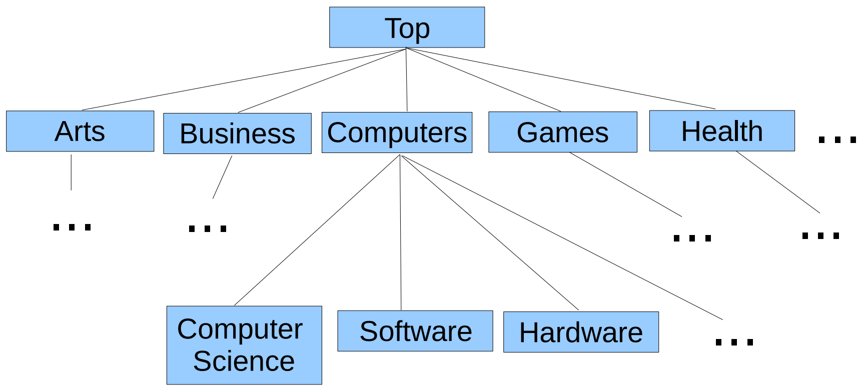

Figure 1: An illustration of ODP structure

Taxonomies have been shown to be useful to some extent in organizing information across various domains, such as organism classification, library classification, disease classification, and web hierarchies. The traditional taxonomies require concepts to be organized in a strict tree structure. Krishnapuram and Kummamuru [10] reviewed existing approaches to automatic taxonomy generation, analyzed the issues with traditional taxonomies, and pointed out their major problem: "most real-world concepts (and subconcepts) are not crisp" in that they usually belong to more than one category. Therefore, it is unnatural to constrain each concept to have exactly one parent. Based on such an analysis, they proposed a new notion – "fuzzy hierarchies", in which "each concept at level k of the hierarchy is a child of all the concepts at level k + 1, albeit to different degrees". Compared with traditional taxonomies, fuzzy hierarchies may better match the fuzzy representation in the human mind. However, they still have strict levels; i.e., users still need to make a fixed sequence of choices to reach a particular leaf. Furthermore, no method has been proposed to generate such a hierarchy.

Ranganathan developed a faceted library classification system called Colon Classification [19, 20], which uses colons to separate identifiers of facets. In this classification scheme, an object is described using a pre-defined set of facets. The introduction of facets into classification systems eliminated the intrinsic restrictions of taxonomies. The same idea powers the modern faceted search/browsing systems, which allows users to filter information from a number of facets in any arbitrary order, as opposed to following a pre-defined path in taxonomies.

Tagging systems became popular along with the rise of social media/social networks. Typical examples of such systems include delicious[3], flickr[4], and bibsonomy[5]. They allow users to assign arbitrary tags to the objects (web pages, photos, scientific publications, etc.), and retrieve objects that are associated with any particular tag. Some also allow sharing of such tags among users. Among the above three methods, the tagging system is the least organized in that it doesn't impose any structural relationship among tags. Such a property makes it easier to assign tags. However, better organized tags (e.g., a tag hierarchy) may enable more effective exploration of information.

2.2 Taxonomy generation

We have briefly reviewed the strength and weakness of taxonomies, faceted search/browsing and tagging systems. Now, we focus on the automatic generation of taxonomies.

A taxonomy is an efficient way to assist users browsing through data collections. It can also be used in classifier training and testing [4, 8, 24, 17], in automatic generation of evaluation datasets for classification approaches [7], in automatic evaluation of search results [1, 2], and in word sense disambiguation [21]. Here, we review existing automatic taxonomy generation approaches by three categories: approaches based on hierarchical clustering using textual content, on associated objects, and on term relationship.

Among the three types of approaches, hierarchical clustering is the most studied. The objects to be clustered range from short queries to whole documents, or sometimes even a collection of documents.

Scatter/Gather [6] is a well-known interactive browsing system. Given a document collection, the system clusters the documents into a number of clusters, and presents the clusters with short summaries to the user. The user selects a subset of clusters of his interest. The system then re-clusters the selected subset of documents, and presents to the user the refined clusters. This iterative process can be repeated until the user finds what he needs. Such an interactive browsing-and-recluster approach is effective on small and medium datasets. On large datasets, the high online cost makes it less effective.

The TaxGen System developed at IBM Germany [14] utilizes bottom-up hierarchical clustering methods to build taxonomies out of text collections. The system is able to expand the generated hierarchies by including new documents using automatic categorization approaches.

Kummamuru et al. [11] proposed an approach to query result clustering based on page titles and snippets. In their work, taxonomies are generated by considering document coverage, compactness of the generated hierarchy, and distinctiveness among sibling nodes. Their experiments showed that the proposed algorithm can help users find information faster.

[3]http://delicious.com/
[4]http://www.flickr.com/
[5]http://www.bibsonomy.org/

The second category focuses on generating taxonomies from tagging systems, taking advantage of human assigned tags for objects. As a typical approach in this category, Heymann and Garcia-Molina [9] proposed to generate taxonomies out of flat tagging systems like `delicious.com`. First, cosine similarities between tags are computed based on the objects they are used to annotate. Then, a graph of tags is generated based on such similarities. After that, starting from the center of the similarity graph, a greedy algorithm selects new edges (and the connected nodes) one by one to add to the taxonomy.

Term co-occurrence is a straight-forward measure for term relatedness, which assumes the asymmetric occurrence relationship between terms indicates their semantic subsumptions. Sanderson and Croft [22] proposed to generate taxonomies from text collections using simple statistics of term co-occurrence. Given a query, a set of keywords are extracted from each result. Then keyword x subsumes y if

$$p(x|y) \geq 0.8 \; and \; p(y|x) < 1. \tag{1}$$

This approach was adopted by Clough et al. [5] in their work on automatic image taxonomy generation. Schmitz [23] also used a similar approach to extract subsumption term pairs from Flickr tags.

Several works used WordNet [13] as a resource to measure term relatedness. Ponzetto and Strube [16] applied filters derived from WordNet to network of Wikipedia categories for extracting a large scale taxonomy containing a large amount of subsumption. Suchanek et al. [25] generated hybrid resource by mapping Wikipedia categories to WordNet. Ponzetto and Navigli [15], presented a method using WordNet subsumption hierarchy to perform category disambiguation, which leads to the integration of Wikipedia and WordNet, in that Wikipedia categories are enriched with accurate sense information from WordNet. Kozareva et al. [27] proposed a weakly supervised algorithm for reading web texts, learning taxonomy terms, and identifying hypernym/hyponym relations, which offers the possibility of automatically generating term taxonomies.

3 Motivation and problem definition

3.1 Motivation

Browsing through a large data collection that is organized in a taxonomy is an interesting while different problem from keyword search. Search is effective when the user knows the name of the target information. When the desired information is a large set of instances (e.g., American sci-fi movies from the 1960s), or something that the user do not recall its name (e.g., the 1980 Steven Spielberg film about a space alien), a generic search often gives poor results. Browsing (or navigating) in taxonomies is usually more effective in such cases.

Typically it is supertopic-subtopic relationships that connect a topic with its child topics in a taxonomy. As natural it may seem, this methodology often poses extra difficulty for both the creator and the user. One reason is that there are often many ways to split a topic. For example, movies can be classified by their genre, country of origin, director, etc. When creating a taxonomy of movies, one may choose to first split by genre then by country of origin, or first by director then by genre. However, no property of any of the above aspects (or facets) by itself is intrinsically a subtopic of another. In fact, they are orthogonal characteristics to some degree. A user may wish to narrow his selection by choosing one property of any reasonably arbitrary facet. Therefore, although there are many reasonable ways to split the movie category, none of them is able to satisfy all users' needs. In this paper, we will refer to this problem as the *multiple facets* problem.

There are at least two ways to address the multiple facets problem. One is to split the topic in question in multiple ways. However, this method, without any fix, will inevitably result in duplicate objects under different subtopics. A simple fix is to treat one group of subtopics as real topics, and others as symbolic links. Figure 2 illustrates a taxonomy generated using this method, which is an actual subgraph of ODP. This method is widely used in ODP to address the multiple facets problem. However, to the best of our knowledge, this method is only used by human editors. No automated system is able to generate taxonomies in similar ways.

Another method to address the multiple facets problem is faceted browsing. Unlike browsing through a tree structure, choosing one option at a time, users are presented with multiple orthogonal facets and offered a chance to narrow the selection by choosing property constraints from multiple facets simultaneously. This method, when implemented properly, is able to address the multiple facets problem well. However, identifying the facets, especially

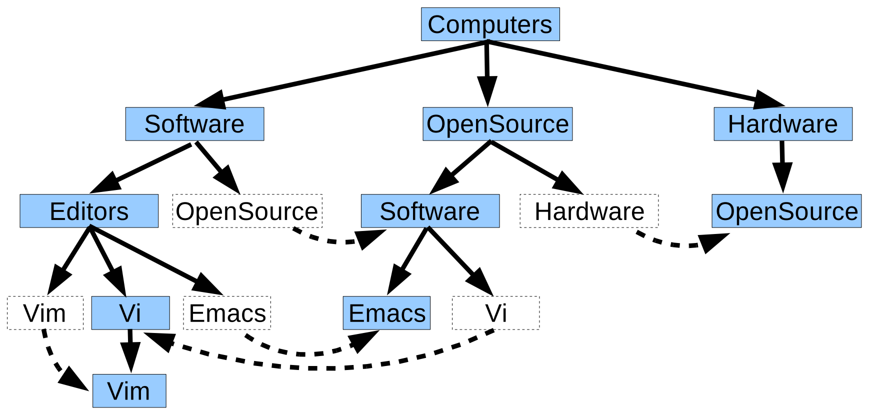

Figure 2: A taxonomy with symbolic links.

from large datasets, is often a challenge for both human and automated systems and still may have too many facets (or even classes of facets) to present at one time. While in this paper we will focus on automating the multiple split method, the results of our approach may be of help to human and automated systems to identify facets and enhance faceted search/browsing.

Another problem of existing taxonomies also originates from the strict supertopic-subtopic relationships. When users need to find a particular object, they have to go all the way along the path from the root to their topic of interest. In this paper, we will refer to this problem as *exhaustive path* problem. For example, even if the Emacs topic can be reached by two alternative paths (i.e., presumably with the multiple facets problem fixed), Software/Editors/OpenSource/Emacs and Software/OpenSource/Editors/Emacs, what if a user who does not know whether the package is open source just wants to find a list of representative text editors? In this case, a link like Software/Editors/Popular/Emacs will come in handy. By presenting popular descendant topics closer to the root, we may reduce the choices a user need to make and thus reduce the time to find desired information. We are interested in back-end algorithms that can facilitate a better presentation.

If the user knows exactly what she is looking for, she can start from a keyword search in the taxonomy, in which case the above problems can be easily solved. However, most users who intend to find answers in a taxonomy usually do not know an appropriate keyword with which to start. The above problems could also be alleviated by building customized taxonomies on a user basis. However, the user profile that is required in a personalized approach is difficult to acquire. Furthermore, a user's interest and browsing preference may change frequently. Given the size of taxonomies we target, how to update the taxonomy to match the user's change in interests is another challenging problem in both efficiency and effectiveness. Therefore, instead of building personalized taxonomies for each user, we propose a method to build an expanded taxonomy with more branches to help users find information easier and faster in the given dataset.

3.2 Problem definition

As mentioned in previous sections, we aim to build a flexible hierarchy which can expose different interpretations of the data. A user traversing the hierarchy to find a target item can be viewed as a series of restrictions which narrow the search scope step by step. At each step, the user is provided a set of candidate topics to further narrow the scope. If we call the internal nodes in a taxonomy the *topics* (e.g., the topics in ODP), and call what is to be classified the *objects* (e.g., the outgoing links of ODP), we can formalize the problem as follows.

Given a graph $G = (V, E, O, R)$,
where V = $\{v$— v is a topic in the taxonomy$\}$,
E = $\{(v_1, v_2)|v_1, v_2 \in V\}$,
O = $\{o$— o is an object in the taxonomy$\}$,
R = $\{(v, o)|v \in V$, and $o \in O\}$;

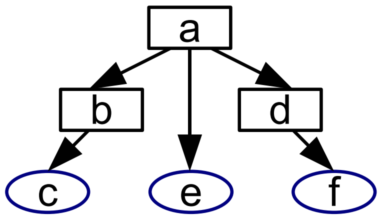

Figure 3: A simple predefined taxonomy

Output a graph $G' = (V, E', O, R)$,
where E' = $\{(v_1, v_2)|v_1, v_2 \in V\}$.

Note that we do not require the input graph to be a tree in order to allow for more relaxed relationships, such as reference links across different branches.

4 Approach

So far, we have formulated the taxonomy generation problem as a graph transformation problem. We will solve this graph transformation problem in two steps: break down the taxonomy hierarchy into a set of tag-object tuples and then rebuild a flexible taxonomy from these tuples.

First, we break down the input taxonomy into a set of tag-object tuples. We consider the original tree structure and treat each internal node (topic) as a tag. We assume the same topic name always has the same meaning. That is, the same topic name appearing in different locations in the original taxonomy is considered to be the same tag. Then each object in the taxonomy (i.e., URLs in the ODP case) is associated with all the tags from its ancestors in the tree. For example, in the original hierarchy in Figure 3, there are three objects c, e and f, and three topics, a, b, and d. After we tag each object with its ancestors, we get $tag(e) = \{a\}$, $tag(c) = \{a, b\}$, $tag(f) = \{a, d\}$. We also overload this function to define the tag set of multiple objects as the union of their tag sets, i.e., $tag(\{e, c\}) = tag(e) \cup tag(c) = \{a, b\}$. We define $obj(x)$ as the set of objects that are associated with tag x. In this example, $obj(a) = \{c, f, e\}$, $obj(b) = \{c\}$, $obj(d) = \{f\}$. Similar to the tag set of multiple objects, we define the object set of multiple tags as the union of their object sets. In this case, $obj(\{b, d\}) = obj(b) \cup obj(d) = \{c, f\}$. After this, we discard the original taxonomy structure, with only the tags, objects, and their relationships left. Since there are no relationships between two tags or two objects, the problem left for consideration now is a bipartite graph, with tags and objects being two sets of nodes, and each edge connecting exactly one tag and one object. Therefore, we now have

$G = (T, O, E)$
$T = \{t|t$ is a tag$\}$
$O = \{o|o$ is an object$\}$
$E = \{(t, o)|t \in T$,$o \in O$, object o has tag $t\}$

We now transform the taxonomy rebuilding problem into a problem to generate a taxonomy based on a bipartite graph of tags and objects.

Hierarchy problem

Given a bipartite graph $G = (T, O, E)$,
Output a graph $G'' = (V'', E'', O, R'')$

where V'' is a set of topics, E'' is a set of edges connecting topics in V'' to objects in O, and R'' is a set of edges connecting topics to their subtopics.

Although such a problem transformation process seems tedious and arguably unnecessary, we will show in the following that the transformation allows us to adapt solutions to a well-studied problem for our purpose of taxonomy generation.

4.1 Set covering

A straightforward approach to taxonomy generation is a top-down method, in which the topic at each leaf node is split into a number of subtopics until no further split is necessary. Every time a topic needs to be split, if we consider the topics (tags) as the sets, and the associated objects as the objects to be covered, the split problem can be seen as a generalized set covering problem.

In a set covering problem, given a universe O consisting of n objects, and T a set of subsets of O, we say a subset $C \subseteq T$ covers O iff $O = \bigcup_{t \in C} t$. The set covering problem is known to be NP-complete. However, there are greedy approximation algorithms with polynomial time complexity (e.g., Algorithm 1 in [26]). Let H_k denote $\sum_{i=1}^{k} 1/i \approx \ln k$, where k is the largest set size. This algorithm is guaranteed to return a set cover of weight at most H_k times the minimum weight of any cover.

Algorithm 1 Greedy Algorithm for Set Cover (Vazirani, 2001)

```
1: // Input T is a set of tags, and O is a set of objects.
2: Initialize $C \leftarrow \emptyset$.
3: // Loop while C does not cover all objects
4: while $Obj(C) \neq O$ do
5:     Choose $t \in T$ to maximize $f(t)$
6:     Let $C \leftarrow C \cup \{t\}$
7: end while
8: Return $C$
```

$f(t)$ is the objective function which the algorithm aims to maximize. In the original version of the set covering algorithm [26], it is defined as $|Obj(C \cup \{t\}) - Obj(C)|$. We choose to base our approach on this algorithm for two main reasons. First, it is a natural requirement for the generated hierarchy to cover all the objects. Second, in order to generate our desired hierarchy, it is convenient just to extend its objective function to match our purpose, leaving the rest of the algorithm unchanged. In the rest of this section, we will show how we change this function to meet our needs in flexible taxonomy generation.

In our approach, every time a topic is split, we generate three types of subtopics. A group of basic subtopics to cover all the associated objects (in some sense, the most obvious split), one or more groups of orthogonal subtopics to provide alternative paths, and a group of popular subtopics to allow fast access to the most popular descendant topics. We will discuss them in the following subsections.

4.2 The basic group

The original bipartite graph consists of the tags and objects as the two groups of vertices, and the relationships among them as the edges. Starting from a tree with a single node *root*, we iteratively split its leaf nodes to generate the taxonomy.

Suppose we are at the point to split a node (tag) t_{cur}, let T_a be the set of tags that has been used in t_{cur}'s ancestor nodes. The tag set $T' = T - T_a$, the corresponding object set O', and the relationship E' between T' and O' make up a bipartite graph $G'(T', O', E')$, a subgraph of the original bipartite graph $G(T, O, E)$. Let T_c be the subtopics that have already been chosen, O_c the set of objects that have been covered. The method of generating the basic group of subtopics is as follows.

Cover Score. For each $t \in T' - T_c$, the cover score is defined as follows.

$$CoverScore(t) = \frac{\sum_{o \in O' - O_c, and (t,o) \in E'} w(o)}{\sum_{o \in O'} w(o)} \tag{2}$$

The score is normalized so that for any t, $CoverScore(t) \in [0, 1]$. $w(o)$ is a weight of the object o. In this work, it is simply set as 1. In the future, it can be generalized to any importance measure of the object (e.g., the PageRank of the web page).

Tag Similarity Score. Define function F_v that maps a tag to an $|O|$ dimensional vector, in which the ith element of $F_v(t)$

$$F_v(t)_i = \begin{cases} 1 & if(t, o_i) \in E \\ 0 & otherwise \end{cases} \tag{3}$$

Then the similarity between two tags t_1 and t_2 is defined as

$$TagSim(t_1, t_2) = \frac{F_v(t_1) \cdot F_v(t_2)}{|O|} \tag{4}$$

At any time during the process of choosing subtopics, let T_c be the set of subtopics that have been chosen, then the similarity score of any tag t for consideration is

$$TagSimScore(t) = \begin{cases} \frac{|T_c|}{\sum_{t' \in T_c} TagSim(t,t')} & if|T_c| > 0 \\ 0 & otherwise \end{cases} \tag{5}$$

The cover score focuses on how many new objects are covered by adding the tag in question; it does not care about the overlap between the newly added tag and the existing tags. The tag similarity score compensates for that by favoring the tags with less overlap.

Based on the cover score and the tag similarity score, we define the objective function $f_b(t)$ as follows and plug it into Algorithm 1 to calculate the basic group of subtopics.

$$f_b(t) = \alpha \cdot CoverScore(t) + \beta \cdot \frac{\mu}{TagSimScore(t)} \tag{6}$$

μ is a normalization factor, which is set to $min_t[1/TagSimScore(t)]$ in our work. The parameters α and β are used to adjust the impact of different scores. We will tune them in the experiments.

4.3 The extension group

In the previous subsection, we discussed how to generate the first group of subtopics. In the extension group, we aim to discover orthogonal dimensions for alternative views. We exploit two types of information to compute this group.

Impurity Score. We employ the entropy impurity to measure this property of tags. Entropy impurity has been used in decision trees [3]. The impurity of tags in our system is defined as follows. Let O'_t be the set of objects which are covered by t, $O'_t = \{o|(t, o) \in E'$ and $o \in O'\}$. Given the basic tag set T_b generated by the method in Section 4.2, and a tag t, we define $P_t(t_i)$ as the fraction of objects for $t_i \in T_b$ which is

$$P_t(t_i) = \frac{\sum_{(t_i,o) \in E', o \in O'_t} w(o)}{\sum_{o \in O'_t} w(o)} \tag{7}$$

Then, the impurity of a candidate subtopic t given T_b is defined as

$$I(t) = -\sum_{t_i \in T_b} P_t(t_i) \log_2 P_t(t_i) \tag{8}$$

By definition, if all the objects covered by t are uniformly covered by the topics in T_b, it will get the maximum value.

Sibling Score. After a set of subtopics are chosen, the sibling score of a candidate topic t measures how likely t is a sibling topic with the chosen tags. The sibling likelihood can be obtained from various sources, such as Wikipedia, WordNet, or the generic web. In this work, we obtain this information from the original taxonomy. For each pair of tags, we examine the positions of the two tags within the original hierarchy and count the number of times that these two tags share the same parent. We define the function $parent(t)$ as the the set of parent topics of t. The sibling function between any two tags is defined as follows.

$$Sibling(t_1, t_2) = \log_2 (1 + |parent(t_1) \cap parent(t_2)|) \tag{9}$$

Then, $Sibling(t_1, t_2)$ is normalized over all tag pairs so that they sum to 1. We note the normalized results as $Sibling_{norm}(t_1, t_2)$.

Based on the pairwise sibling likelihood, the sibling score of a candidate subtopic t given a set of already chosen subtopics T_c is computed as follows. For any t, $SiblingScore(t) \in [0, 1]$.

$$SiblingScore(t) = \begin{cases} \frac{\sum_{t' \in T_c} Sibling_{norm}(t,t')}{|T_c|} & if|T_c| > 0 \\ 0 & otherwise \end{cases} \tag{10}$$

Similar to the generation of the basic group, when generating the extension groups, we still use the greedy approximation algorithm of set covering problem, only with a different objective function.

$$f_e(t) = \gamma \cdot CoverScore(t) + \delta \cdot SiblingScore(t) + \eta \cdot I(t) \tag{11}$$

The weights γ, δ, η are used to adjust the impact of different scores. We will tune them in the experiments.

Depending on the particular situation, the extension set may need many topics to cover all objects. In order not to overwhelm users with too many choices, we slightly change the stopping criterion to let it be satisfied when the number of subtopics exceeds k, where k is set to be $1.5 \times |T_b|$.

Above we presented how to generate one group of extension topics. After this, a new group of extension topics can be generated by removing the selected topics from the candidate, and running the above process again.

4.4 The shortcut group

We discussed the generation of the extension groups in the previous subsection. The extension groups aim to address the multiple facets problem. In this subsection, we discuss how to address the exhaustive path problem using a group of shortcut topics.

Popularity Score. The idea of the shortcut group is to increase the visibility of popular/important topics by putting them closer to the root, and thus reduce the time for users to find them. There are a variety of metrics to assess whether a descendant topic should be promoted. Here, we use the number of times the topic name is used as a tag of the bookmarks in delicious. The popularity score is defined as follows:

$$popularity(t) = \frac{|bookmark(t)|}{\max_{t \in T} |bookmark(t)|} \tag{12}$$

where $|bookmark(t)|$ is the number of distinct bookmarks which are associated with tag t in Delicious. The value of popularity also is normalized to fit in the range [0,1].

We again use Algorithm 1 but with a different objective function to generate the shortcut group:

$$f_s(t) = popularity(t) \tag{13}$$

Although the data in Delicious changes continuously, it is safe to assume that the relative frequencies of tags are somewhat stable over time. Given that the Delicious data is only used as a *rough* estimate of tag popularity/importance, it is not necessary to always keep it up-to-date. In addition, such a measure can also be estimated from other sources like Wikipedia, query log, and click log of a web hierarchy. Once there is a significant change in tag popularity, the hierarchy can be regenerated by running the algorithm on the updated data. Although directly linking a topic to its indirect descendants may break the logical consistency, we expect this method can help a user to find popular topics faster. Furthermore, putting such shortcuts in a separate group can reduce any user-perceived confusion.

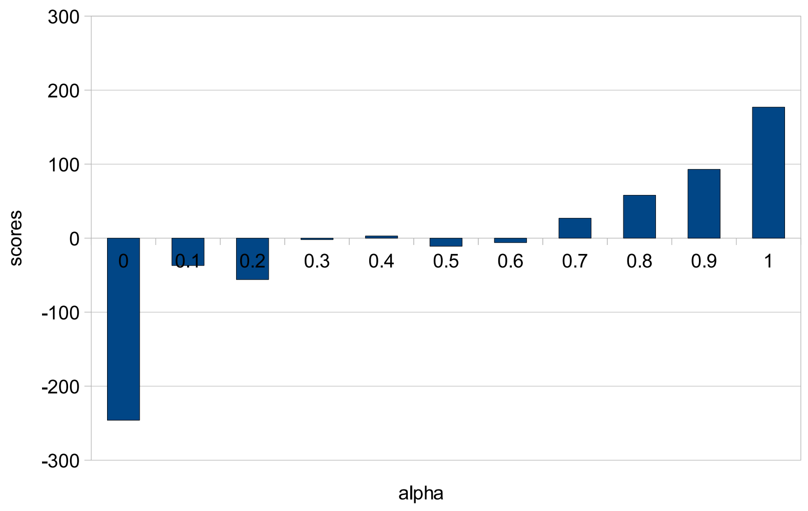

Figure 4: User evaluation results when tuning parameter to generate the basic set.

5 Experiments

5.1 Datasets and experimental setup

We used an ODP dump from April 2009 to test our algorithm. It contained 4,225,962 external links and 763,377 categories. We consider categories with the same name to be the same tag, which gives us 292,550 tags. In the original ODP hierarchy, the average depth is 7.25 and maximum depth is 14.

We used two subsets of ODP to experiment with our approach. One is “Top/Computers/Computer_Science/” and all its subcategories and objects, which contains 184 unique tags and 2061 URLs. The other contains “Top/Science/” and all its subcategories and objects, with 9415 tags and 100,109 URLs. We also collected popularities for 31,342 tags from Delicious.

5.2 Parameter tuning

So far, we have introduced a taxonomy generation method with parameters. α and β are used to combine the cover score and tag similarity score when generating the basic set. We also used γ, δ, and η to balance the weight of cover score, sibling score, and impurity score. Here we tune the parameters in our method before generating the final taxonomy.

We changed the value of α and β by 0.1 in each step while maintaining their sum to be one. Thus we generated 11 taxonomies. We asked users to conduct pairwise comparison of the subtopics under one particular category at a time, and record their rating. Users can choose to rate quality of subtopics in one taxonomy better than, worse than, or almost the same as another. They can also choose not to make a judgment when uncertain. When a user considers the subtopics in one taxonomy better than another, we add one point to the former taxonomy’s score, and subtract one from the latter. Nine users participated in the evaluation. The average scores across users are plotted in Figure 4, showing a clear trend that the more weight we put on covering score, the better result we get. The best result is achieved when only using covering score.

We continue using this approach to tune the parameters to generate the extension set. We fixed $\alpha = 1$ and $\beta = 0$, then changed each of the other parameters by 0.1 at each step, while maintaining the sum to be one. Although there are significant difference in user ratings from different parameter settings, the result did not show any clear pattern. The best score is achieved when $\gamma = 0.8$, $\delta = 0.1$, and $\eta = 0.1$.

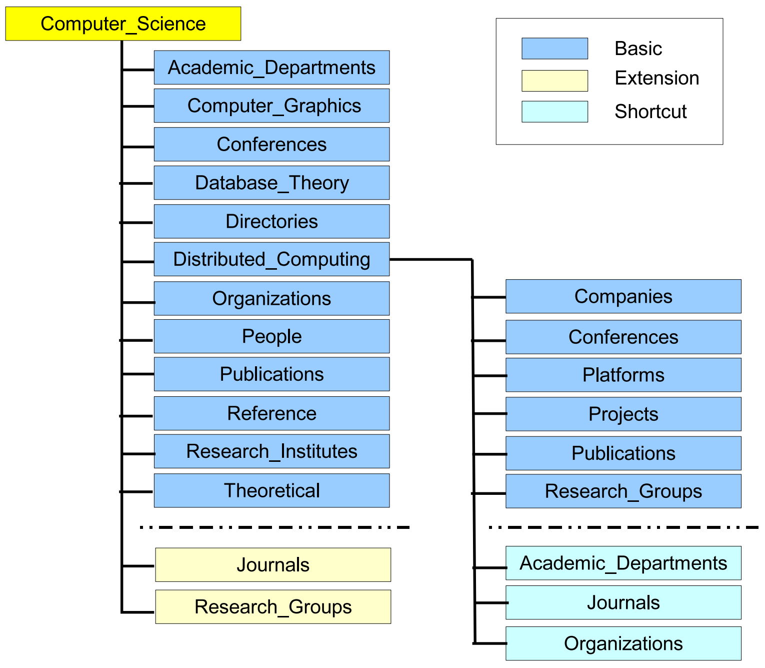

Figure 5: Automatically generated subtree under Top/Computers/Computer_Science

5.3 Taxonomy comparison and analysis

After the parameters are tuned, we applied our algorithm on the two subsets of ODP using the best parameter setting. Figure 5 and Figure 6 illustrate the hierarchies generated by our algorithm for "Top/Computers/Computer_Science" and "Top/Science", respectively. From the results, we can see our automatically generated hierarchies are reasonable. For comparison, we implemented the approach proposed by Heymann and Garcia-Molina [9], which will be referred to as "Communal Taxonomies". The generated hierarchy is illustrated in Figure 7. The algorithm output depends on a tag similarity threshold. We tried our best to tune it in order to get the best result. However, the generated hierarchy does not seem to be reasonable. It has 183 categories and 3 levels, with 25.9 subcategories per category on average. We think the following can provide one possible explanation. The "Communal Taxonomies" approach is designed for generating taxonomies out of flat tagging systems, as opposed to a tag-object dataset extracted from an existing hierarchy. It starts building the hierarchy from the center of the tag similarity graph. If the original hierarchy is dramatically unbalanced, the center may shift from the original root to its most developed subtree, resulting in a more balanced yet less reasonable taxonomy.

We expect that the subtree consisting of only the basic sets generated by our algorithm ("basic subtree") should be similar to the original ODP hierarchy. Our observation of the resulting hierarchies matches this hypothesis. In order to test it analytically, we implemented the taxonomy comparison metric called "taxonomy overlap" proposed by Maedche et al. [12]. The context of a concept in a taxonomy is defined by all its superconcepts and subconcepts. Then for each common concept across two taxonomies, its overlap is computed by the Jaccard similarity of its context in each taxonomy. The overall taxonomy overlap of two taxonomies is the average of individual overlap over all concepts. The value of taxonomy overlap ranges between 0 and 1, where a higher score means more similar. We computed the overlap between ODP and our basic subtree, as well as the overlap between ODP and the one generated by "Communal Taxonomies". The overlap between ODP and our basic subtree is 0.98, meaning they are extremely similar. The overlap is 0.77 for ODP and "Communal Taxonomies". One may question the necessity of the method generating the basic tag group. Given that the generated basic subtree highly resembles the original hierarchy, a simple alternative is to directly operate on the original hierarchy instead of regenerating it. However, the method proposed in Section 4.2 is useful for at least two reasons. First, we need to generate basic groups of subtopics under the extension group of topics, which are not present in the original tree. Second, such a method makes it possible to

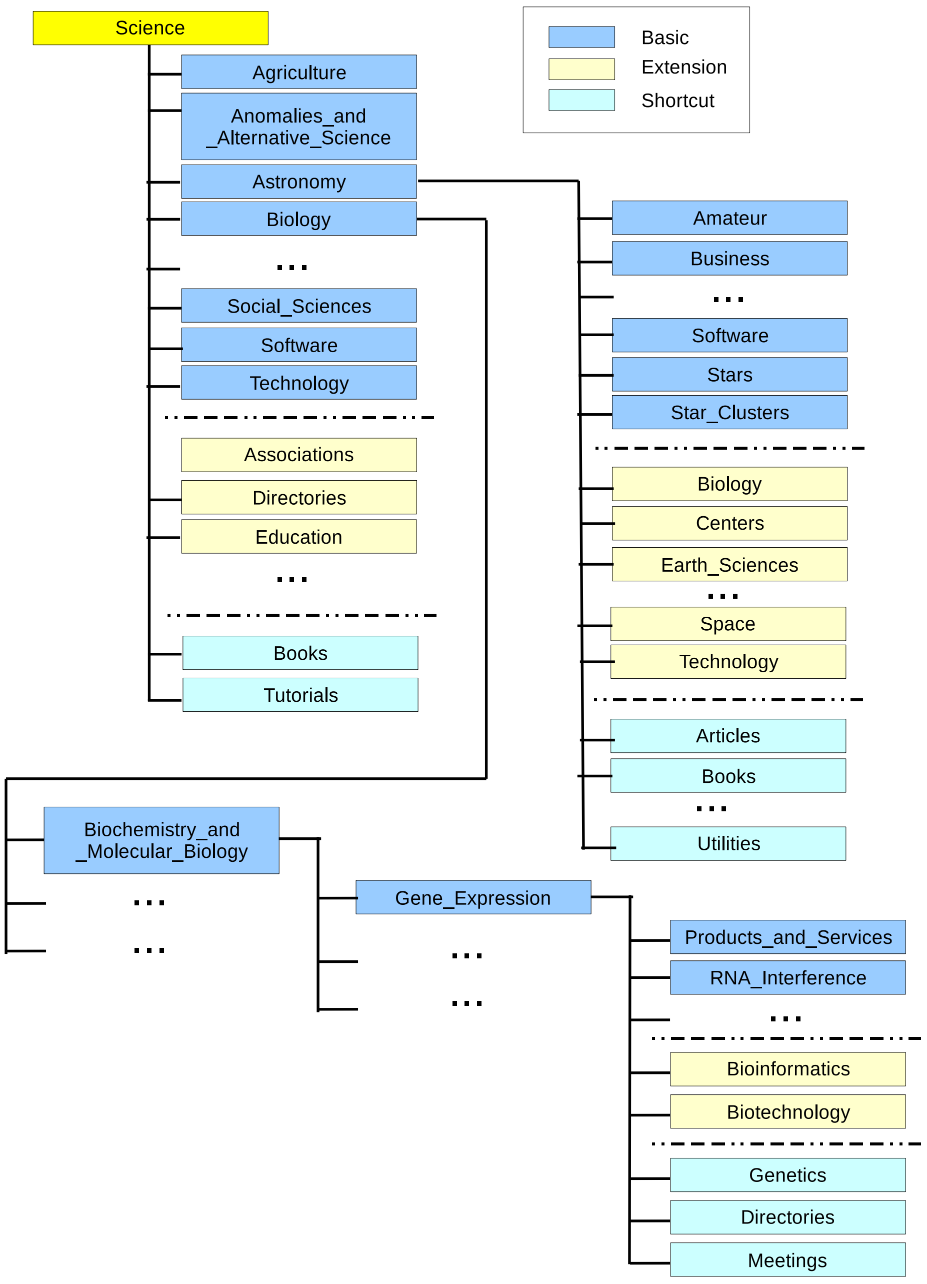

Figure 6: Automatically generated subtree under Top/Science

extend our approach to work on not only corpora with existing tree structures, but on flat tagging systems found in social bookmarking systems like Delicious.

5.4 User studies

We conducted two types of user studies to evaluate user satisfaction of our hierarchy.

In the first user study, we compare three hierarchies: our hierarchy, the ODP hierarchy and the ODP hierarchy with random extensions. The random extensions are generated by randomly selecting descendant categories and promote

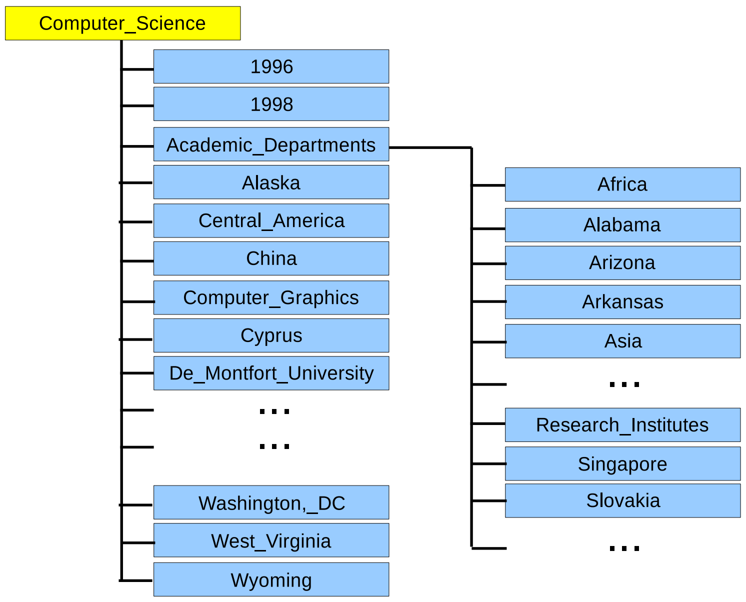

Figure 7: A subtree under Top/Computers/Computer_Science generated by "Communal Taxonomies"

them as the direct children of the current category. This random selection algorithm is a variation based on reservoir sampling. While a generic random sampling method gives each candidate equal probability to be selected, our method favors descendant categories that are closer to the current category (i.e., more generic topics have higher chances to be selected). The algorithm is shown in Algorithm 2. This random extension is used as an alternative baseline than the original hierarchy to eliminate the possible effect that users may simply choose the hierarchy with more branches. In total, we randomly selected 27 categories from "Top/Computers/Computer_Science" and 50 categories from "Top/Science" to be evaluated by users. For each category, we randomly reordered the sequence of these three hierarchies before presenting them to the users. For each of the selected categories, we only show a local view of each hierarchy, i.e., the current category and its children. Evaluators are asked if the quality of the child concepts are good(3), fair(2), or bad(1).

Algorithm 2 Reservoir Sampling with a Bias Towards High Level Topics

```
1:  {INPUT: an array candidate[n] consists of n candidate topics;}
2:  { an integer k, the number of topics to be selected; and,}
3:  { a function getLevel, given a topic, returns its level.}
4:  {OUTPUT: an array selected[k] consists of the k selected topics.}
5:  for $i = 1$ to $k$ do
6:      $selected[i]$ = $candidate[i]$
7:  end for
8:  for $i = k + 1$ to $n$ do
9:      r = random (i) {Generate a random integer between 1 and i, inclusively.}
10:     if $r$¡=$k$ and getLevel(candidate[i]) ¡= getLevel(candidate[r]) then
11:         $selected[r]$ = $candidate[i]$
12:     end if
13: end for
```

Figure 8 shows the average score for each of the hierarchies. Our hierarchy is considered to be much better than the random extension hierarchy, and is even better than the manually created hierarchy from ODP. In the "Top/Science" dataset, our extension branches give users positive impressions and the satisfactory score

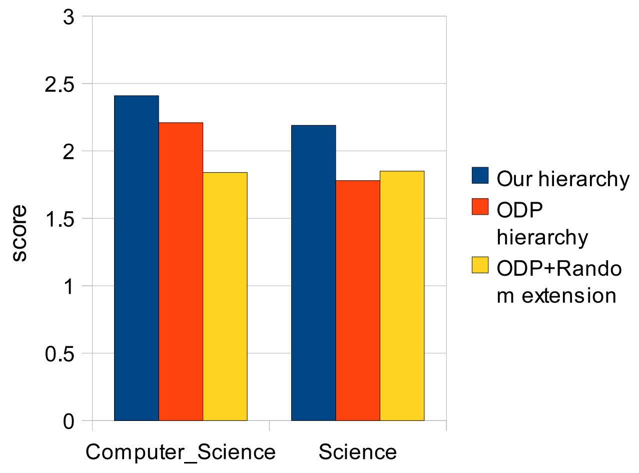

Figure 8: Results of user study on quality of hierarchies.

judged by users is improved by 23% when compared with the original ODP. Similar results can be found on the "Top/Computers/Computer_Science" dataset.

We also designed specific tasks, asking users to find particular topics in the generated taxonomies. There are two main types of tasks. The first is to find the subtopics under some given topics. The second is to find a specific topic. Three users participated in this experiment.

For the first kind of task, under a given level, if there is the target topic presented as a choice, the user can just click it and enter into the subtopic list. However, if there is no such alternative choice, the only way to find the subtopics of this topic is to try every possible branch, and find whether these branches contain this topic. In this case, our algorithm will be very helpful. This is because our algorithms can mine this kind of topic, which is contained by several other topics (these can be treated as potential subtopics of the current topic).

In the second kind of task, the user should find the exact topics. Our algorithms can provide extra paths to give user more options. So, intuitively, the user should find the topic faster on average. when a user tries to find a certain topic, he must formulate a model of how to find it. For example, the user is trying to find C, and $A->B->C$ is his original idea. However, the system only provides $B->A->C$, forcing the user to discover this structure and change his mental model of the topic structure. Our algorithms can additionally provide $A->B->C$ to help fit this kind of user. Thus, on this task, our algorithms can also help users find what they want faster.

In our experiment, we design five tasks, which are

1. Find software related links.
2. Find fields which contain research groups in this web site.
3. Find journal related links.
4. Find theoretical publications.
5. Find conferences in 2008.

Task 1, task 2, and task 3 belong to the first type of task (subtopic task); task 4 and task 5 belong to the second type of task (specific topic task). Figure 9(a) shows the number of clicks of users for each task and Figure 9(b) shows the elapsed time of users for each task. We can see that the two figures are very similar; that is, if the number of clicks is higher, the elapsed time is also higher. For the subtopic task (task 1, task 2 and task 3), our hierarchy can outperform the original ODP hierarchy consistently. These results verify our hypothesis. The comparison of click counts for task 4 and 5 are extremely consistent. In task 4, in ODP hierarchy, the "Publication" is under some subtopics of "Theoretical",

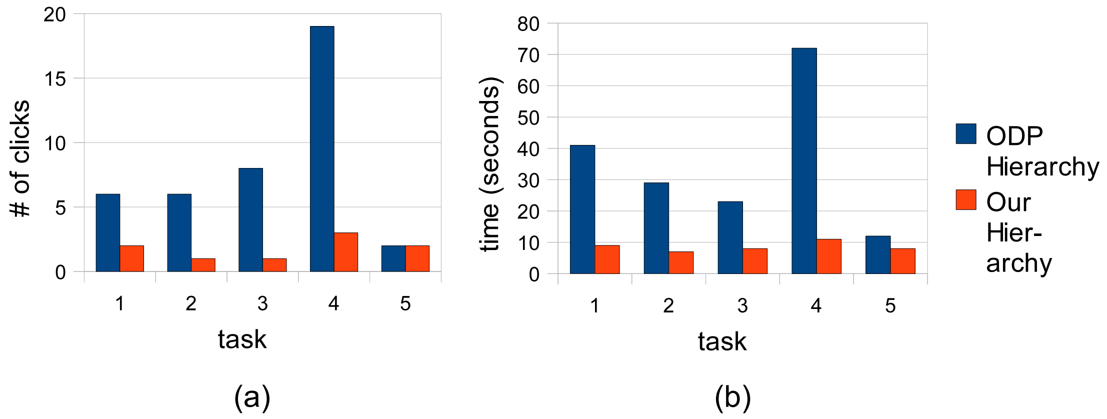

Figure 9: Average number of click counts and time for users to find designed items

such as “Complexity_Theory”. So if users want to find the “Publication” of “Theoretical”, they have to try subtopics to find “Publication”. In our hierarchy, we have promoted “Publication” to the same level of “Theoretical”. In task 5, our algorithm provides one extra path to the target, but the original hierarchy already provides a path which can be easily found by users. So the used time and the number of clicks for two hierarchies are similar. Note that these tasks are designed to demonstrate the advantages of our generated taxonomy. We expect the difference to be less dramatic on more generic tasks.

6 Discussion and Conclusion

In conclusion, we proposed a new model to automatically expand an existing taxonomy by providing more paths. Our experiments show that our approach is able to generate a more flexible and comprehensive hierarchy from an existing hierarchy, leading to significant reductions in user effort and time for hierarchy-centric task completion.

In this work, we assumed that category names are unique. However, this is not always the case; a word may have different meanings in different contexts. Polysemy detection and disambiguation may be of help in this situation. Another issue is that people may use different terms (synonyms) to express the same meaning; our current approach does not take such situations into account.

In the future, we plan to adjust our method so that it can generate flexible taxonomies out of a collections of arbitrarily tagged documents like Delicious and BibSonomy. In addition, with the help of appropriate tag suggestion methods, we plan to apply this idea to any generic collections where tags may not be already available.

Acknowledgments

We thank Liangjie Hong for helpful discussions, and the people who participated in our user study for their valuable input. The random extension generation method (Algorithm 2) was inspired by helpful discussions with Cliff Brunk. This material is based upon work supported by the National Science Foundation under Grant Number IIS-0545875.

References

[1] S. M. Beitzel, E. C. Jensen, A. Chowdhury, D. Grossman, and O. Frieder. Using manually-built web directories for automatic evaluation of known-item retrieval. In *Proceedings of the 26th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–374. ACM Press, July 2003.

[2] S. M. Beitzel, E. C. Jensen, A. Chowdhury, and D. A. Grossman. Using titles and category names from editor-driven taxonomies for automatic evaluation. In *Proceedings of the 12th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 17–23. ACM Press, 2003.

[3] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth & Brooks/Cole, Pacific Grove, CA, 1984.

[4] S. Chakrabarti, B. E. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of ACM SIGMOD Int'l Conf. on Management of Data*, pages 307–318, 1998.

[5] P. Clough, H. Joho, and M. Sanderson. Automatically organising images using concept hierarchies. In *Proceedings of the SIGIR Workshop on Multimedia Information Retrieval*, 2005.

[6] D. R. Cutting, D. R. Karger, J. O. Pederson, and J. W. Tukey. Scatter/Gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 318–329, 1992.

[7] D. Davidov, E. Gabrilovich, and S. Markovitch. Parameterized generation of labeled datasets for text categorization based on a hierarchical directory. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 250–257. ACM Press, 2004.

[8] J. Fürnkranz. Exploiting structural information for text classification on the WWW. In *Proc. of the 3rd Symp. on Intelligent Data Analysis (IDA)*, volume 1642 of *LNCS*, pages 487–497. Springer-Verlag, 1999.

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

[10] R. Krishnapuram and K. Kummamuru. Automatic taxonomy generation: Issues and possibilities. In *Fuzzy Sets and Systems - IFSA*, page 184. Springer, 2003.

[11] K. Kummamuru, R. Lotlikar, S. Roy, K. Singal, and R. Krishnapuram. A hierarchical monothetic document clustering algorithm for summarization and browsing search results. In *Proc. 13th Int'l Conf. on World Wide Web*, pages 658–665, 2004.

[12] A. Maedche and S. Staab. Comparing ontologies - similarity measures and a comparison study. In *Proc. of EKAW*, number 2473 in LNCS. Springer, 2002.

[13] G. A. Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[14] A. Möller, J. Dörre, P. Gerstl, and R. Seiffert. The TaxGen framework: Automating the generation of a taxonomy for a large document collection. In *Proc. 32nd Annual Hawaii Int'l Conference on System Sciences (HICSS)*, volume 2, page 2034, 1999.

[15] S. P. Ponzetto and R. Navigli. Large-scale taxonomy mapping for restructuring and integrating Wikipedia. In *Proceedings of the 21th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2083–2088, July 2009.

[16] S. P. Ponzetto and M. Strube. Deriving a large scale taxonomy from Wikipedia. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI)*, pages 1440–1445. AAAI Press, 2007.

[17] X. Qi and B. D. Davison. Classifiers without borders: Incorporating fielded text from neighboring web pages. In *Proc. 31st Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 643–650, July 2008.

[18] X. Qi, D. Yin, Z. Xue, and B. D. Davison. Choosing your own adventure: Automatic taxonomy generation to permit many paths. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1853–1856, Oct. 2010.

[19] S. R. Ranganathan. *Colon Classification*. Madras Library Association, 1933.

[20] S. R. Ranganathan. *Classification, Coding, and Machinery for Search*. UNESCO, Paris, 1950.

[21] P. Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130, 1999.

[22] M. Sanderson and B. Croft. Deriving concept hierarchies from text. In *Proc. 22nd Annual Int'l ACM SIGIR Conf. on Research and Dev. in Information Retr.*, pages 206–213, 1999.

[23] P. Schmitz. Inducing ontology from flickr tags. In *Proc. of the WWW2006 Workshop on Collaborative Web Tagging*, 2006.

[24] S. Slattery and T. Mitchell. Discovering test set regularities in relational domains. In *Proc. of the 17th Int'l Conf. on Machine Learning (ICML)*, 2000.

[25] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, pages 697–706. ACM, 2007.

[26] V. V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[27] E. H. Zornitsa Kozareva and E. Riloff. Learning and evaluating the content and the structure of a term taxonomy. In *AAAI 2009 Spring Symposium: Learning by Reading and Learning to Read*. AAAI Press, 2009.