Choosing Your Own Adventure: Automatic Taxonomy Generation to Permit 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

ABSTRACT

A taxonomy organizes concepts or topics in a hierarchical structure and can be created manually or via automated systems. A major drawback of taxonomies is that they require users to have the same view of the topics as the taxonomy creator. Users who do not share that mental taxonomy are likely to have difficulty in finding the desired topic. 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 an expanded taxonomy with flexibility in user browsing choices. In experiments on the dmoz Open Directory Project, the rebuilt taxonomies provide more alternative paths and shorter paths to information. User studies show that our expanded taxonomies are preferred compared to the original.

Categories and Subject Descriptors: H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—*Information filtering*; I.5.2 [Pattern Recognition]: Design Methodology—*Classifier design and evaluation*

General Terms: Algorithms, Performance

Keywords: Taxonomy Generation, Hierarchies

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. Taxonomies can be created and maintained manually or automatically. Existing automatic taxonomy generation approaches can be divided into three categories: approaches based on hierarchical clustering using textual content, approaches based on associated objects, and on term relationships. Examples of the first category include the TaxGen System [6] and the query result clustering approach proposed by Kummamuru et al. [4]. The second category focuses on generating taxonomies from tagging systems [3], taking advantage of human assigned tags for objects. Approaches in the third category use term co-occurrence as the measure for term relatedness, assuming the asymmetric occurrence relationship between terms indicates their semantic subsumptions [8, 2]. Our full report [7] on this work contains a more comprehensive review of the broad data organization problem and taxonomy generation approaches.

A taxonomy (or hierarchy) is usually constructed based on supertopic-subtopic, or hypernym-hyponym relationships. 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. 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.

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 more flexibility in user browsing choices. In our experiments on the dmoz Open Directory Project, 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 as those provided by Yahoo! and the dmoz ODP.

Our main contributions include:

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

2. MOTIVATION

Browsing through a taxonomy supports the user differently than 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 does not recall its name (e.g., the 1980 Steven Spielberg film about a space alien), browsing in taxonomies is usually more effective than search.

Typically supertopic-subtopic relationships connect a topic with its child topics in a taxonomy. As natural it may seem, this form often poses extra difficulty for both the creator and the user. One

[1] http://www.dmoz.org/
[2] http://dir.yahoo.com/

CIKM'10, October 26–30, 2010, Toronto, Ontario, Canada.

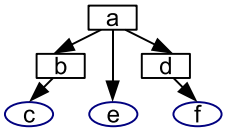

Figure 1: A simple predefined taxonomy.

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. They are orthogonal characteristics in that none of them is logically a subtopic of another. 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. This problem can be addressed by adding symbolic links or introducing faceted browsing. However, both solutions come at a significant extra cost.

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 the *exhaustive path* problem. For example, even if the Emacs topic can be reached by two alternative paths, 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 needs to make and thus reduce the time to find desired information.

These problems could also be alleviated by building customized taxonomies on a per-user basis. However, the user profile is difficult to acquire. Furthermore, a user's interest and browsing preference may change frequently. 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 a given dataset.

3. APPROACH

We propose a method to solve the taxonomy expansion problem: 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 1, 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 can also 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 a set of tags T, a set of objects O, and a set of edges E that connect tags in T to objects in O.

Now we pose the taxonomy expansion problem as 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.

3.1 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 [9]).

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 ← ∅.
3: // Loop while C does not cover all objects
4: while Obj(C) ≠ O do
5:     Choose t ∈ T to maximize f(t)
6:     Let C ← C ∪ {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 [9], it is defined as $|Obj(C \cup \{t\}) - Obj(C)|$. We choose to base our approach on this algorithm for two 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 discuss each in the following subsections.

3.2 The basic group

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 have 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 and O_c the set of objects that have been covered. For each $t \in T' - T_c$, we define the objective function $f_b(t)$ as follows and plug it into Algorithm 1.

$$f_b(t) = \frac{\sum_{o \in O' - O_c, and (t,o) \in E'} w(o)}{\sum_{o \in O'} w(o)} \quad (1)$$

Here, $w(o)$, the weight of the object o, is always set to 1, although it can be generalized to any importance measure (e.g., the page's PageRank). We denote the tags selected for the basic group as T_b.

3.3 The extension group

For the extension group, we exploit two types of information to discover orthogonal dimensions for alternative views.

Impurity Score. We employ entropy impurity to measure this property of tags. Entropy impurity has been used in decision trees [1]. 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 3.2, and a tag t, we define $P_t(t_i)$ as the fraction of objects for $t_i \in T_b$ which is $P_t(t_i) = \frac{\sum_{(t_i,o) \in E', o \in O'_t} w(o)}{\sum_{o \in O'_t} w(o)}$. Then, the impurity of a candidate subtopic t given T_b is defined as $I(t) = -\sum_{t_i \in T_b} P_t(t_i) \log_2 P_t(t_i)$. 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. The sibling score of a candidate topic t measures how likely t is a sibling topic with the already chosen tags. We define $parent(t)$ as the the set of parent topics of t in the original taxonomy. The sibling function is defined as $Sibling(t_1, t_2) = \log_2(1 + |parent(t_1) \cap parent(t_2)|)$. 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.

$$SiblingScore(t) = \begin{cases} \frac{\sum_{t' \in T_c} Sibling_{norm}(t,t')}{|T_c|} & \text{if } |T_c| > 0 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

We still use the greedy approximation algorithm for set covering problem to generate the extension groups, only with a different objective function:

$$f_e(t) = \gamma \cdot CoverScore(t) + \delta \cdot SiblingScore(t) + \eta \cdot I(t) \quad (3)$$

Weights γ, δ, η will be tuned in the experiments. After generating one group of extension topics, another extension group can be generated by removing the selected topics from the candidates, and running the above process again.

3.4 The shortcut group

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. Among a variety of possible metrics to assess the popularity of a topic, we choose to use the raw frequency of the topic name being used as a tag in delicious. The popularity score is defined as:

$$popularity(t) = \frac{|bookmark(t)|}{\max_{t \in T} |bookmark(t)|} \quad (4)$$

where $|bookmark(t)|$ is the number of distinct bookmarks which are associated with tag t in Delicious. We again use Algorithm 1 but with a different objective function to generate the shortcut group: $f_s(t) = popularity(t)$.

4. EXPERIMENTS

4.1 Datasets and parameter tuning

We used two subsets of ODP to experiment with our approach. One is "Top/Computers/Computer_Science/" and all its subcategories and objects, containing 184 unique tags and 2061 URLs. The other contains "Top/Science/" and all its subcategories and objects, with 9415 tags and 100,109 URLs. Here, we only report important results because of the length limit. For detailed results and analysis, please refer to our technical report [7].

We tune the parameters in our method before generating the final taxonomy. We changed the value of γ, δ, and η by 0.1 in each step while keeping them sum to one. We generated a taxonomy for each parameter setting, and asked users to conduct pairwise comparison of the subtopics under one particular category at a time, and recorded their preferences. 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. Although there are significant differences in user ratings among different parameter settings, the result did not show any clear pattern. The best score is achieved when $\gamma = 0.8$, $\delta = 0.1$, and $\eta = 0.1$.

4.2 Taxonomy comparison and analysis

After parameter tuning, we applied our algorithm on the two subsets of ODP using the best parameter setting. Figure 2 illustrates the hierarchy generated by our algorithm for "Top/Computers/Computer_Science". Thus qualitatively we can see that the automatically generated hierarchies are reasonable. For comparison, we implemented the approach proposed by Heymann and Garcia-Molina [3], which will be referred to as "Communal Taxonomies". The algorithm output depends on a tag similarity threshold. Even after tuning, the generated hierarchy appears to be unreasonable. It has 183 categories and 3 levels, with 25.9 subcategories per category on average.

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. [5]. According to our implementation, 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".

4.3 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 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). This random extension is used

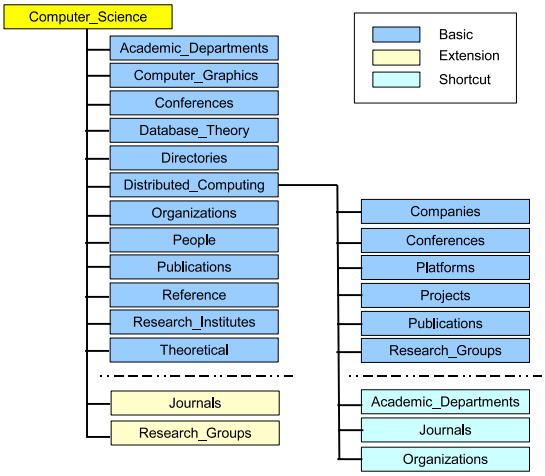

Figure 2: Automatically generated subtree under Top/Computers/Computer_Science.

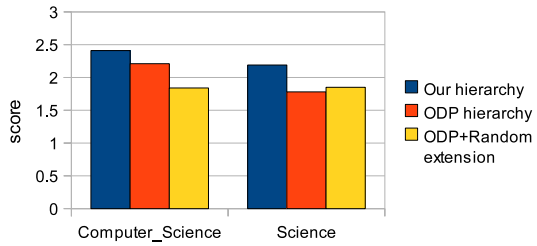

Figure 3: Results of user study on quality of hierarchies.

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).

Figure 3 shows the average score for each of the hierarchies, showing that our method outperforms the other two. In the "Top/Science" dataset, the satisfaction score judged by users is improved by 23% compared with the original ODP. Similar results can be found on the "Top/Computers/Computer_Science" dataset.

The second user study asks the users to perform specific information seeking tasks. We designed five tasks:

- Find software related links.
- Find fields which contain research groups in this web site.
- Find journal related links.
- Find theoretical publications.
- Find conferences in 2008.

Three users participated in this experiment. Figures 4(a) and (b) show the number of clicks and the elapsed time of users for each task, respectively, indicating that our hierarchy can reduce users' effort in finding information. 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.

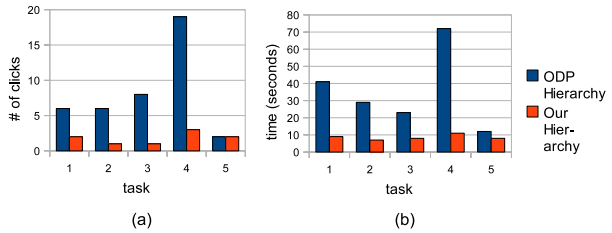

Figure 4: Average number of click counts and time for users to find designed items.

5. CONCLUSION

This paper described 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.

Acknowledgments

We thank Liangjie Hong for helpful discussions, and those who participated in our user study for their valuable input. This material is based upon work supported by the National Science Foundation under Grant Number IIS-0545875.

6. REFERENCES

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

[2] P. Clough, H. Joho, and M. Sanderson. Automatically organising images using concept hierarchies. In *SIGIR Workshop on Multimedia Information Retr.*, 2005.

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

[4] 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.

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

[6] 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.

[7] X. Qi, D. Yin, Z. Xue, and B. D. Davison. Enhancing taxonomies by providing many paths. Technical Report LU-CSE-10-005, Dept. of Computer Science and Engineering, Lehigh University, 2010.

[8] 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.

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