Construction of Online Catalog Topologies Using Decision Trees*

David Yang[†‡] Wing-kin Sung[§] Siu-Ming Yiu[†] David Cheung[§] Wai-Shing Ho[†]
Tak-Wah Lam[†] Sau-Dan Lee[§]

Abstract

Organization of a web site is important to help users get the most out of the site. A good web site should help visitors find the information they want easily. Visitors typically find information by searching for selected terms of interest or by following links from one web page to another. The first approach is more useful if the visitor knows exactly what he is seeking, while the second approach is useful when the visitor has less of a preconceived notion about what he wants. The organization of a website is especially important in the latter case. Traditionally web site organization is done by hand. In this paper, we introduce the problem of automatic web site construction and propose a solution for solving a major step of the problem based on decision tree algorithms. The solution is found to be useful in automatic construction of product catalogs.

Keywords: Web Organization, Online Catalog, Decision Tree, Metrics, Tree Optimization.

1 Introduction

As more and more companies either start on the web or join the web, there are an increasing number of choices for any product or service that may be desired. Several aspects of the site affect the choice made by potential customers. While the content provided by the site should be carefully selected, the ease and speed of finding the information they want are also considered to be important. The organization of the web site is thus an important subject for study. Ease and speed of access can be improved through the provision of search engines and indices, but the way in which the web pages are linked together is equally important.

Search engines and indices are typically considered more useful for those who know what they are seeking, while the links on a page are considered more useful for those who are less certain. Note that in practice, knowing what you are seeking does not guarantee a search engine or an index will provide satisfactory results. The vocabulary used by the site visitor must match that used by the site. Searches can be ineffective because the site and the visitors may use similar, but different, terms, resulting in no matches. Search results can be overwhelming if the visitor uses general terms or terms with multiple meanings.

At present, the organization of a website is still largely handled as something of an art and is typically done by hand. Rosenfeld and Morville [7] discuss many of these issues that come from different fields of study. However, very little has been discussed in automating the process of web site construction. The only previous work that has been done is to restructure and supplement but not create the organization. Garofalakis et al. [1] reorganize an existing web site by swapping children and parent pages when the child page is more popular. Perkowitz and Etzioni [5] automatically construct index pages that supplement an existing organization. Green [2] adds some cross links between pages that are likely to be related. In this paper, we automate the creation of the organization for the use of online catalogs. The objective of the automation process is to produce a web site organization such that visitors can locate popular information easily. While consultation with users is of course essential, we suggest here that it is possible to do this in conjunction with automatic methods. In particular, we propose one solution using decision trees. The solution can be shown to be useful in organizing online catalogs. In the next section, this automation problem will be presented. Section 3 will discuss our proposed solution which attempts to solve part of the problem. Discussion and conclusion will be presented in Section 4.

*Part of this work was developed at the IBM Centre for Advanced Studies in Toronto.

[†]Department of Computer Science and Information Systems, University of Hong Kong, Hong Kong; {dyang, smyiu, wsho, twlam}@csis.hku.hk.

[‡]on leave from the Department of Mathematics and Computer Science, St. Joseph's University, Philadelphia; yang@sju.edu.

[§]E-business Technology Institute, University of Hong Kong, Hong Kong; {wksung, dcheung, sdlee}@eti.hku.hk.

2 Problem Description

Generally speaking, the process for web site construction consists of the following steps, though the steps may be mixed together.

- Choose the contents of the web site
- Create the pages for the web site
- Decide on the organization of the web site
- Test the web site to make sure it satisfies not only the original specification, but the needs and desires of the site visitors
- Refine the design

2.1 Automating site organization

We now consider how these steps may be susceptible to automation, at least in part. The owner of the web site generally chooses the content. If the content is based on other sites, as with a search engine like Alta Vista, agents can automatically scan for content [3, 4].

Designing the organization for a website is still mostly done by hand. A number of things can be automated here, as well. This includes creation of certain types of pages as well as automatic linking of pages. The pages that can be created automatically will be supplemental in nature, assisting in directing site visitors to the content that interests them. The pages may be navigational (including index pages) that provide static links to other pages or search pages that will dynamically produce pages of links based on input from the user. These pages may contain data, but this will generally be metadata that describes the content. This scenario is particularly relevant to the creation of online catalogs. In such a situation, some pages are primarily intended to guide the visitor to the pages that contain details of a specific item. In this paper, we propose a solution which tries to automate the creation of these navigational pages.

For the last two steps, automation is also possible especially for the last step. Researchers have suggested a number of methods to reorganize the web site structures and/or adding additional index pages or links between existing web pages [1, 2, 5].

The rest of this paper will focus on the automation of the step deciding the organization of the web site, in particular, the online catalog, the part of the site which allows site visitors to navigate through the list of products or other items the site offers.

2.2 Model for online catalog design

Before we address the issue of helping automate the organization of an online catalog, we need to have a general conception of what a good online catalog should look like.

We note that when users navigate through an online catalog, they essentially traverse a tree to find the product that interests them. The inner nodes of the tree correspond to the navigational pages. By navigational, we mean that the primary purpose of these pages is to guide the user towards the appropriate information. The information on a navigational page generally assists in this purpose, as opposed to describing any specific product. The leaves of the tree correspond to the product information pages. These pages may contain links as well, but any links that lead to product information tend to be cross-links to related products. As products may fit in more than one category, product information pages may appear at more than one location in the structure, but always along the bottom of the tree.

We want this style of organization to be supportive of easy access to product information, especially popular pages. By easy, we mean that users will not miss the information, nor will they need to spend a lot of time to find it. The popularity of an existing page can be measured by examining site logs or page counters. For the case of a page for an existing product, the popularity of the page can be estimated using the past sales history of the product.

To take advantage of the tree-like structure to provide easy access, we suggest that each inner node of the tree is based on one attribute. [1] Each link to children nodes is based on one value, or a range of values, for the attribute. The decision of which link to follow then corresponds to answering one simple question: "Which value of attribute X interests you the most?" A concrete example would be the question, "Are you interested in men's clothing or women's clothing?" We note that this is similar to the way traditional store directories and catalogs are organized.

Basing each navigational page on one attribute resembles the use of one attribute at each inner node of a decision tree. In this sense, construction of online catalogs bears a resemblance to classification. Traditional classification forms a topology of several known classes of items. Our model of an online catalog is essentially a topology of the set of products.

Our approach does assume that the attributes for the information pages are provided. We have several

[1] Many online catalogs compress two levels of an index onto one page—our approach is compatible with this.

justifications for this assumption. Many companies already use a database to store product information. Furthermore, XML seems destined to overtake HTML as the language of choice for web data. XML supports easy extraction of attributes from a page. There is already great support in the marketplace for XML (e.g., *http://www.xmledi.org http://www.biztalk.org*).

Of course, using XML does not guarantee that the data will be highly structured. There are also a large number of pages that have already been created using HTML. For existing web pages, we note that Sahuguet and Azavant [8] have already presented a wrapper-based solution for extracting complicated structure from HTML documents to convert them to XML.

3 Our proposed solution

3.1 Metrics

The goodness of a catalog organization or any web site is quite subjective. This section will describe some objective criteria to measure the goodness of the catalog organization.

We begin by suggesting some guidelines for a good catalog organization. First, the catalog organization should not include orphaned pages. In the worst case, orphaned pages include pages that cannot be reached at all by following links from the main page. Even when this does not happen, there are often pages that can *only* be reached via a non-intuitive path, such as through a cross-link from another data page. Also, the catalog organization should try to avoid links from a navigational page to a data page that is totally unrelated or more general.

Second, for a good catalog organization, the depth of a frequently visited page should be small. Otherwise, many users will be forced to visit many navigational pages before they can find their target page. Loading the additional pages would also put a much heavier load on the web server.

Third, we should try to use attributes that are useful to the average users. In other words, some technical attributes like the model number of the chipset of the motherboard of an PC should be avoided if possible.

The use of decision tree addresses the first issue. Note that every inner node corresponds to a subset of data pages (that is, the set of data pages in the subtree rooted at this inner node). Since we input attribute values for every product, the root of the decision tree corresponds to all products, and no product page will be excluded by the tree. For every inner node v, every child of v corresponds to the subset of products which has the corresponding value for the partitioning attribute, so child nodes will always be more specific than their parents.

To handle the second issue, we can try to minimize the average height of the decision tree. Since the depth of a general website organization does not directly relate to the quality of the organization, we would like to reduce the height of the tree while maintaining the clarity of the organization.

We do recognize that some attributes may generally be less clear than others. We deal with this issue by introducing a weight for each attribute. The weight is used to model the clarity of the attribute. For example, we may want to give a large weight to the model number of the chipset as this attribute is too technical and not easy to understand. Burdening the user with having to specify an exact weight for each attribute is not desirable, however. From preliminary experimental experience, we find that the resulting organization is sensitive to the relative weights. We suggest that the user only needs to mark some attributes as "better" and some as "worse". The limit of three choices (with one being the default, "neutral") should allow the user some input without forcing a particular organization.

Based on the above discussion, we suggest two possible metrics for comparing different web site organizations based on decision trees. One metric is the average weighted depth of the tree, where each inner node on the path from the root to a page is weighted by the weight of the attribute used to partition the pages at that node. For a decision tree T with a set of data pages P, the average weighted depth is $f_1(T) =$

$$\frac{\sum_{p \in P} pop_p \sum_{n \in N_p} w_n}{\sum_{p \in P} pop_p}$$

where pop_p is the popularity of page p, N_p is the set of inner nodes on the path from the root to page p, and w_n is the weight of the partitioning attribute at node n.

The above metric may favor an organization which includes some relatively unclear attributes. This may happen if the paths containing such attributes are relatively short. If we are willing to allow for longer paths that contain fewer unclear attributes, we can increase the effect of attributes with large weights by using the average maximum $f_2(T) =$

$$\frac{\sum_{p \in P} pop_p \max_{n \in N_p} w_n}{\sum_{p \in P} pop_p}$$

In this paper, we model the automatic catalog organization problem as follows. Figure 2 demonstrates an example.

An algorithm which restructures a catalog topology according to a metric f

1. Build a decision tree T which consists of all the data pages;
2. Let r be the root of T;
3. R = Improve(T, r, f);
4. Return R;

Improve(T, v, f)

1. For every attribute a, let T_a = Rebuild(T, v, a);
2. Among all the T_a, let R be the T_a such that $f(T_a)$ is minimized;
3. For every child u of v, let R = Improve(R, u, f);
4. Return R;

Figure 1: Algorithm for automatic catalog organization

Input:
1. A set of attributes where each attribute has a weight to describe its clarity.
2. A set of data pages where each page is defined by setting particular values to the attributes.
3. Each data pages has a popularity measurement.
4. A metric $f()$ which measures the goodness of the catalog organization.

Output: A decision tree T with a small $f(T)$.

3.2 Algorithms

This section presents an algorithm which offers a solution to the automatic catalog organization problem. The algorithm is quite general in the sense that it can be used to construct a "good" tree for any metrics. Section 3.2.1 describes the details of the algorithm. Section 3.2.2 demonstrates our algorithm based on the average weighted depth metric and the average maximum metric.

3.2.1 Details of the algorithm

Our algorithm can be divided into two steps. The first step generates an initial decision tree T for the data pages. The next step of the algorithm is to restructure this tree so as to improve the value of $f(T)$, where f is a metric.

We could build a tree by using some classical decision tree construction algorithm like C4.5 [6], but the situation is different since each catalog item corresponds to a single class. There are thus no issues concerning misclassification.

We instead try to target the construction of a tree that favors a good tree according to the chosen metric. We start building the initial decision tree by arbitrarily choosing the partitioning attribute at the root and partitioning the set of pages by using this attribute. For each child with more than one data page, we arbitrarily pick another attribute and continue until every remaining node contains a single data page or all attributes have been used. Figure 3 shows a decision tree which is based on the home equipment database in Table 1.

After getting the decision tree T from the first step, the second step tries to restructure T to reduce the value of the metric $f(T)$. The basic routine for restructuring a decision tree is Rebuild(T, v, a). This routine tries to change the attribute of inner node v of T to a. Note that such modification affects the structure of the subtree rooted at v. Therefore, as a side effect, Rebuild(T, v, a) will also restructure the whole subtree of T rooted at v. The details of the procedure Rebuild(T, v, a) are as follows. Figure 4 shows an example to demonstrate the restructuring.

1. Change the attribute in node v of T to a.
2. Partition the data pages in the leaves of the subtree of T rooted at v into groups $G_1, G_2, \ldots, G_k$ so that within each group, the values of the attribute a of the data pages are the same.
3. For each group G_i, let S_i be the subtree of T such that
   (a) all the data pages not in G_i are removed; and
   (b) all the inner nodes which have one child are removed.
4. Replace all the subtrees which are attached to v by $S_1, S_2, \ldots, S_k$.

Given the Rebuild routine, we can improve the decision tree T by updating the attribute used in each node. In Figure 1, we suggest an algorithm which produces this improvement by updating the attributes of all nodes in T in a top-down order.

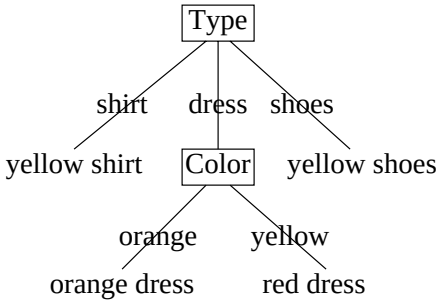
3.2.2 Examples

This section shows an example to demonstrate how the popularity of the data pages and the weights of the attributes affect our procedure to organize the web.

w eigh t	1	1	5
	color	**type**	**product ID**
yellow shirt	*yellow*	*shirt*	*001*
orange dress	*orange*	*dress*	*002*
yellow shoes	*yellow*	*sho es*	*003*
red dress	*red*	*dress*	*004*

popularity
100
20
5
5

(a)



(b)

Figure 2: An example: (a) is the input and (b) is the output tree T. Note that $f_1(T) = 1.19$ and $f_2(T) = 1$.

w eigh t	1	2	3
	type	**size**	**shape**
radio	*electric*	*small*	*rectangle*
clock	*electric*	*small*	*circle*
air-conditioner	*electric*	*large*	*rectangle*
tv	*electric*	*medium*	*rectangle*
toaster	*kitchen*	*small*	*rectangle*
refrigerator	*kitchen*	*large*	*rectangle*
microwave	*kitchen*	*medium*	*rectangle*
chair	*furniture*	*small*	*circle*
table	*furniture*	*large*	*square*

popularity
5
500
10
5
50
10
5
200
1000

Table 1: The home equipment database

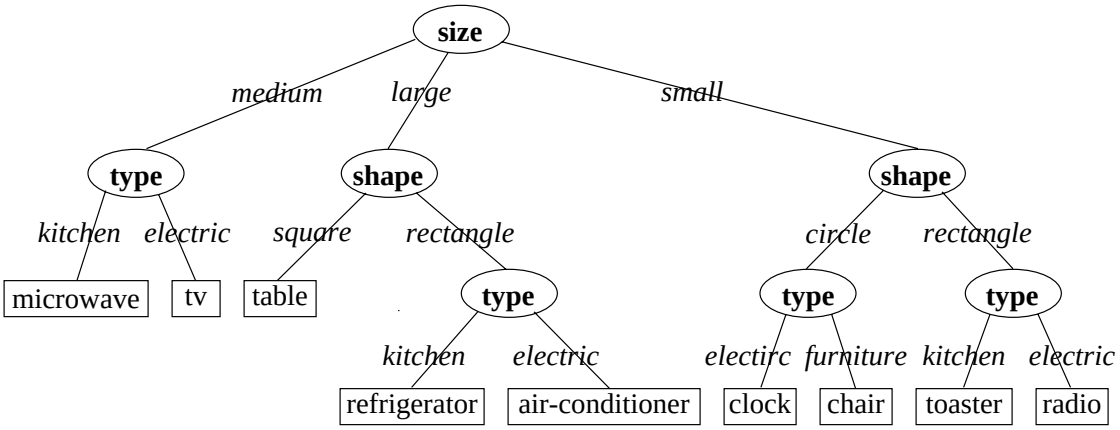


Figure 3: An initial decision tree of the home equipment database (average weighteddepth = 5.42, average maximum = 2.99)

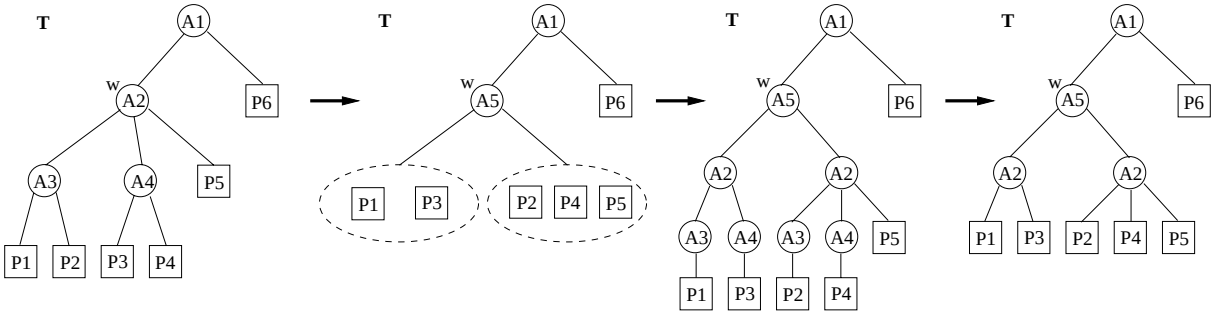


Figure 4: An example which demonstrates Rebuild(T, w, A_5)

Consider the home equipment database in Table 1 again. Based on our catalog organization routine, we get the decision tree in Figure 5 if we try to optimize the average weighted depth. On the other hand, if we try to optimize the average maximum, we get the decision tree in Figure 6.

For Figure 5, observe that the popular products like table, chair and clock are placed near the top of the tree. This is due to the fact that the routine tries to reduce the average weighted depth. Hence, the routine tries to push the popular pages to the top of the decision tree.

The decision tree in Figure 6 is built based on the average maximum metric. Note that apart from the two paths from the root to radio and clock, none of the paths contain the attribute shape (which has a weight 3). The reason is that the average maximum metric tries to reduce the number of paths which have attributes with large weights.

4 Discussion and Conclusion

This paper introduces the problem of automatic catalog organization and proposes a decision tree-based solution for solving a major step of the problem. Our solution is quite general in the sense that it allows the designer to decide what is a good catalog organization by defining the metric. We have presented two simple metrics.

The current algorithm is intended to produce a correct and efficient organization, but may of course be modified further by users. The Rebuild() function allows such reorganization to be done without creating orphaned pages. The algorithm can process a large database of products, but not in real-time. We are interested in finding faster alternatives that still produce an efficient organization.

We will need to address a number of important details regarding our approach. We are studying the impact of the attribute weight and product popularity on the resulting catalog so that attribute weights are not assigned arbitrarily. We also want to study the impact of the initial tree on the results and develop a better way to create the initial tree.

While multi-valued attributes (e.g., the color of a striped shirt) can be processed by our current algorithm, the impact of these attributes on the metrics has not been fully analyzed. More difficult issues to handle include how the partitioning of pages should handle a page that does not have a value for the partitioning attribute (e.g., the gender of a blender). A related problem is how to handle users who do not care about or do not understand an attribute used on a navigational page. A practical implementation of our approach should also consider attributes with a large domain. The standard solution of using ranges of values or subsets of the domain may not always be satisfactory. A consumer is more likely to be interested in a set of choices like, "less than $20", "less than $40", and "any price" instead of "less than $20", "$20–$40", and "over $40".

More importantly, this paper only considers one step of the process. We are interested in other steps as well. For instance, this paper assumes that the attributes and their values for each page are available. We are interested in exploring techniques to extract this information. This would facilitate use of our approach with existing data. We are also interested in improving our heuristic and more thoroughly evaluating results when using different metrics.

References

[1] John Garofalakis, Panagiotis Kappos, and Dimitris Mourloukos. Web site optimization using page popularity. *IEEE Internet Computing*, 3(4):22–29, 1999.

[2] Stephen J Green. Auomated link generation: can we do better than term repetition? *Computer Networks and ISDN Systems*, 30(1–7):75–87, 1998.

[3] B. Kao, J. Lee, D.W. Cheung, and C.Y. Ng. Recommending anchor points in structure-preserving hypertext document retrieval. In *Proceeding of Twenty-Second Annual International Computer Software and Application Conference*, pages 582–587, 1998.

[4] J. Lee, D. W. Cheung, B. Kao, J. Law, and T. Lee. Intelligent agents for matching information providers and consumers on the world-wide-web. In *Proceeding of Thirtith Hawaii International Conference on System Scienee*, pages 189–199, 1997.

[5] Mike Perkowitz and Oren Etzioni. Adaptive web sites: automatically synthesizing web pages. In *AAAI*, pages 727–732, 1998.

[6] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufman, San Mateo, CA, 1993.

[7] Louis Rosenfield and Robert Morville. *Information architecture for the world wide web*. O'Reilly, Cambridge, 1998.

[8] Arnaud Sahuguet and Fabien Azavant. Building light-weight wrappers forlegacy web data-sources using w4f. In *VLDB*, pages 730–733, 1999.

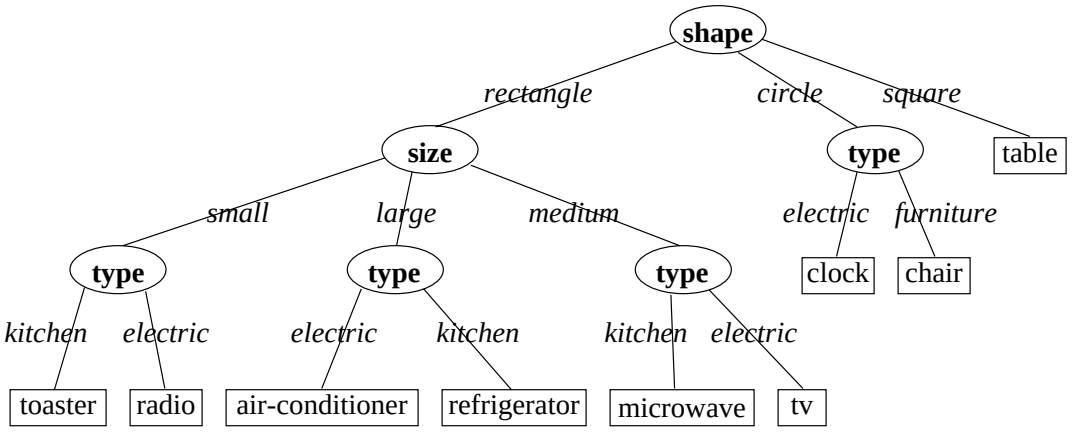


Figure 5: Decision tree obtained based on average weighted depth (avg. wtd. depth = 3.54)

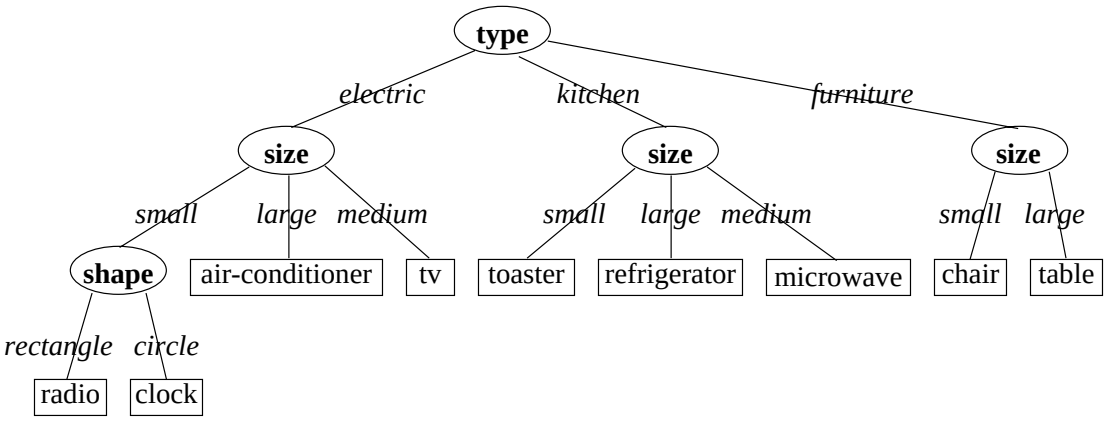


Figure 6: Decision tree obtained based on average maximum (avg. max. = 2.28)