行政院國家科學委員會專題研究計畫 成果報告

彈性化之數位典藏瀏覽系統建置(Ⅱ)
研究成果報告(完整版)

計 畫 類 別 ：個別型
計 畫 編 號 ：NSC 95-2422-H-002-017-
執 行 期 間 ：95 年 03 月 01 日至 96 年 04 月 30 日
執 行 單 位 ：國立臺灣大學資訊管理學系暨研究所

計 畫 主 持 人：莊裕澤
共 同 主 持 人：吳玲玲
計畫參與人員：博士班研究生-兼任助理：莊雅嵐
碩士班研究生-兼任助理：陳宇翔、詹子儀

處 理 方 式 ：本計畫可公開查詢

中 華 民 國 96 年 09 月 10 日

An Ad-hoc Browsing System for National Digital Archives Program—Continuation

Yuh-Jzer Joung
Department of Information Management
National Taiwan University
Taipei, Taiwan
joung@ntu.edu.tw

1 Introduction

"National Digital Archives Program" (NDAP) was launched in 2002 to digitize our national cultural treasures and heritage so that they can be preserved and utilized in the digital era. The information system developed for NDAP is not only open to experts and researchers, but also to everyone who wishes to explore the beauty of our culture. The project was motivated by the need of a flexible and user-friendly interface for browsing such a large archives system.

We use *multi-faceted categorization* [2, 6, 3] to support a flexible browsing path that may vary dynamically depending on the user's goal, and on the information he has observed and recalled during the exploration process. Multi-faceted categorization adopts a flat structure of categories, treating them as equal and independent. This type of flat organization allows users to conduct *multi-dimensional browsing*, or *ad-hoc browsing*, where users can freely move from any category to any other category as their information needs change along the process of information seeking. The browsing interface with multi-faceted categorization can thus provide an environment for dynamic, flexible and spontaneous information seeking.

Although multi-faceted categorization opens up myriad possible browsing paths and confers great flexibility during browsing, it may offer more freedom than most web users can handle, given the limited capacity of human information processing [4]. In the first place, users need to choose which category to start browsing. Then, users need to do the choice every single time when they want to move to another category. In addition, the users might need to navigate all categories when choosing which one to browse. All these tasks come along with the benefit of flexibility inherited in multi-faceted categorization. Therefore, the large degree of freedom in this kind of browsing interface entails a call for balance between flexibility and limited capacity of human information processing.

From the user-centered perspective of interface design, a meaningful organization of the flat structure may facilitate more effective navigation for users performing multi-faceted browsing, especially when the organization is meaningful to the users. For instance, user-defined relevance mechanisms could be incorporated into the browsing interface to reflect which categories are

perceived by users as more important, and which are deemed as less important. By doing so, the multi-faceted categorization can still remain flat to support multi-dimensional browsing, which allows space for users to perform navigation in any way they need. At the same time, this browsing interface can be easier for users to navigate because this organization matches users' cognitive structure of the information at hand. The purpose of our research was to develop a user-centered interface of multi-faceted categorization, thereby to strike a balance between the need for flexibility in browsing and the limits in human information processing capacities.

Scientific literature has clearly documented the profound impacts of context on many aspects of human information processing. In particular, context has been shown to play an essential role in relevance evaluation of category attributes [1, 5]. Therefore, a user-centered interface for multi-faceted browsing system should reflect this contextual relevance of categories. Based on this idea, we develop a context-sensitive interface for multi-faceted categorization, and design an experiment to investigate its browsing effectiveness for users. Two browsing interfaces were constructed for experimental assessment: context-sensitive and context-insensitive interfaces. The context-sensitive interface, called **COAD (Contextually-Ordered Attribute Display)**, organizes attributes dynamically according to the context. This browsing interface was similar to the interface used in Flamenco [6], but embedded contextual organization in the facet display. For comparison, we used a baseline interface, called **ROAD (Randomly-Ordered Attribute Display)**, which simply displayed facets in some arbitrarily-determined order. Both interfaces employed multi-dimensional categorization techniques.

The browsing effectiveness was measured in terms of user perceptions: ease-of-information-access, confidence in decision making, ease-of-browse, and user satisfaction. Four hypotheses, regarding the browsing effectiveness of the context-sensitive interface, were proposed:

Hypothesis 1: The average rating for ease-of-access to information for COAD users will be higher than for ROAD users.

Hypothesis 2: The level of confidence in the quality of decision-making will be higher for COAD users than for ROAD users.

Hypothesis 3: The average rating for ease-of-browsing will be higher for COAD users than for ROAD users.

Hypothesis 4: The overall level of satisfaction of COAD users will be higher than that of ROAD users.

The empirical results of the experiment, as summarized below, suggest that the context-sensitive arrangement of categories allows users to find information more easily and that users perceive this interface to be easier to use.

Browse Evaluation for each Context Scenario: Ease-of-information-access and confidence in the decision-making

The empirical result indicates that evaluations for both context scenario questions—ease-of-information-access and confidence in decision-making quality—are significantly higher for the COAD interface than for the ROAD interface. Student's t-test

is performed on the effect of the interface for both of the questions. The result shows that subjects using the context-sensitive interface (COAD) found it significantly easier to find the information they needed than those using the context-insensitive interface (ROAD) (5.37 vs. 5.03; $t(37) = 3.28$, $\hat{\sigma}_{ij} = .10$, $p < .001$). The results also demonstrate that subjects using the context-sensitive interface felt more confident about their car purchase decisions than those using the context-insensitive interface (5.32 vs. 5.10; $t(37) = 2.16$, $\hat{\sigma}_{ij} = .11$, $p < .019$). These empirical results strongly support Hypotheses 1 and 2.

Overall Browse Evaluation: Ease-of-browse and User satisfaction

For the overall browse evaluation, we used the ANOVA procedure to determine whether or not the interface design had a significant effect on the perceptions of ease-of-browse and overall user satisfaction. The results show that the interface has significant effect on ease-of-browse (5.45 for COAD vs. 5.05 for ROAD; $F(1, 71) = 9.81$, $MSe = 0.30$, $p < .003$). Subjects do think that it is easier to browse when using the context-sensitive interface, COAD. However, the effect of the interface is not significant in terms of overall satisfaction (5.02 vs. 4.72; $F(1, 71) = 1.37$, $MSe = 1.17$, $p < .251$). In other words, Hypotheses 1, 2, 3 are supported, but Hypothesis 4 is not supported by the empirical results.

The empirical results strongly support our hypotheses about the merits of context-sensitive browsing interfaces. The above results show that COAD, the context-sensitive interface, is consistently rated as superior to ROAD, the context-insensitive interface, for browsing. Users find it easier to browse and to locate needed information when using the context-sensitive interface (COAD) and feel more confident about the quality of their decision making than when they use the context-insensitive interface (ROAD). The idea of adjusting the attribute order according to user information needs in different contexts does appear to make browsing more effective.

For more details of the empirical study, please refer to the following paper. Most of the results in the paper are done in the first year of the project (intended for a two-year term).

- Contextual Multi-Dimensional Browsing. Ling-ling Wu, Ya-lan Chuang, and Yuh-jzer Joung. Submitted to *Computers in Human Behavior*. (in 2nd round review)

In the above empirical results, we notice that Hypothesis 4 is not supported. This may be attributed to the slow response time of the system, which ranges from 6 to 10 seconds for each click. Note that the slow response time is not uncommon in the browsing interface of a multi-faceted categorization system. For example, the response time of the Flamenco interface (Hearst, 2006, http://flamenco.berkeley.edu/demos.html) we have measured also ranges between 5-10 seconds. (Their system does show the response time to generate the results.) The slow response time can be attributed to the fact that in each browsing step, the database engine needs to calculate, in addition to the number of items that have matched the current selection, the number of items that may match a particular attribute, should this attribute be

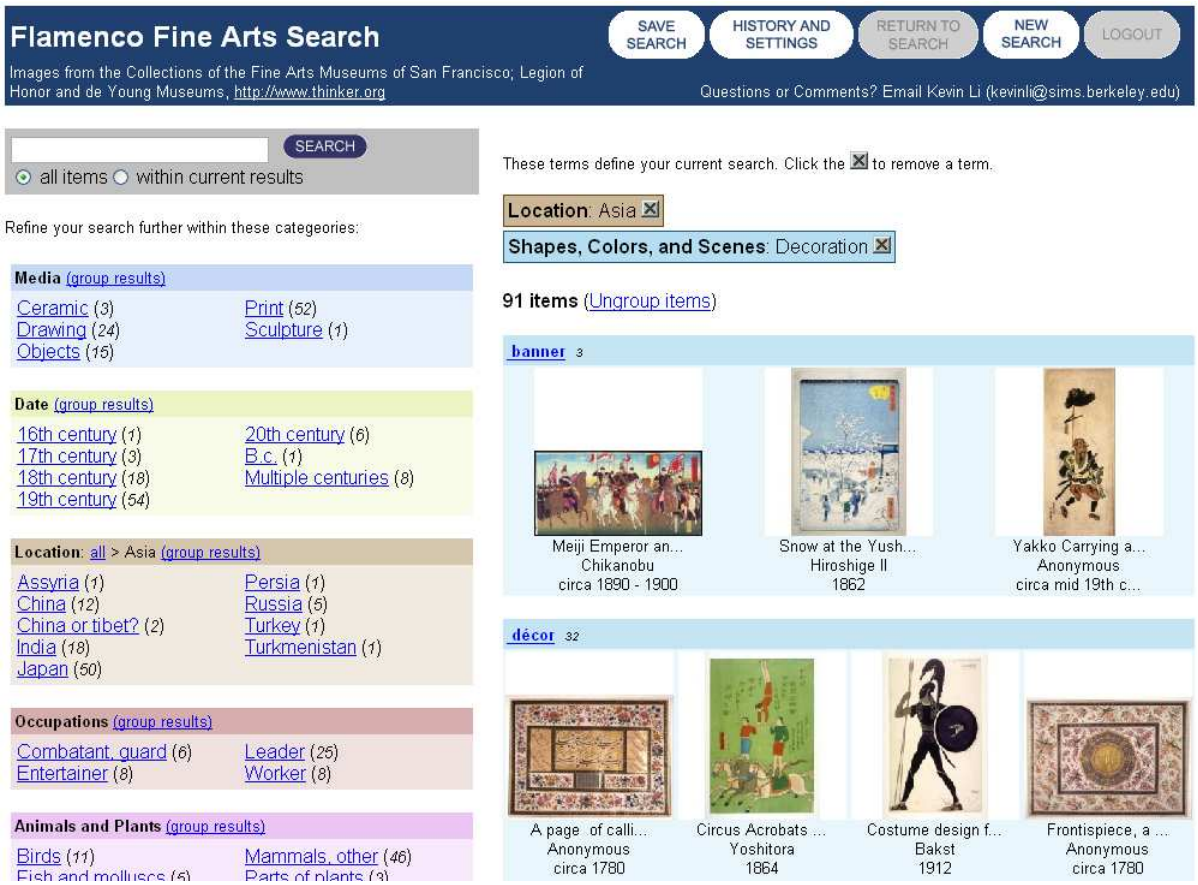

Figure 1: The Flamenco interface.

selected next. For example, in the Flamenco interface shown in Fig. 1, the number in each attribute in the left panel indicates the number of matching objects if this attribute is selected. Calculating this number is important as it allows us to remove unnecessary features that may lead to nowhere, as well as providing the user the status of the database in the browsing path. Unfortunately, this is also the most time-consuming step in the process, especially when the number of attributes is large. (For a straightforward implementation, the time complexity in updating the attribute information is in proportion to the number of attributes times the size of the database.) Thus some optimization must be incorporated to the interface design in the future for the system to scale.

The second year of the project, therefore, is to focus on the design and implementation of a mechanism to speed up the query processing. For this, we propose a hyper-based indexing scheme called **HyperFacet** as middleware to bridge the internal data layout of the information system and the external user browse interface. The overall system is shown in Fig. 2, where the component **HyperFacet** is added in between the database and the original system to provide an index scheme to speed up calculating the numbers of matched items in each facet.

To explain the idea behind the HyperFacet middleware, let us take a look at the system interface shown in Fig. 3. The content we used in this research is automobiles, where more than 8,030 cars, each with 63 attributes, have been collected in our database. Among the 63

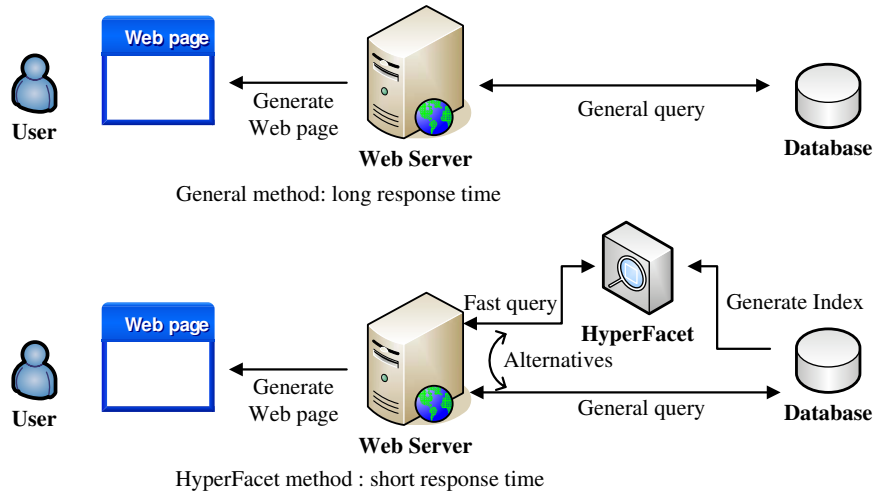

Figure 2: HyperFacet system overview. For comparison, the original system without the speedup middleware is shown on the top.

attributes, 44 were chosen to be our facets (e.g., brand, price, seat capacity, sunroof (Yes/No), horsepower, engine types, cargo volumes, and etc). The **facet** part shows the facets and its terms. The terms chosen by the user will be shown on the **constraint** part, and can be unselected by pressing the "X" button after it. The **Result** part shows the items that satisfy the current constraint, along with their total count.

A **facet** is an aspect of the content (e.g., make, year, or engine). Within each facet, items can be further classified into several disjoint subcategories according to different **terms** specified by the system designers (e.g., BMW, 2006, or 2000 2500 c.c.). Terms can also be viewed as constraints or attributes that a user can further select to narrow down his search. In our web page, each term is followed by a number, called **MI** (number of matching items), that tells you the new number of matching items *if* you add this constraint to the search. Terms that have zero value in this number means that no item in the constraint contains this attribute. To make the interface user-friendly, we give these **unavailable terms** a gray color and make them not selectable. Therefore, the main task of the HyperFacet is to speed up counting the number of matching items for a given constraint.

A straightforward way of implementing it is to use SQL: " **select** *field_id*, count(field_id), **from** *table_name*, **where** *constraints* **group by** *field_id*" for every facet. Such command will make the system traverse all the records in the database, determine which group each record should fall into, and count the sum of every group in the end. Hence, a system with 10,000 items in the database and 20 facets on the interface would have to do search within 10,000 records for 20 times! As a result, the system takes much longer time to return the results than traditional browsing systems, and the time is proportional to the number of dimensions and the scale of the database. So calculating the number of matching items is the bottleneck of multi-faceted browsing.

Our approach is to convert the important information of an item into a bit array according to

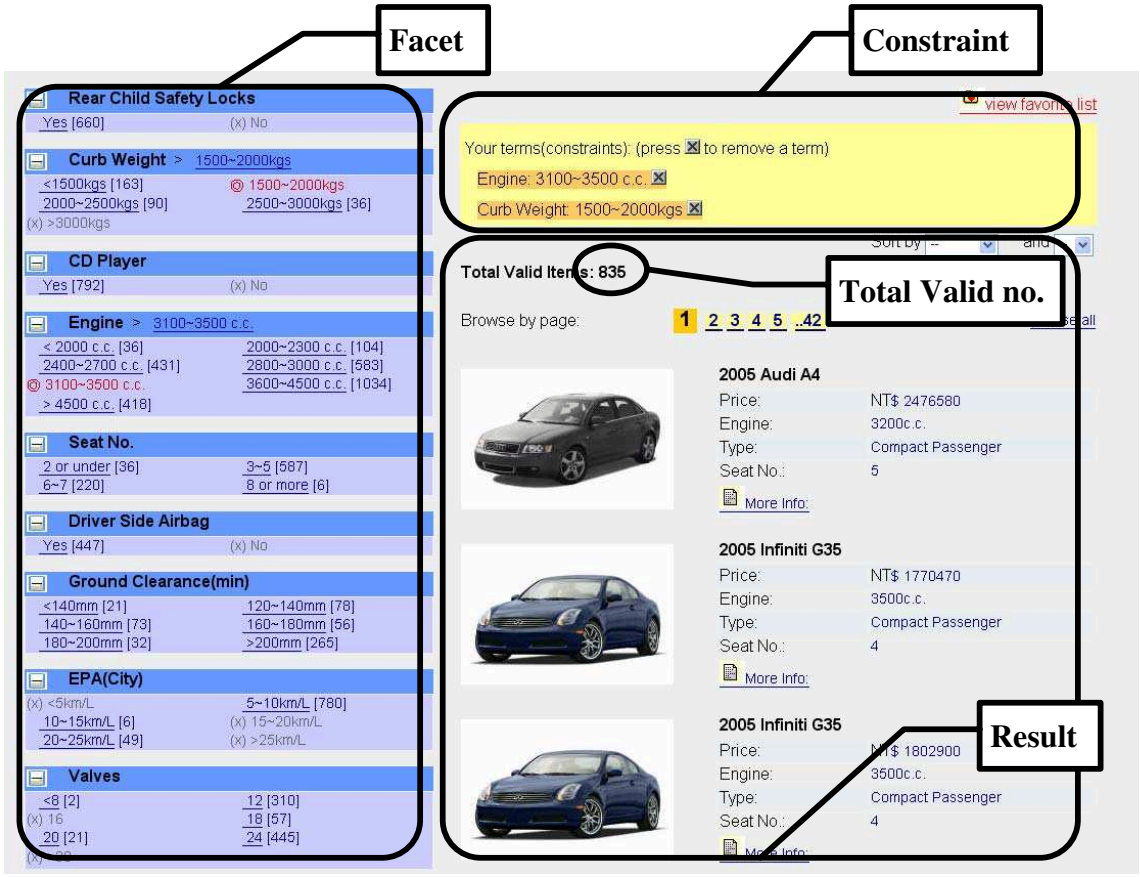

Figure 3: HyperFacet screenshot

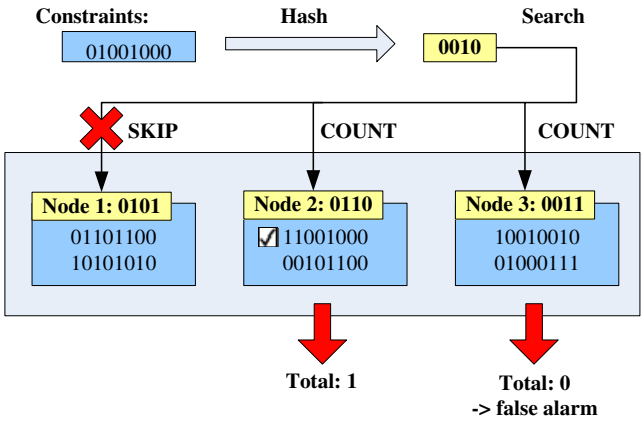

Figure 4: The filtering process.

its attributes. Every bit represents a unique attribute. If the item contains a certain attribute, the corresponding bit in the bit array should have '1' for its value. Therefore, two items with identical bit arrays if and only if they have exactly the same attributes. If we further transfer a user's constraints into the form of a bit array, it would be very simple to see if an item satisfies a constraint.

In order to do this, we need to first convert our data in the content table to bit arrays, which we called **attribute_id**. After that, we must find a way to manage and compare them with the bit array transferred from a user's constraint. A straightforward way is to store all the bit arrays into a vector, and check them one by one when calculating the MI numbers. This, however, is time-consuming, as the length of the vectors depends on the total number of possible attributes. Instead, a hashing technique is called to improve the performance. We give each item in the content table a **hash_id**, which can be viewed as a condensed version of attribute_id. The length of the hash_id is called the *dimension* of this system. This number will highly affect the performance of the system. We start the hash_id with all '0' bits in the array. For every attribute the item contains, we hash it to an integer h, where $0 \leq h \leq dimension - 1$. Then we turn the h^{th} bit of the hash_id to '1'. In such way, items containing a certain attribute Ai will all have '1' in the i^{th} bit of their hash_ids, where $i = hash(Ai)$. In another word, items with hash_ids that do not have '1' in their i^{th} bit absolutely will not contain attribute Ai.

Hence, we can filter items by doing a binary operation to their hash_ids in advance. We group items that have the same hash_ids together in a node. When calculating MI numbers, the system first checks the hash_ids of the nodes. If the hash_id does not satisfy the hash_id transferred from the constraints, all items in that node need not be traversed. This process is illustrated in Fig. 4. The user's constraints are hashed to a hash_id first. Then the system compares it with the hash_ids of each node. Node 1 does not satisfy the hash of the constraint and therefore is skipped.

Since hash_ids usually have much fewer bits than attribute_ids do, and the number of nodes is also fewer than the number of items, the filtering process reduces query time substantially with only little cost. This idea of managing attribute_ids with nodes also gives the system a better control over them. We can find an item faster and also do advanced queries such as "find

similar items". The system is able to count the number of different bits between the target and other attribute_ids. By retrieving the ones with differences under a certain threshold, it can complete the task of finding similar items, which could be very complicated when using SQL.

As a result, by loading the small sized information into the memory (hash_ids, attribute_ids, and the items' unique ids from the content table), our middleware will be able to query and calculate quickly with simple binary operations. If further queries are needed, the middleware could still query the database using the unique ids produced from earlier search. Overall, on an average, the HyperFacet middleware can reduce the response time in our database from 5 seconds to within 1.5 seconds!

For more detailed implementation of the HyperFacet middleware and the experimental results, please refer to the following thesis.

- HyperFacet: An Accelerating Middleware for Multi-faceted Browsing Interface. Yu-Hsiang Chen. Master Thesis. Dept. of Information Management, National Taiwan University. Sep. 2006.

We are working on a conference and journal version of the results.

References

[1] Lawrence W. Barsalou. Ideals, central tendency, and frequency of instantiation as determinants of graded structure in categories. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 11(4):629–654, October 1985.

[2] Marti Hearst, Ame Elliott, Jennifer English, Rashmi Sinha, Kirsten Swearingen, and Ka-Ping Yee. Finding the flow in web site search. *Commun. ACM*, 45(9):42–49, 2002.

[3] Marti A. Hearst. Clustering versus faceted categories for information exploration. *Commun. ACM*, 49(4):59–61, 2006.

[4] George A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psycological Review*, 63:81–97, 1956.

[5] S. Ratneshwar, Lawrence W. Barsalou, Cornelia Pechmann, and Melissa Moore. Goal-derived categories: The role of personal and situational goals in category representations. *Journal of Consumer Psychology*, 10(3):147–157, 2001.

[6] Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti Hearst. Faceted metadata for image search and browsing. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401–408, New York, NY, United States, April 2003. ACM Press.