Multi-Select Faceted Navigation Based on Minimum Description Length Principle

Chao He [†‡], **Xueqi Cheng** [†], **Jiafeng Guo** [†], **Huawei Shen** [†]
[†] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, P.R.China
[‡] Graduate University of the Chinese Academy of Sciences, Beijing, P.R.China
{hechao,guojiafeng,shenhuawei}@software.ict.ac.cn cxq@ict.ac.cn

Abstract

Faceted navigation can effectively reduce user efforts of reaching targeted resources in databases, by suggesting dynamic facet values for iterative query refinement. A key issue is minimizing the navigation cost in a user query session. Conventional navigation scheme assumes that at each step, users select only one suggested value to figure out resources containing it. To make faceted navigation more flexible and effective, this paper introduces a multi-select scheme where multiple suggested values can be selected at one step, and a selected value can be used to either retain or exclude the resources containing it. Previous algorithms for cost-driven value suggestion can hardly work well under our navigation scheme. Therefore, we propose to optimize the navigation cost using the Minimum Description Length principle, which can well balance the number of navigation steps and the number of suggested values per step under our new scheme. An emperical study demonstrates that our approach is more cost-saving and efficient than state-of-the-art approaches.

1 Introduction

A tremendous number of databases have been published online, like e-Commerce websites (e.g. Amazon and eBay) and digital libraries (e.g. ACM digital libraries). These databases, referred to as *Deep Web* [Bergman, 2001], are estimated to be orders of magnitude larger than traditional Web [He *et al.*, 2007]. Users often access them by filling a form to specify the attribute values of targeted resources. When a daunting list of resources is returned, however, it is hard for users to specify appropriate values to zoom in.

Recently, faceted navigation [Hearst, 2006; Sinha and Karger, 2005; Tunkelang, 2006; Dakka and Ipeirotis, 2008; Koren *et al.*, 2008; Lee *et al.*, 2009; van Zwol and Sigurbjornsson, 2010; Li *et al.*, 2010] has become an alternative of the form-filling approach. A *facet* represents a dimension for classifying resources. For example, {*science*, *horror*} and {$<$ \$50, \$50 ~ \$100, $>$ \$100} represent a theme and price facet of books, respectively. Users reach targeted resources by a series of navigation steps, named a *session*. At each step, a set of facet values are suggested for query refinement. User studies show that faceted navigation is effective in reducing user efforts of exploring and searching databases [Yee *et al.*, 2003; Marchionini, 2006].

Unfortunately, the navigation scheme of current faceted navigation systems is rather simple, which limits a further cost reduction. It assumes that only one suggested value can be selected for refinement at each step, even if multiple ones are qualified. This would lead to superfluous steps. Meanwhile, suggested values can only be used to retain resources containing them, while sometimes it is also important to exclude resources with certain values for fast navigation.

This paper thus proposes a new navigation scheme. It supports multiple selection on suggested values, and a selected value can be used to either retain or exclude the resources containing it. Hence a query result can be refined much smaller at a single step, and we can significantly save the navigation cost of browsing the same value at different steps in a session.

Then, the challenge is to suggest the set of facet values which minimize the expected navigation cost in a session. Suggestion algorithms in previous work [Basu Roy *et al.*, 2008; Kashyap *et al.*, 2010] are under the presumption of conventional navigation scheme, i.e. one click per step. In our navigation scheme, the presumption changes so that existing suggestion algorithms are no longer optimal and hence they can hardly keep good performance.

In this paper, we propose to adopt the Minimum Description Length (MDL) principle to reduce navigation cost under our navigation scheme. As we can see, a tradeoff is required between the number of suggested values at one step and the number of navigation steps in a session. MDL-based approaches have been found effective in balancing conciseness and preciseness for data summarization [Lee *et al.*, 2007; Navlakha *et al.*, 2008]. In our setting, the conciseness property of suggested values means to minimize user efforts at one step, while the preciseness property means to minimize the number of session steps. Hence our MDL-based algorithm can suggest the values which minimize the navigation cost of a session under our navigation scheme.

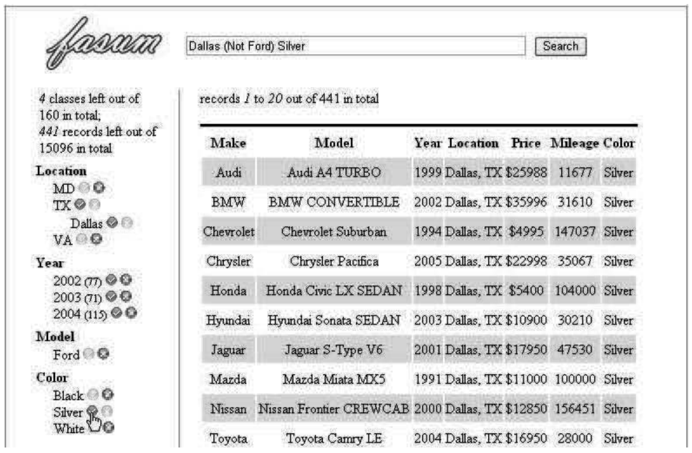
The remainder of this paper is organized as follows. Section 2 discusses the navigation model and cost model of our navigation scheme. In Section 3, we propose to suggest values based on the MDL principle. Section 4 presents a greedy algorithm for optimizing the MDL-based cost function. An empirical study is demonstrated in Section 5. Section 6 concludes our work.

2 Multi-Select Faceted Navigation

Figure 1 depicts the user interface of our multi-select faceted navigation system for searching used cars. It supports multiple selection on suggested values, and a selected value can be used to either retain or exclude the resources containing it. It has three major components, a top input box for query, a left-side panel for suggested values, and a right-side table for presenting a part of query result. Each suggested facet value is associated with two buttons, the left one for retaining the resources containing the value, i.e. *true*, while the right one for excluding them, i.e. *false*.

Here, users have already selected *Dalls*, *Ford* and *Silver* all in one step, which are set *true*, *false* and *true*, successively. In conventional navigation scheme, at least one additional step is required after the selection of *Dalls*, and hence users have to browse another set of suggested values. Besides, *Ford* cannot be used to filter out the resources containing it although *Ford* is the brand of many cars except for the targeted car.

To reduce user efforts and prevent dead ends, our navigation system automatically decide what values can be fixed during users' selection process. If a value is contained by each remaining resource, its left button becomes unselectable (turns to gray). If a value is not contained by any remaining resource, its right button becomes unselectable. When all suggested values are decided (either by user or system), a finest descriptor is formed, and users move on to the next step with a new set of suggested values. For example, *TX* is automatically set *true* since each remaining car contains it, while *MD*, *VA*, *Black* and *White* are automatically set *false* since no remaining car contains them.



Dallas (Not Ford) Silver

4 classes left out of 160 in total;
441 records left out of 15096 in total

Location
MD
TX
Dallas
VA
Year
2002 (77)
2003 (71)
2004 (115)
Model
Ford
Color
Black
Silver
White

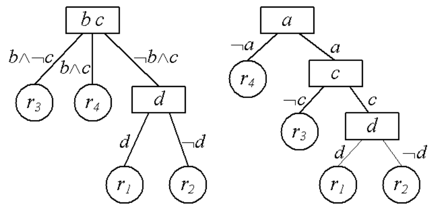
records *1* to *20* out of 441 in total

Make	Model	Year	Location	Price	Mileage	Color
Audi	Audi A4 TURBO	1999	Dallas, TX	$25988	11677	Silver
BMW	BMW CONVERTIBLE	2002	Dallas, TX	$35996	31610	Silver
Chevrolet	Chevrolet Suburban	1994	Dallas, TX	$4995	147037	Silver
Chrysler	Chrysler Pacifica	2005	Dallas, TX	$22998	35067	Silver
Honda	Honda Civic LX SEDAN	1998	Dallas, TX	$5400	104000	Silver
Hyundai	Hyundai Sonata SEDAN	2003	Dallas, TX	$10900	30210	Silver
Jaguar	Jaguar S-Type V6	2001	Dallas, TX	$17950	47530	Silver
Mazda	Mazda Miata MX5	1991	Dallas, TX	$11000	100000	Silver
Nissan	Nissan Frontier CREWCAB	2000	Dallas, TX	$12850	156451	Silver
Toyota	Toyota Camry LE	2004	Dallas, TX	$16950	28000	Silver

Figure 1: The user interface for multi-select faceted navigation.

id	types	prob
1	*a, c, d*	*0.1*
2	*a, c, e*	*0.1*
3	*a, b, e*	*0.5*
4	*b, c*	*0.3*

(a)



(b) (c)

Figure 2: Two decision trees, (b) and (c), for exploring the database (a).

2.1 The Navigation Model

Let $\mathbb{F}$ be the set of all the facet values and $\mathbb{R}$ be the set of all the resources in database.

Definition 1 *A* descriptor *is a conjunctive clause of facet values, where the only propositional operators are* AND *($\wedge$) and* NOT *($\neg$). Given $T \subseteq \mathbb{F}$ and a descriptor D, D is called a* finest descriptor *of T if D is a conjunctive clause of all the values in T.*

The descriptor in Figure 1 is $Dallas \wedge \neg Ford \wedge Silver$, which refines the whole database of size 15096 into a result of size 441. It is appended into the input box automatically.

Users arrive at targeted resources by a series of navigation steps, named a *session*. Each step is represented as a triple $< R, T, D >$, where R is current query result, T is the set of suggested values, and D is a finest descriptor of T which refines R into the query result at the next step. By default, the query result at the initial step is the whole database.

Denotation 1 *Given $R \subseteq \mathbb{R}$ and a descriptor D, $R^D = \{r : r \in R \wedge r$ satisfies $D\}$. Given a set T of suggested values, $des(R,T) = \{D :$ (D is a finest descriptor of T) $\wedge\ R^D \neq \emptyset\}$ and $par(R,T) = \{R^D : D \in des(R,T)\}$.*

Each resource in query result corresponds to a unique finest descriptor of suggested values. Hence $des(R,T)$ partitions R into the classes $par(R,T)$. Consider Figure 2(a). Suppose $X = \{b, c\}$ is suggested initially. There are four combinations of the suggested values, $b \wedge c$, $b \wedge \neg c$, $\neg b \wedge c$ and $\neg b \wedge \neg c$. No resource satisfies $\neg b \wedge \neg c$. Hence $des(X) = \{b \wedge c, b \wedge \neg c, \neg b \wedge c\}$, and $par(X) = \{\{r_4\}, \{r_3\}, \{r_1, r_2\}\}$.

Suppose $f(\cdot)$ is the function for suggesting values upon a given resource set. Our navigation model can be described by the following decision tree. Its root corresponds to the whole database, labeled with $f(\mathbb{R})$. Let n be a node corresponding to resource set R. Then n is labeled with $f(R)$. For each finest descriptor $D \in des(R, f(R))$, there is a child n' of n which corresponds to R^D. The edge between n and n' is labeled with D. Each leaf corresponds to a unique resource. Accordingly, the path from the root to a leaf represents the session of reaching the resource in the leaf. Figure 2(b) and (c) depict two different decision trees for exploring the database in Figure 2(a), which have different navigation cost.

2.2 The Cost Model

The navigation cost at one step consists of the effort of browsing suggested values and the effort of deciding a finest descriptor. The browsing effort is in proportion to the number of suggested values, while the decision effort is in proportion to the number of partitioned classes by the suggested values. Let T be the set of suggested values upon query result R. The navigation cost at this step is $(1-\alpha)|T| + \alpha|par(R,T)|$, where α is the weight for decision effort and $0 < \alpha < 1$.

The decision tree helps calculate the expected cost of reaching a targeted resource in database. Let $tree(f)$ be the decision tree created from the decision function f. For any node n, let T_n, R_n and o_n be its label, its corresponding resource set and the number of its children, respectively. Equation 1 depicts the expected navigation cost of reaching a resource in database, where $p(r)$ is the probability of visiting resource r. Note that no value is suggested in leaf.

$$nav\_cost(f) = \sum_{\text{node } n \in tree(f)} ((1-\alpha)|T_n| + \alpha\, o_i) \sum_{r \in R_n} p(r) \quad (1)$$

Let f_1 and f_2 be the decision functions leading to the decision trees depicted in Figure 2(b) and (c), respectively. Then $nav\_cost(f_1) = 2.2 + 1.2\alpha$ and $nav\_cost(f_2) = 1.9 + 1.9\alpha$. f_1 is better than f_2 in case that $\alpha > \frac{3}{7}$.

Definition 2 *For $R \subseteq \mathbb{R}$ and $u \in \mathbb{F}$, u is a* refining value *of R if and only if $\emptyset \subset R^u \subset R$. The set of all the refining values of R is denoted by $ref(R)$.*

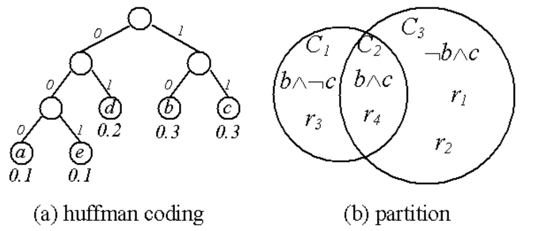
Obviously, only the refining values of query result should be suggested. Hence the problem is finding out the decision function $f^\star$ minimizing Equation 1, with the constraint that for any $R \subseteq \mathbb{R}$, there is $f(R) \subseteq ref(R)$.

It is prohibitive to calculate $f^\star$ since we need to consider every subset of refining values at each step. We observe that both the number of the nodes in the decision tree and the length of node labels are contradictory to each other. For example, the decision tree depicted in Figure 2(c) has shorter node label but more nodes than that in Figure 2(b). To minimize the expected navigation cost, a tradeoff is required between these two factors. In the next section, we will use the MDL principle to achieve the tradeoff.

3 The MDL Principle for Suggesting Facet Values

Given a query result R, the set T of suggested values should possess two desirable properties, preciseness and conciseness. In our navigation scheme, the preciseness property means to minimize the number of session steps, while the conciseness property means to minimize user efforts at one step. Preciseness can be achieved by suggesting all the refining values of R, which maximizes $par(R,T)$. Conciseness can be achieved by suggesting a single value, which minimizes $par(R,T)$. Hence preciseness and conciseness are contradictory to each other.

The Minimum Description Length (MDL) principle is widely used for finding an optimal tradeoff between preciseness and conciseness in information theory. It roughly states that the best hypothesis H to explain a dataset D is the one which minimizes $mdl\_cost(H) = L(H) + L(D|H)$, where $L(H)$ is the size of the hypothesis and $L(D|H)$ is the size of the dataset when encoded with the help of the hypothesis.



$$L(H)=(1-\alpha)(hcl(b)+hcl(c))+3\alpha(1+hcl(b)+1+hcl(c))$$

$$L(D|H)=p^*(r_1)L(r_1|H)+p^*(r_2)L(r_2|H)+p^*(r_3)L(r_3|H)+p^*(r_4)L(r_4|H)$$
$$=0.2\times(2+hcl(d))+0.2\times(2+hcl(e))+1.0\times2+0.6\times2$$

Figure 3: The MDL cost at the root in Figure 2(b).

In our setting, the dataset is R and the hypothesis is $des(R,T)$. This is quite natural because our goal is finding the optimal partitioning of R, which translates to finding the best hypothesis using the MDL principle.

Figure 3 depicts our representation of $L(H)$ and $L(D|H)$. We encode facet values by a Huffman code [Huffman, 1952] of their user preferences, so that high-preference values are prior to be suggested to reduce users' decision effort. Figure 3(a) depicts the Huffman tree organizing a, b, c, d, e with preference $0.1, 0.3, 0.3, 0.2, 0.1$, respectively. Figure 3(b) shows the partitioned classes of the whole database by $\{b, c\}$. The descriptors of the partitioned classes C_1, C_2 and C_3 are $b \wedge \neg c$, $b \wedge c$ and $\neg b \wedge c$, respectively.

$$L(H) = (1-\alpha)\sum_{u \in T} hcl(u) + \alpha \sum_{d \in des(R,T)} \sum_{u \in d} (1 + hcl(u)) \quad (2)$$

$$L(D|H) = \sum_{C \in par(R,T)} \sum_{r \in C} p^\star(r)(|T| + \sum_{u \in r \cap ref(C)} hcl(u))$$
$$\text{where } p^\star(r) = \frac{p(r)}{max_{t \in R}\{p(t)\}} \quad (3)$$

We formulate $L(H)$ by Equation 2, where $hcl(.)$ denotes the code length of a given value. $L(H)$ consists of two parts, the code length of all the suggested values and the code length of all the classes' descriptors, which represent the browsing effort and decision effort, respectively. For each value in a descriptor, an additional bit is used to indicate its being *true* or *false*. In Figure 3, each descriptor has the same code length $1 + hcl(b) + 1 + hcl(c) = 6$, and $L(H) = 4 + 14\alpha$.

We formulate $L(D|H)$ by Equation 3. The code of a resource consists of two parts. The first part is the identifier of the class it belongs to, whose length is equal to the number of suggested values. The second part is a set of its values u such that u is not in the descriptor of C and that u is a refining value of C, where C is the class the resource belongs to. In Figure 3, r_3 belongs to class C_1 which has no refining value. Hence r_3 is just encoded as the identifier of C_1, the length of which is 2. So does r_4. Since a is not a refining value of class C_3, the code length of r_1 is $2 + hcl(d)$, while that of r_2 is $2 + hcl(e)$. Note that the visiting probability of each resource is taken into account, so that popular resources can be reached soon.

We note that $L(H)$ measures the degree of conciseness, while $L(D|H)$ that of preciseness. The optimal set of suggested values that minimizes $mdl\_cost(H)$ achieves the tradeoff between preciseness and conciseness. It is prohibitive to find out the optimal solution since we need to consider every subset of refining values of a given resource set.

4 A Greedy Algorithm for Value Suggestion

In this section, we present our approach for computing the suggested values, called *GREEDY*. Suppose R be current query result. There are $2^{|ref(R)|}$ kinds of value suggestion. *GREEDY* approximates the global optimum by a local optima, as depicted in Figure 4. First, initialize the set T of suggested values as empty. Then iteratively select the value from $ref(R)$ which can reduce the MDL cost to a largest extent, and add the value into T. T is decided when no value can reduce the MDL cost any further.

Given $u \in ref(R)$, its reduced MDL cost $\nabla(u)$, i.e., $mdl\_cost(T) - mdl\_cost(T \cup \{u\})$, is described in Equation 4, where $U = \{C : C \in par(R,T) \wedge u \in ref(C)\}$.

$$\nabla_T(u) = -(1-\alpha)hcl(u) - \alpha(|par(R,T)|(1+hcl(u)) + |U| \sum_{v \in T \cup \{u\}} (1+hcl(v))) + \frac{1}{max_{t \in R}\{p(t)\}}(\sum_{C \in U} \sum_{v \in ref(C)} red_u(C,v)hcl(v) - 1), \tag{4}$$

$$\text{where } red_u(C,v) = \begin{cases} \sum_{r \in C^u} p(r) & C^v \supseteq C^u \\ \sum_{r \in C^v - C^u} p(r) & C^v \cup C^u = C \\ 0 & \text{otherwise} \end{cases}$$

After adding u, the bit length of the hypothesis increases. The bit length of the dataset, however, may either increase or decrease. Given a resource, the length of its class identifier increases by one. If the resource belongs to any class in $par(R,T) - U$, it has no other change. Otherwise, it can be compressed further when more of its values become non-refining with respect to a new class the resource belongs to.

Let v be a refining value of the class $C \in U$. In case that $C^v \supseteq C^u$, v is not a refining value of C^u. Hence the code length of each resource in C^u is reduced by $hcl(v)$. In case that $C^v \cup C^u = C$, v is not a refining value of $C^{\neg u}$. Hence the code length of each resource in $C^v - C^u$ is reduced by $hcl(v)$. Both cases cannot hold simultaneously since $C^v \subset C$.

4.1 Fast Computation of Cost Reduction

Real-time response is important for a faceted navigation system. It is time-consuming to calculate $red_u(C,v)$ in $\nabla(u)$, which requires a linear scanning of all the resources in C.

We adapt FP-Tree [Han *et al.*, 2000] to index a resource set. Figure 5 depicts the FP-Tree indexing $R = \mathbb{R}^a$ in Figure 2. It is constructed as follows. First, a header table keeps $ref(R)$, ranked in decreasing order of $|R^v|$ for $v \in ref(R)$. Then, for $r \in R$, the values in $r \cap ref(R)$ are rearranged in the same order as in the header table, and the resulting sequence is inserted into a prefix tree. Each node i in the prefix tree also preserves the number of the resources passing it, denoted by $supp(i)$, as well as the sum of their probability, denoted by $prob(i)$. Each entry v in the header table also keeps the list of all the tree nodes with the label v, denoted by $N(v)$.

We index each $C \in par(R,T)$ by a different FP-Tree to help calculate $red_u(C,v)$ fast. Based on set theory, $C^v \supseteq C^u$ if and only if $|C^{u \wedge v}| = |C^u|$, while $C^v \cup C^u = C$ if and only if $|C^u| + |C^v| - |C^{u \wedge v}| = |C|$. Without loss of generality, suppose v is before u in the header table. Denote $N_v(u) = \{i : i \in N(u) \wedge \exists j \in N(v)$ (j is an ancestor of i)$\}$. We have $|C^{u \wedge v}| = \sum_{i \in N_v(u)} supp(i)$. Hence the conditions in $red_u(C,v)$ can be decided fast. $red_u(C,v)$ is also calculated quickly since $\sum_{r \in C^u} p(r) = \sum_{i \in N(u)} prob(i)$ and $\sum_{r \in C^v - C^u} p(r) = \sum_{i \in N(v)} prob(i) - \sum_{i \in N_v(u)} prob(i)$.

Algorithm 1 The *GREEDY* algorithm

```
Input: a resource set $R$
Output: a set $T$ of suggested faceted values
1: $T = \emptyset$; $X = ref(R)$; $cls = \{R\}$; $\gamma = \frac{1}{max_{t \in R}\{p(t)\}}$;
2: while $X \neq \emptyset$ do
3:   $max\_reduc = 0$; $max\_value =$ null;
4:   for all $u \in X$ do
5:     $sum = -(1-\alpha)hcl(u) - \alpha|cls|(1+hcl(u)) - \gamma$;
6:     $\theta = \alpha \sum_{v \in T \cup \{u\}}(1+hcl(v))$;
7:     for all $C \in cls$ such that $u \in ref(C)$ do
8:       $sum -= \theta$;
9:       for all $v \in ref(C)$ do
10:        if $C^v \supseteq C^u$ then
11:          $sum += \gamma \cdot hcl(v) \sum_{r \in C^u} p(r)$;
12:        else if $C^v \cup C^u == C$ then
13:          $sum += \gamma \cdot hcl(v) \sum_{r \in C^v - C^u} p(r)$;
14:        end if
15:      end for
16:    end for
17:    if $sum > max\_reduc$ then
18:      $max\_reduc = sum$; $max\_value = u$;
19:    end if
20:  end for
21:  if $max\_value$ is null then
22:    break;
23:  else
24:    $X = X - \{max\_value\}$; $T = T \cup \{max\_value\}$;
25:    for all $C \in cls$ such that $u \in ref(C)$ do
26:      $cls = cls \cup \{C^u, C^{\neg u}\} - C$;
27:    end for
28:  end if
29: end while
30: return $T$;
```

Figure 4: The *GREEDY* algorithm.

5 Empirical Evaluation

We now present the empirical evaluation of our approach. The primary goal is to evaluate its effectiveness in decreasing navigation cost as well as its efficiency for value suggestion.

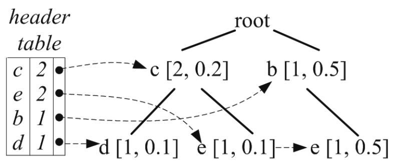


Figure 5: The FP-Tree indexing $\mathbb{R}^a$ in Figure 2.

Table 1: Dataset Characteristics

UsedCars (size=15, 096, ave_len=7, max_len=7)

facet	model	year	location	price	mileage	color
#value	37	15	164	15	15	450

IMDB (size=23, 919, ave_len=14.9, max_len=315)

facet	year	genre	rating	director	actor	actress
#value	40	26	10	16, 387	161, 233	92, 821

5.1 Experimental Setup

We evaluate our approach, referred to as MULTI, on two datasets, *UsedCars* [1] and *IMDB* [2]. Table 1 describes their size, average resource length *ave_len*, maximal resource length *max_len*, facets and the number of different values in each facet. We assume that numeric facets have been appropriately discretized.

We compare MULTI with the state-of-the-art approach FACETOR [Kashyap *et al.*, 2010]. Both the human-computer interaction style and the suggestion algorithm in MULTI contribute to the reduction of navigation cost. The former can be easily implemented in current faceted navigation systems to reduce their navigation cost as well. Someone may wonder how much effectiveness is achieved by our suggestion algorithm. Hence we also compare MULTI with FACETOR-E, a naive extension of FACETOR by allowing multiple selection and exclusion option.

For each dataset, we randomly select 100 resources as target, and the navigation of each targeted resource is simulated 1000 times [3]. At each navigation step in FACETOR, we assume a user randomly selects a suggested value contained in the targeted resource. At each navigation step in FACETOR-E, we assume a user selects all the suggested values contained in the targeted resource, and one more value for excluding resources. At each navigation step in MULTI, we assume a user iteratively and randomly selects an undecided value until a finest descriptor is formed. Note that quite a few suggested values can be decided automatically by the system.

The cost model in FACETOR is slightly different from ours. To make a fair comparison, the navigation cost at one step in all the three approaches is set to be the sum of the number of suggested values and the number of mouse click.

The visiting probability of each resource was assumed the same. So was the user preference of each facet value. The coefficient for decision effort α is set 0.8 empirically. All the experiments were conducted on a Lenovo M7000 PC with 2.93GHZ dual-core CPU and 2G RAM.

[1]`http://autos.yahoo.com/used_cars.html`

[2]`http://www.imdb.com/interfaces`

[3]We also test other groups of resources by the same experimental methodology, whose results are quite similar.

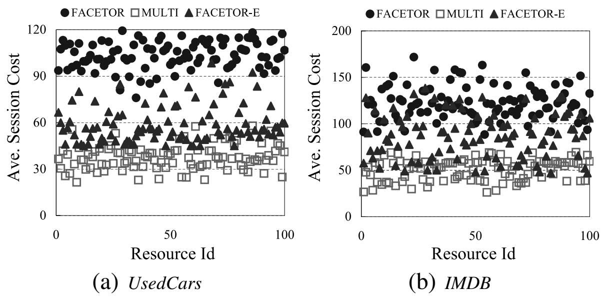


Figure 6: The navigation cost of a session.

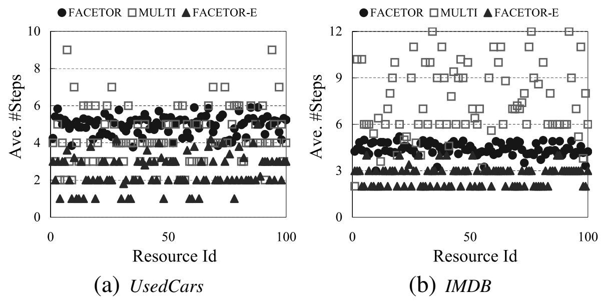


Figure 7: The number of steps in a session.

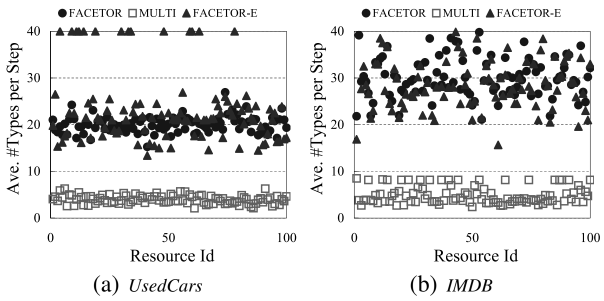


Figure 8: The number of suggested values per step.

5.2 Experiments with Navigation Cost

Figure 6 depicts the average navigation cost of reaching a targeted resource. In *UsedCars*, the session cost of FACETOR, FACETOR-E and MULTI is 103.5, 58.8 and 37.4 in average, respectively. In *IMDB*, the average session cost of FACETOR, FACETOR-E and MULTI is 123.3, 85.9 and 50.6 in average, respectively. It can be seen that MULTI leads to a significant saving and that our suggestion algorithm is critical in reducing navigation cost.

The above results can be explained from two aspects, the average number of the steps in reaching a targeted resource and the average number of suggested values per step.

Figure 7 depicts the average number of the steps in reaching a targeted resource. In *UsedCars*, the step number is 5.0, 2.5 and 4.5 in average for FACETOR, FACETOR-E and MULTI, respectively. In *IMDB*, the step number is 4.4, 2.7 and 8.7 in average for FACETOR, FACETOR-E and MULTI, respectively. Consider *UsedCars*. At each step in FACETOR, one suggested value contained by targeted resource must be selected. Hence its step number cannot exceed the maximal length of targeted resource, i.e. 7. This explains why its step number is quite stable around 5.0. Since FACETOR-E extends FACETOR by allowing multiple selection and exclusion option, it requires much fewer steps than FACETOR. Both FACETOR and FACETOR-E demand that each resource in current query result contains at least one suggested value. In MULTI, all the selected values at a step may be used to filter out the resources containing them. Hence its step number can exceed 7. A single value may be suggested, if it is an optimal tradeoff between conciseness and preciseness. This explains the fluctuation of the step number in MULTI.

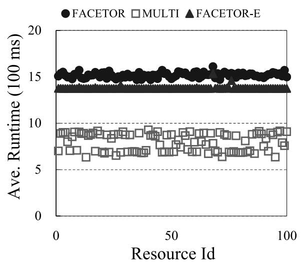
Figure 8 depicts the average number of suggested values at a step. In *UsedCars*, it is 20.0, 23.2 and 4.0 in average for FACETOR, FACETOR-E and MULTI, respectively. In *IMDB*, it is 30.3, 29.3 and 4.9 in average for FACETOR, FACETOR-E and MULTI, respectively. The closeness between FACETOR and FACETOR-E is due to their sharing the same suggestion algorithm. MULTI suggests much fewer values. The first reason is that FACETOR(-E) prefers to suggesting the values which correspond to a medium portion of current query result and have small overlapping, while MULTI prefers to suggesting the values which correspond to a large portion of current query result and have medium overlapping. The second reason is that FACETOR(-E) has the coverage requirement while MULTI not, which demands that each resource in current query result contains at least one suggested value. Take the initial step in *UsedCars* as an example. MULTI suggests 11 values, only a quarter of 40 suggested values in FACETOR(-E).

5.3 Experiments with Execution Time

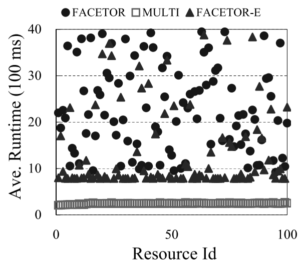
Figure 9 depicts the total running time of computing suggested values in a session. In *UsedCars*, it is 1518.3, 1377.9 and 794.6 milliseconds in average for FACETOR, FACETOR-E and MULTI, respectively. In *IMDB*, it is 2303.0, 1454.9 and 246.0 milliseconds in average for FACETOR, FACETOR-E and MULTI, respectively. Compared with FACETOR, MULTI runs two times faster in *UsedCars* while nearly an order of magnitude in *IMDB*, although our computation task is much more complex than FACETOR's. The first reason is that MULTI suggests fewer values and requires fewer steps. The second reason is that FACETOR adopts a brute-force scanning of current query result while MULTI creates an FP-tree indexing for acceleration.

6 Conclusion and Future Work

Faceted navigation is effective in reducing information overload in the process of reaching targeted resources. The effectiveness of conventional faceted navigation systems is limited by their navigation scheme. We propose a new navigation scheme to allow multiple selection and a retaining/exclusion option. Furthermore, we employ the Minimum Description Length principle to minimize the navigation cost



(a) *UsedCars*



(b) *IMDB*

Figure 9: The runtime for suggesting values in a session.

of a session, which fits our navigation scheme. An empirical study demonstrates that our approach is more cost-saving and efficient than state-of-the-art approaches.

For simplifying our navigation model, we assume users refine query result by a finest descriptor at each step. In practice, however, users may not be able to decide some suggested values. In our future work, we will explore a more general navigation model in which users can decide any number of suggested values at each step.

Acknowledgments

This research work was funded by the National High-tech R&D Program of China under grant No. 2010AA012500, and the National Natural Science Foundation of China under grant No. 61003166 and grant No. 60933005.

References

[Basu Roy *et al.*, 2008] Senjuti Basu Roy, Haidong Wang, Gautam Das, Ullas Nambiar, and Mukesh Mohania. Minimum-effort driven dynamic faceted search in structured databases. In *CIKM '08*, pages 13–22, 2008.

[Bergman, 2001] M.K. Bergman. The deep web: Surfacing hidden value. *Journal of Electronic Publishing*, 7(1), 2001.

[Dakka and Ipeirotis, 2008] Wisam Dakka and Panagiotis G. Ipeirotis. Automatic extraction of useful facet hierarchies from text databases. In *ICDE '08*, pages 466–475, 2008.

[Han *et al.*, 2000] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *SIGMOD '00*, pages 1–12, 2000.

[He *et al.*, 2007] Bin He, Mitesh Patel, Zhen Zhang, and Kevin Chen-Chuan Chang. Accessing the deep web. *Commun. ACM*, 50:94–101, May 2007.

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

[Huffman, 1952] D.A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[Kashyap *et al.*, 2010] Abhijith Kashyap, Vagelis Hristidis, and Michalis Petropoulos. Facetor: Cost-driven exploration of faceted query results. In *CIKM '10*, pages 719–728, 2010.

[Koren *et al.*, 2008] J. Koren, Y. Zhang, and X. Liu. Personalized interactive faceted search. In *WWW '08*, pages 477–486, 2008.

[Lee *et al.*, 2007] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: a partition-and-group framework. In *SIGMOD '07*, pages 593–604, 2007.

[Lee *et al.*, 2009] Bongshin Lee, Greg Smith, George G. Robertson, Mary Czerwinski, and Desney S. Tan. Facetlens: exposing trends and relationships to support sensemaking within faceted datasets. In *SIGCHI '09*, pages 1293–1302, 2009.

[Li *et al.*, 2010] C. Li, N. Yan, S.B. Roy, L. Lisham, and G. Das. Facetedpedia: dynamic generation of query-dependent faceted interfaces for wikipedia. In *WWW '10*, pages 651–660, 2010.

[Marchionini, 2006] Gary Marchionini. Exploratory search: from finding to understanding. *Commun. ACM*, 49:41–46, April 2006.

[Navlakha *et al.*, 2008] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph summarization with bounded error. In *SIGMOD '08*, pages 419–432, 2008.

[Sinha and Karger, 2005] V. Sinha and D.R. Karger. Magnet: Supporting navigation in semistructured data environments. In *SIGMOD '05*, pages 97–106, 2005.

[Tunkelang, 2006] D. Tunkelang. Dynamic category sets: An approach for faceted search. In *SIGIR '06 Workshop on Faceted Search*, 2006.

[van Zwol and Sigurbjornsson, 2010] R. van Zwol and B. Sigurbjornsson. Faceted exploration of image search results. In *WWW '10*, pages 961–970, 2010.

[Yee *et al.*, 2003] Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti Hearst. Faceted metadata for image search and browsing. In *SIGCHI '03*, pages 401–408, 2003.