Minimum-Effort Driven Dynamic Faceted Search in Structured Databases

Senjuti Basu Roy
Dept of Computer Sci and Eng
University of Texas at Arlington
Arlington TX, USA
roy@cse.uta.edu

Haidong Wang
Dept of Computer Sci and Eng
University of Texas at Arlington
Arlington TX, USA
haidong.wang@uta.edu

Gautam Das*
Dept of Computer Sci and Eng
University of Texas at Arlington
Arlington TX, USA
gdas@cse.uta.edu

Ullas Nambiar
IBM India Research Lab
New Delhi, India
ubnambiar@in.ibm.com

Mukesh Mohania
IBM India Research Lab
New Delhi, India
mkmukesh@in.ibm.com

ABSTRACT

In this paper, we propose minimum-effort driven navigational techniques for enterprise database systems based on the faceted search paradigm. Our proposed techniques dynamically suggest facets for drilling down into the database such that the cost of navigation is minimized. At every step, the system asks the user a question or a set of questions on different facets and depending on the user response, dynamically fetches the next most promising set of facets, and the process repeats. Facets are selected based on their ability to rapidly drill down to the most promising tuples, as well as on the ability of the user to provide desired values for them. Our facet selection algorithms also work in conjunction with any ranked retrieval model where a ranking function imposes a bias over the user preferences for the selected tuples. Our methods are principled as well as efficient, and our experimental study validates their effectiveness on several application scenarios.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems—*query processing, relational databases*; H.3.3 [**Information Search and Retrieval**]: Query formulation, search process

General Terms

Algorithms, Design, Performance

Keywords

Dynamic Facet Generation, Minimum-effort Entity Search, Data browsing

*Part of the work done as a visiting researcher at IBM India Research Lab.

CIKM'08, October 26–30, 2008, Napa Valley, California, USA.

1. INTRODUCTION

One of the primary problems that many organizations face is that of facilitating effective search for data records within vast data warehouses. For example, consider the customer database of a large financial institution such as a bank. A data analyst or a customer service representative for the bank often has to search for records of a specific customer or a specific account in such databases. Of course, if the relevant tuple is uniquely identifiable by an identifier known to the user, this problem is trivial. But in most cases the user only has partial information about the tuple (e.g., perhaps the values of a few of its attributes) and thus it is necessary to enable an effective search procedure. As another example, consider a potential car buyer searching for a suitable used car listed in an online auto dealer's database, where each car is described via numerous attributes such as Make, Model, Mileage, and so on. While the buyer is eventually interested in buying only one car, at the beginning of her search she may only have a few preferences in mind (e.g., a late model family sedan with low mileage); thus a search is necessary to narrow down the choices.

One approach for enabling tuple search in databases is IR style *ranked retrieval* from databases. For the cars example above, a query such as "CarType=sedan, Age<5, Mileage<10k" can be specified via a form-based interface, and rather than simply executing the query using SQL - which will result in a flood of results since there are presumably many cars in the database that satisfy such broad query conditions - ranking-based systems will attempt to rank and retrieve the top-k most "relevant" tuples that satisfy these conditions (where k is usually a small number, such as 10-20). Much of the recent research has focused on the design of suitable ranking functions, as well as on the design of efficient retrieval algorithms [7, 12, 13].

However, recently, other search paradigms have gained popularity in certain specialized IR domains, including for searching over image and text data. In particular, it has been argued that *faceted search* interfaces can be extremely useful in user navigation and search [3, 6]. E.g., a user searching for a photograph of the Great Wall at a photo hosting website may have the option of drilling down via different facets of the dataset, e.g., first by geographical regions (such as Asia → China → Beijing), then via age (such

as period → ancient), then via phototype (man made → historical monuments). While it remains to be seen if faceted search is a viable option for searching at the Web scale, it does offer a promising alternative in specialized domains such as these examples.

Main Goal - Investigate Faceted Search in Databases: The main goal of this paper is to explore the opportunities of adopting principles of the faceted search paradigm for tuple search in structured databases. However, unlike past works on images and text data, where the primary task is to design hierarchical meta-data and facets to enable faceted search, structured databases come with the tremendous advantage that they are already associated with rich meta-data in the form of tables, attributes and dimensions, known domain ranges, and so on. Instead, the challenge is to determine, from the abundance of available meta-data, which attributes of the tuples are best suited for enabling a faceted search interface. In the cars database example above, a very simple faceted search interface is one where the user is prompted an attribute[1] (e.g., Make), to which she responds with a desired value (e.g., "Honda"), after which the next appropriate attribute (e.g., Model) is suggested to which she responds with a desired value (e.g., "Accord"), and so on. In this paper we focus on two broad problem areas. We briefly elaborate on these problems and our solutions below.

1. Faceted Search as an Alternative to Ranked-Retrieval: We first consider the problem where we don't assume any tuple relevance and ranking function as being available. Thus when a user poses an initial selection query, without any further information from the user we can only assume that all of the selected tuples are equally preferred by the user. Our task is then to develop a dialog with the user to extract more information from her on other desired attribute values - essentially initiate a facet-by-facet drill down procedure to enable her to zoom in on the tuple(s) of interest. Our overall goal is to judiciously select the next facets dynamically at every step, so that the user reaches the desired tuples with *minimum effort*. While the effort expended by a user during a search/navigation session may be fairly complex to measure, we focus on a rather simple but intuitive metric: *the expected number of queries that the user has to answer in order to reach the tuples of interest.*

Variants of this problem have been considered in [2] in the context of interactive question-answer applications. It was shown that the problem is intractable, and an approximation algorithm suggested with provably good performance. While we adopt the same cost metric, we extend the idea in several important ways. We propose a novel cost model for fast tuple search which assumes that attributes are associated with *uncertainties*, where the uncertainty of an attribute refers to the probability with which a user can provide a value that belongs to the domain of the attribute. We develop facet selection techniques that take into account such uncertainties.

Also, we formally show that the approximation algorithm for building minimal cost decision trees given in [2] generates trees different from those generated by other classical decision tree construction algorithms based on information gain, as well as other classical dimensionality reduction techniques such as principal component analysis (PCA).

[1]Henceforth in this paper *facets* and *attributes* will be used interchangeably.

2. Faceted Search that Leverages Ranking Functions: We next ask whether faceted search procedures can work *in conjunction with* ranking functions. This is a novel problem area, and to the best of our knowledge, has not been investigated before. Recall that a ranked-retrieval system typically assigns relevance scores to all selected tuples and returns only the top-k tuples. From a faceted search perspective, we may view the ranking function as imposing a *skew* over the user preferences for the selected tuples, and thus would like to select the facet that directs the user towards the most preferred tuples as efficiently as possible. One interesting complication is that these tuple preferences (or scores) may change as the faceted search progresses; this is because as new attribute information is provided by the user, the ranking function may re-evaluate the scores of the remaining tuples still in contention. Thus a faceted search system in conjunction with a ranking function offers the benefits of focused retrieval as well as drill-down flexibility. We provide a formal definition of this problem, and offer a solution for facet selection that is based on minimum-effort driven principles.

The main contributions of our paper may be summarized as follows:

1. We initiate research into the problem of automated facet discovery for enabling minimal effort browsing of entities (tuples) in structured databases. We adopt a simple approximation algorithm, and show how this approach can be extended to incorporate the notion of user uncertainty in providing binding values for facets. We discuss how this approach is different from other attribute selection techniques.

2. We also extend our methods to work in conjunction with ranking functions for tuples. We show how our methods are different from other attribute selection techniques in the presence of ranking functions such as [18].

3. We develop novel scalable implementation techniques of our algorithms using a modified Rainforest framework [15].

4. We describe the results of a thorough experimental evaluation of our proposed techniques.

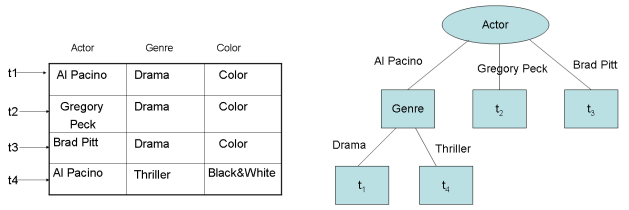
The rest of this paper is organized as follows. Section 2 presents faceted search as an alternative to ranked retrieval. In Section 3, we discuss faceted search algorithms that leverage ranking functions. Section 4 describes the evaluations performed and discusses the results. Related work is described in Section 5 and conclusions are given in Section 6.

2. FACETED SEARCH AS AN ALTERNATIVE TO RANKED RETRIEVAL

Let D be a relational table with n tuples $\{t_1, t_2, \ldots, t_n\}$ and m categorical attributes $A = \{A_1, A_2, \ldots, A_m\}$, each with domain Dom_i (for the rest of this paper we only consider categorical data, and assume that numeric data has been suitably discretized). Assume that no two tuples are identical and that a user wishes to retrieve a tuple from this database. The faceted search system will prompt the user with a series of questions, where each question takes the form of an attribute name, and to which the user responds

with a value from its domain. This drill-down process terminates when a unique tuple has been isolated. The task is to design a faceted search system which asks the minimum number of questions on the average, assuming that each tuple is equally likely to be preferred by the user (thus we do not assume the presence of a ranking function).

Essentially, the task is to build a *decision tree* which distinguishes each tuple by testing attribute values (asking questions). Each node of the tree represents an attribute, and each edge leading out of the node is labeled with a value from the attribute's domain. As an example consider Figure 1(a) which refers to a toy *movie* database with three attributes and four tuples. A decision tree for identifying each of the tuples in the tuple set $D = \{t_1, t_2, t_3, t_4\}$ is shown in Figure 1(b). The leaves of the tree represent the tuple set D and each tuple appears exactly once. A user reaches her tuple of interest by picking a path after each non-leaf node in the tree i.e. by assigning a value to each attribute query on the path leading to the tuple.



(a) A small movies database (b) An optimal decision tree

Figure 1: A Small Movie Database and an Optimal Decision Tree

Given such a tree T, $cost(T)$ can be defined as the average tree height, $\sum_i ht(t_i)/n$ where $ht(t_i)$ is the height of leaf t_i. Equivalently, cost (i.e., effort) represents the expected number of queries that needs to be answered before the user arrives at a preferred tuple (assuming all tuples are equally likely to be preferred). It is easy to verify that the tree in Figure 1(b) is optimal (with minimum cost $= (2 + 2 + 1 + 1)/4 = 1.5$).

The problem of determining the minimum cost tree has been studied in the past in the context of question-answering dialog systems, and shown to be NP-complete (see [2] and references therein). A greedy approximation algorithm has been developed [2] which achieves an approximation factor of $O(\log d \log n)$ in the cost, where d is the maximum domain size of any attribute. Although the approximation factor appears large, it is the only theoretical approximation bound known for this problem. Moreover, as our experiments show, this algorithm performs quite well in practice. We describe this algorithm next as it forms the foundation for all our facet selection procedures.

The intuition is that any decision tree should distinguish every pair of distinct tuples. The approach is to make the attribute that distinguishes the maximum number of pairs of tuples as the root of the tree, where an attribute A_l is said to distinguish a pair of tuples t_i, t_j if $t_i[l] \neq t_j[l]$. Picking the attribute A_l as the root node partitions the database D into disjoint tuple sets $D_{x_1}, D_{x_2}, \ldots, D_{x_{|Dom_l|}}$, where each D_{x_q} is the set of tuples that share the same attribute value x_q of A_l. Using this intuition, we seek to select as root attribute A_l that minimizes the number of indistinguishable pairs of tuples. Hence, formally the function, Indg() seeks to minimize,

$$Indg(A_l, D) = \sum_{1 \leq q \leq |Dom_l|} |D_{x_q}|(|D_{x_q}| - 1)/2 \qquad (1)$$

This process is recursively repeated for all sets D_{x_q}, until each set reduces to singleton tuples. Applying this algorithm to the database in Figure 1(a) gives the same resultant decision tree as shown in Figure 1(b). We see that $Indg(Actor) = 1$, while $Indg(Genre) = Indg(Color) = 3$. Thus Actor should be the root.

2.1 Comparing Against Other Attribute Selection Procedures

Comparing Against Information Gain: Decision tree construction is a very well understood process in machine learning and data mining, and several popular algorithms such as ID3 and C4.5 have been developed [19]. These algorithms are designed for the *classification* problem, and seek to maximize classification accuracy and avoid over-fitting. In contrast, our goal is not to solve a classification problem - rather our aim is to build full decision trees (where each leaf is a tuple) that minimizes average root-to-leaf path lengths. A popular heuristic used by these algorithms (e.g., ID3) for selecting the next feature, or "splitting" attribute, is the *information gain* measure. Since there is no class variable associated with the database, we may imagine that each tuple consists of its own unique class, and thus the information gain of an attribute A_l is equivalent to

$$InfoGain(A_l, D) = \log n - \frac{1}{n}\left(\sum_{1 \leq q \leq |Dom_l|} |D_{x_q}| \log |D_{x_q}|\right) \qquad (2)$$

The selected facet may be the one with the largest information gain. Unlike the $Indg()$ based approach for which there are known approximation bounds, it is open whether similar approximation bounds exist for information gain based approaches. In fact, as we show now, the information gain heuristic produces different trees than the approach of minimizing $Indg()$.

LEMMA 2.1. *Given a database D, the decision tree constructed by selecting facets that minimize $Indg()$ may be different from the decision tree constructed by selecting facets that maximize $InfoGain()$.*

Proof: Consider two attributes A and B of a database table D. Let A be a Boolean attribute with domain $\{a_1, a_2\}$. Let $n(x)$ represent the number of tuples with attribute value x. Let $n(a_1) = n(a_2) = n/2$. Let the domain of B be $\{b_1, b_2, \ldots, b_{n/(2+\sqrt{2})+1}\}$ where $n(b_1) = n/\sqrt{2}$ and $n(b_2) = \cdots = n(b_{n/(2+\sqrt{2})+1}) = 1$. We then have

$$Indg(A, D) = \frac{n}{2}\left(\frac{\frac{n}{2} - 1}{2}\right) + \frac{n}{2}\left(\frac{\frac{n}{2} - 1}{2}\right) = \frac{n(n-2)}{4}$$

$$Indg(B, D) = \frac{n}{\sqrt{2}}\left(\frac{\frac{n}{\sqrt{2}} - 1}{2}\right) = \frac{n(n-\sqrt{2})}{4}$$

Clearly $Indg(B, D) > IndgA, D)$, and thus A will be preferred over B during facet selection. We next consider the

information gain heuristic. We then have

$$InfoGain(A, D) = \log n - \frac{1}{n}\left(\frac{n}{2}\log\left(\frac{n}{2}\right) + \frac{n}{2}\log\left(\frac{n}{2}\right)\right) = 1$$

$$InfoGain(B, D) = \log n - \frac{1}{n}(\frac{n}{\sqrt{2}}\left(\frac{n}{\sqrt{2}}\log\left(\frac{n}{\sqrt{2}}\right)\right))$$

$$= \log n - \frac{\left(\log n - \frac{1}{2}\right)}{\sqrt{2}}$$

Clearly $InfoGain(B, D) > InfoGain(A, D)$ and thus B will be preferred over A during facet selection. These arguments demonstrate that the tree produced by maximizing information gain may be different from the tree produced by minimizing $Indg()$. □

Comparing Against Principal Component Analysis (PCA): We explore the popular technique of *principal component analysis* (PCA) [17] to see if it is applicable in facet selection. PCA has traditionally been developed for dimensionality reduction in numeric datasets, thus extending PCA to categorical databases such as ours requires some care. We illustrate these ideas by again considering the small movies database in Figure 1(a). Suppose we wish to reduce the dimension of this database from three to two and decide to retain the dimensions Genre and Color. In that case, the attribute Actor has to be homogenized (i.e., all values have to be transformed to a single common value) such that the number of values that are changed is minimized. It is easy to see that if we make all Actors as "Al Pacino", this will require minimum number of changes (two changes, i.e., the Actor field in tuples t_2 and t_3). Hence the cost of the reduction is two in this case. On the other hand, if we decide to retain the dimensions Actor and Genre, only one value in the database needs to be changed (the Color field of t_4 has to be changed to "Color"). Thus, reducing the dimensions to Actor and Genre is cheaper than (and thus preferable to) reducing the dimensions to Genre and Color. More specifically, the best k attributes we retain are the ones that have the smallest modes. Mode of an attribute is defined as:

$$Mode(A_l, D) = \max\{|D_{x_q}|, |1 \leq q \leq |Dom_l|\}$$

LEMMA 2.2. *Given a database D, the decision tree constructed by selecting facets that minimize $Indg()$ may be different from the decision tree constructed by selecting facets that minimize $Mode()$.*

The proof for above lemma can be derived using similar intuition as for prior lemma. The details are omitted in the interest of space. Among all three heuristics, only the $Indg()$ based approach has a known approximation factor associated with it and performs better in experimental evaluation.

2.2 Modeling Uncertainty in User Knowledge

The facet selection algorithm presented above assumes that the user knows the answer to any attribute that is selected as the next facet. In a practical setting, this is not very realistic. For example, a customer service representative of a bank searching for a specific customer may not know exactly the street address of the customer's residence; likewise a user searching for a movie may not be sure of the director of the desired movie, and so on. One of the contributions of this paper is to recognize that there are inherent uncertainties associated with the user's knowledge of an entity's attribute values, and accordingly to build decision trees that take such uncertainties into account.

In the simplest case, each attribute A_i of the database is associated with a probability p_i that signifies the likelihood that a random user knows the answer to the corresponding query. For example, in a cars database, the attribute Car Type may be associated with a probability of 0.8 (i.e., 80% of users know whether they want a sedan, hatchback, SUV, etc.) For simplicity we assume no correlations between attribute uncertainties (i.e., a user who does not know the car type is still assumed to specify heated seats with a finite probability) nor other more general uncertainty models. Estimating these probabilities require access to external knowledge sources beyond the database such as domain experts, user surveys, and analyzing past query logs.

In this paper, we assume that the uncertainty models have already been estimated. In designing our decision trees to cope with uncertainty, we assume that users can respond to a question by either (a) providing the correct value of the queried attribute A_i, or (b) responding with a "don't know". In either case, the faceted search system has to respond by questioning the user with a fresh attribute. Consider Figure 2, which shows the decision tree of the same database of Figure 1(a). Assume each of the attributes has associated uncertainties. Consequently, each node in the decision tree also has an associated "don't know" link. As can be seen, the leaf nodes in this decision tree are either a single tuple, a set of tuples, or, at worst, the entire database. Moreover, note that the tuples of the database do not occupy unique leaves in the decision tree. For example, there are 7 different path instances of tuple t_1. This implies that when attempting to reach a tuple, different users may follow different paths through the tree.

At this context, we organized a small survey among 20 people selected from the students and faculty of our university.In that survey, each person assigned a value (between 0 to 1) for each attribute. This value denotes the likelihood (probability) with which she is able to answer the question corresponding to that attribute. The overall probability of each attribute is calculated by averaging all 20 values.

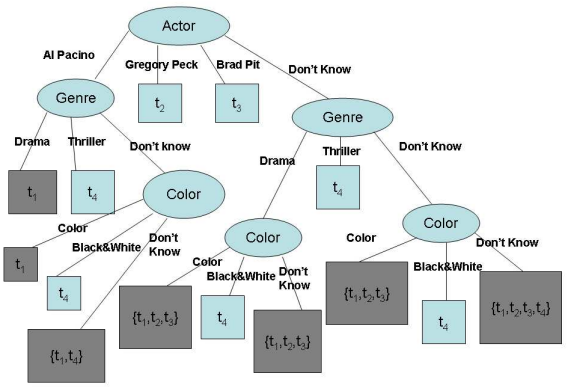


Figure 2: The decision tree of Figure 1(a) with uncertainty models

Thus our challenge is to build such decision trees such that the expected path length through the tree is minimized. Our Single Facet based search algorithm is shown in Algorithm 1.

Algorithm 1 Single Facet Based Search(D, A')

Input: D, a set $A' \subset A$ of attributes not yet used
Global parameters: an uncertainty p_i for each attribute $A_i \in A$
Output: A decision tree T for D
begin
if $|D| = 1$ **then**
  Return a tree with any attribute $A_l \in A'$ as a singleton node
end if
if $|A'| = 1$ **then**
  Return a tree with the attribute in A' as a singleton node
end if
Let A_l be the attribute that distinguishes the maximum *expected* number of pairs in D:
$A_l = argmin_{A_s \in A'}(1 - p_s) \times |D|(|D| - 1)/2 + p_s \times Indg(A_s, D)$
Create the root node with A_l as its attribute label
for each $x_q \in Dom_l$ **do**
  Let $D_{x_q} = \{t \in D | t[l] = x_q\}$
  T_{x_q} = Single-Facet-Based-Search$(D_{x_q}, A' - \{A_l\})$
  Add T_{x_q} to T by adding a link from A_l to T_{x_q} with label x_q
end for
Create the "don't know" link:
T' = Single-Facet-Based-Search$(D, A' - \{A_l\})$
Add T' to T by adding a link from A_l to T' with label "don't know"
Return T with A_l as root
end

However, we note that each node A_l now has $|Dom_l| + 1$ links, with one of the links labeled as "don't know". This link is taken with probability $1 - p_l$, whereas the rest of the links are taken with probability p_l. Thus, in the former case, the attribute A_l cannot distinguish any further pairs of tuples (the query was essentially wasted), whereas in the latter case, only $Indg(A_l, D)$ pairs were left indistinguishable. Thus, we can see that if we select A_l as the root node, then the *expected* number of tuple pairs that cannot be distinguished is $(1 - p_l) \times |D|(|D| - 1)/2 + p_l \times Indg(A_l, D)$. Consequently, an obscure attribute that has little chance of being answered correctly by most users, but is otherwise very effective in distinguishing attributes, will be overlooked in favor of other attributes in the decision tree construction.

2.3 Extending to k-Facets Selection

Next, we extend our model further by giving the user more flexibility at every step. As a practical consideration, a decision tree as shown in Figure 2 can sometimes be tedious to a user. It may be more efficient to present, at every step, *several* (say k) attributes to the user at the same time, with the hope that the user might know the correct value of one of the proffered attributes.

It may appear that for designing the root node of the decision tree for the k-facet case, instead of considering only m possible attributes as we did for the single-facet case, we will need to consider m_{C_k} sets of attributes of size k each, and from them, select the set that is the best at disambiguating tuple pairs. However, if we restrict the user to answering only one question at each iteration, the problem of determining best k-facets at any node in this decision tree has a much simpler solution - we order the unused attributes from the one that distinguishes most number of tuple pairs to the one that distinguishes the least number of tuple pairs, and select the top-k attributes from this sequence.

In this tree, the probability that a random user will follow "don't know" links is much smaller than the single-facet case. For example, given the set of attributes A'' at the root, the probability that a random user will be unable to answer any of the k questions is $\prod_{A_l \in A''}(1 - p_l)$. Thus we expect such trees to be more efficient (i.e., shallower) than the trees in the single-facet case.

2.4 Designing a Fixed k-Facets Interface

In certain applications, it is disconcerting for the user to be continuously presented with new sets of attributes after every response. Such users would prefer to be presented with a single fixed form-like interface, in which a reasonably large (k) number of attributes are shown, and the user assigns values to as many of the preferred attributes as she can. If the space available on the interface is restricted such that only $k < m$ attributes can be shown, the task is then to select the best set of k attributes such that the expected number of tuples that can be distinguished via this interface can be maximized. We formalize this problem as follows: Given a database D, a number k, and uncertainties p_i for all attributes A_i, select k attributes such that the expected number of tuples that can be distinguished is maximized. If we assume that there are no uncertainties associated with attributes, this problem has similarities with the classical problem of computing minimum-sized *keys* of database relations, and with the problem of computing approximate keys of size k (see [11]).

However, in our case the problem is complicated by the fact that attributes are associated with uncertainties, thus such deterministic procedures [11] appear difficult to generalize to the probabilistic case. Instead, we propose a greedy strategy for selecting the k facets that is based on some of the underlying principles developed in our earlier algorithms. The overall idea is, if we have already selected a set A' of k' attributes, the task is then to select the next attribute A_l such that the expected number of pairs of tuples that cannot be distinguished by $A' \cup \{A_l\}$ is minimized.

Ignoring attribute uncertainties, the algorithm can be described as follows. Let $A' \cup \{A_l\}$ partition D into the sets $D_1, D_2, \ldots, D_d$ where within each set the tuples agree on the values of attributes in $A' \cup \{A_l\}$. Thus, we should select A_l such that the quantity $\sum_i |D_i|(|D_i| - 1)/2$ is minimized. Introducing attribute uncertainties implies that $A' \cup \{A_l\}$ does not always partition D into the sets $D_1, D_2, \ldots, D_d$. Rather, depending on the user interactions, the possible partitions could vary between finest possible partitioning, $P_{fine}(A' \cup \{A_l\}) = \{D_1, D_2, \ldots, D_d\}$, to the coarsest possible partitioning $P_{coarse}(A' \cup \{A_l\}) = \{D\}$ (the latter happens if the user responds to each attribute with a "don't know"). Each intermediate partitioning occurs when the user responds with a "don't know" to some subset of the attributes.

Consider any partitioning $P = \{U_1, U_2, \ldots U_u\}$. Let the quantity $IndgPartition(P)$ be defined as $\sum_i |U_i|(|U_i| - 1)/2$. This represents the number of tuple pairs that fail to be distinguished. Since each partitioning is associated with a probability of occurrence, we should thus select A_l such that

the expected value of $IndgPartition(P)$ is minimized. However, this process is quite impractical since the number of partitionings are exponential in $|A' \cup \{A_l\}|$, i.e., exponential in $k'+1$. We thus chose a simpler approach, by assuming that there are only two partitionings, the finest, as well as the coarsest. The probability of occurrence of the coarsest partitioning is $p(coarse) = \prod_{A_s \in A' \cup \{A_l\}}(1-p_s)$. Thus, we select A_l that minimizes

$$IndgPartition(P_{coarse}(A' \cup \{A_l\}))p(coarse) + IndgPartition(P_{fine}(A' \cup \{A_l\}))(1-p(coarse))$$

2.5 Implementation

We have implemented our algorithms by modifying scalable decision tree frameworks Rainforest [15]. While Rainforest [15] aims to identify a class of tuples efficiently for a large data set, our task here is to identify each tuple. Since there is no class variable associated with the database, we may imagine that each tuple consists of its own unique class. Precisely, we can assume At every leaf node of the partially built tree, a single scan of the database partition associated with that node can be used to score each tuple and simultaneously and incrementally compute $Indg(A_l, D)$ for all facets A_l, and eventually the most promising facet is selected.

For the case where the database is static and the search queries are provided beforehand, our proposed approaches can simply pre-compute the decision trees. However, when the search queries are initiated on-the-fly with a regular SQL-like query, then building faceted search interface would require us to build the tree online (or in realtime). For such cases, instead of building the complete tree immediately, we can stay in sync with the user while she is exploring the partially constructed tree, and build a few "look ahead" nodes at a time. Finally, in the highly dynamic scenario where the database is frequently updated, a simple solution is to persist with the decision tree created at the start of the search, except that if a path through the tree terminates without a tuple being distinguished, the algorithm can then ask the remaining attributes in decreasing order of attribute probability until either the tuple gets distinguished or we run out of attributes. Thus, a fresh construction of the decision tree can be deferred to reasonable intervals, rather than after each update to the database.

3. FACETED SEARCH IN CONJUNCTION WITH RANKING FUNCTIONS

In this section we develop faceted search procedures that can work in conjunction with ranking functions. Given a query Q, a ranking function typically assigns relevance scores $S(Q,t)$ to all selected tuples t, and a ranked-retrieval system will score and return only the top-n' tuples where $n' << n$. Developing ranking functions for database search applications is an active area of research, and ranking functions range from simple distance-based functions to probabilistic models (see [7, 20]). But in this paper we shall treat such ranking functions as "black boxes"; thus our methods are aimed at very general applicability.

Our facet selection algorithm calls one such "black box" ranking function at every node in the decision tree during its construction and uses the ranked scores of the returned tuples as inputs to the facet selection algorithm. However, because the ranking function is a black box, it is challenging to develop methods for facet selection that are theoretically rigorous. In our approaches, we shall make one assumption: that the scores are normalized so that they are (a) positive, and (b) $\sum_{t \text{ selected by } Q} S(Q,t) = 1$. In other words, the ranking function can be imagined as inducing a non-uniform "probability distribution" over the selected tuples, such that $S(Q,t)$ represents the probability that tuple t is preferred by the user. Of course, in the case that scoring functions are derived from probabilistic IR as well as language models, this assumption is justifiable. In the case of more ad-hoc ranking functions (such as distance-based, or vector-space models popular in IR), this assumption is perhaps a stretch. However, other than this specific assumption, we strive to be as principled as possible in our approaches.

From a faceted search perspective the task is to select the facet that directs the user towards the most preferred tuples (according to the ranking function) as efficiently as possible. One interesting complication is that these tuple preferences may change as the faceted search progresses; this is because as new attribute information is provided by the user, the ranking function may re-evaluate the scores of the remaining tuples still in contention. As an example, consider the car buyer who starts her search with an initial query Q = "Mileage = low AND Age = recent AND Car Type = sedan". Suppose a ranking function when applied to such cars ranks cars with good reliability ratings the highest. After this initial query, a faceted search process starts which allows her to drill down further into the query results. But as the faceted search progresses, the buyer could select attribute values that may cause the ranking function to rank the remaining cars differently. For example, if the user also desires a "powerful engine" (i.e., the query has now been extended to Q = "Mileage = low AND Age = recent AND Car Type = sedan AND Engine Power = high") then the ranking function may score cars with top speeds higher over good reliability. Thus a faceted search system in conjunction with a ranking function offers the benefits of focused retrieval as well as drill-down flexibility.

Defining the Cost of a Decision Tree: Given the above discussion, the cost of a specific decision tree T becomes more complicated than the corresponding definition in Section 2 where no ranking function was assumed. Consider a database D selected by an initial query Q, and consider a decision tree T with each tuple of D at its leaves. We will thus derive a formula for $cost(T,Q)$. Note that Q needs to be a parameter in the cost, as the ranking function uses Q to derive preference probabilities for each tuple. Note that in this cost definition we are not considering attribute uncertainties.

Let the root of the tree select the facet A_l. The root partitions D into the sets $D_{x_1}, \ldots, D_{x_{|Dom_l|}}$ where D_{x_q} is the set that satisfies the query $Q \wedge (A_l = x_q)$ for each $x_q \in Dom_l$. Let the corresponding subtrees for each of these partitions be $T_{x_1}, \ldots, T_{x_{|Dom_l|}}$. Clearly $cost(T_{x_q}, Q \wedge (A_l = x_q))$ is the (recursive) cost of each subtree. The quantity $\sum_{t \in D_{x_q}} S(Q,t)$ is the cumulative probabilities of all tuples in D_{x_q} and represents the probability that when the user is at the root, she will prefer any of the tuples in D_{x_q}. Thus we have

$$cost(T,Q) = \sum_{x_q \in Dom_l} \sum_{t \in D_{x_q}} S(Q,t) \times (cost(T_{x_q}, Q \wedge A_l = x_q) + 1) \quad (3)$$

It is easy to see that if no ranking functions are assumed, i.e., each tuple is uniformly preferred by the user, the cost of a tree reduces to the definition in Section 2, i.e., $\sum_{t \in D} ht(t)/n$. Our task is then the following: *Given an initial query Q that selects a set of tuples D, to determine a tree T such that $cost(T, Q)$ is minimized.* Since the problem is NP-Hard even without a ranking function, this problem too is intractable.

3.1 Facet Selection Algorithms

We develop a greedy heuristic that is motivated by our facet selection approaches presented in Section 2. Assume that we are at a particular node v of the decision tree. Let Q be the current query at that node. Thus Q is the initial query at the root, concatenated (i.e., AND'ed) with all conditions along the path from the root to v. Let D be the set of tuples of the database that satisfy Q. For any attribute A_l we can define a function $Indg(A_l, D)$ as follows:

$$Indg(A_l, D) = \sum_{x_q \in Dom_l} \left(\sum_{t_i, t_j \in D_{x_q}, i<j} S(Q, t_i) \times S(Q, t_j) \right) \quad (4)$$

The rest of the algorithm for selecting a single facet, even considering attribute uncertainty, is exactly the same as in Algorithm 1, except that Line 12 of Algorithm 1 is replaced selecting the attribute A_l that minimizes the expected value of Equation 4. The extensions to selecting k-facets, or building a fixed k-facet interface are similarly straightforward, and details are omitted from this version of the paper.

3.2 Comparing Against Other Attribute Selection Procedures

In a recent paper [18], algorithms were described that automatically select attributes of the results of a ranking query. Several selection criteria were examined, with the overall objective of attempting to select attributes that are most "useful" to the user. Attributes are consider most useful if, when the database is projected only on these attributes, the ranking function will re-rank the tuples in almost the same order. By listing the useful attributes, the motivation was to provide the end user the reasons why the top tuples were ranked so high. While such attribute selection algorithms can be used for faceted search, the following lemma shows that they do not necessarily achieve our goal of minimizing effort during the drill-down process.

LEMMA 3.1. *Given a query Q that selects a set of tuples D, and a scoring function $S()$, the decision tree constructed by selecting facets that minimize $Indg()$ may be different from the decision tree constructed by selecting facets according to the* Score-Based *and* Rank-Based *attribute selection algorithms in [18].*

Proof (sketch): We sketch the proof by describing an example. Consider a cars database, and assume a ranking function exists, such that when a user poses an initial query for cars available in Texas, it ranks cars with air-conditioners very high. The ranking function assigns scores of 1 to the latter cars, and 0 to the rest. Both the Score-Based and Rank-Based algorithms in [18] will select the Boolean attribute AirCon as the most influential attribute. However, our minimum effort driven approach would not prefer to select the AirCon attribute. This is because all cars with air-conditioners will have high scores and will group together to produce a rather high value for $Indg(AirCon)$. In contrast, consider another attribute such as AutoTrans, which splits the total tuple set such that the highly ranked cars are evenly divided into each group. It is easy to see that $Indg(AutoTrans)$ is smaller than $Indg(AirCon)$ and hence more preferable. □

Basically the attribute AirCon does not really help in further narrowing down the highly ranked tuples, because of the correlation with Texas cars via the ranking function. Offering some other facet such as AutoTrans will help the user narrow down the tuples more efficiently. Our experiments corroborate this observation in general.

3.3 Implementation

Although we assume that we are provided with a black box scoring function $S(Q, t)$, the way such a scoring function is implemented greatly affects the performance of our attribute selection algorithms. We define *single-result interface* for the ranking black box which is supported by previous works [18] on top-k computations. The *single-result interface* $S(Q, t)$ takes as input a query Q and a tuple t and outputs the score of the tuple. This interface incurs unit cost.

Facet Selection using Single Result Interface: A scalable implementation of facet selection (Equation 4) using the single result interface is straightforward using ideas from the Rainforest framework [15]. We point out that even though the definition of $Indg()$ appears to require a quadratic-time algorithm, it can be computed in a single linear scan. The extensions to selecting k-facets as well as designing a fixed k-facet interface are straightforward. The extensions to include attribute uncertainties, k-facet selection, as well as designing a fixed k-facet interface are straightforward and omitted.

4. EVALUATION

In this section we describe our experimental setup, our different results of facet selection algorithms (without and in conjunction with ranking functions) and draw conclusions on the quality and performance of the techniques. We validated the quality of the our solutions by measuring *cost*, which is defined as the average number of user interactions (i.e., number of attributes or facets selected) before the desired tuple is identified. Experiments evaluating the time complexity of the node creation step of our tree building algorithms were also conducted. This measure is especially relevant for exploratory interactive users and hence a fast scalable implementation is desirable. In case the trees can be built in a preprocessing step, this measure is less critical. We also implemented several existing attribute selection techniques to compare against our approaches. Evaluation results clearly show that our solutions perform significantly better.

All implementation is done using Java and C# and the evaluations performed on a Windows XP machine with 3.0Ghz Intel Xeon processor and 2GB RAM.

Database Used: We evaluated our methods using two data sets, *IMDB movie database*[2] - a real world movie database accessible over the internet and *Yahoo Autos*[3], a online used-car listing database. Using the IMDB database, we generated a single movie database containing about 234,000 tu-

[2]http://www.imdb.com
[3]http://autos.yahoo.com

ples with 19 attributes including null values in some fields. Similarly, we built a car database with 43 attributes and more than 40,000 tuples. We also generated a large synthetic dataset having nearly 10 million rows and 100 attributes from the car dataset by maintaining the original distribution of the dataset.

Uncertainty Model: As we discussed in Section 2.2, we use external knowledge about user uncertainty for ranking the attributes of our databases. For our evaluation, we organized a small survey among 20 randomly selected users comprising students and faculty members. In the survey, each person was asked to assign a value (between 0 to 1) for each attribute in the IMDB movie database. This value denotes the likelihood (probability) with which the user thinks she would be able to answer a question over that attribute. We took average probability scores for all attributes in our evalution. Note that the question of whether the survey accurately reflects the true uncertainty model for the user population at large is an orthogonal problem, and is not extremely relevant for our purposes. The survey was conducted merely to obtain uncertainty values that are somewhat realistic for the related domain. Developing techniques for ascertaining uncertainty values is a future direction of research.

4.1 Experiments on Faceted Search Without Ranking Function

In this set of experiments all tuples were considered equally desirable to the end user as no ranking function was assumed. We conducted evaluations to check the quality and robustness of the algorithms we developed.

4.1.1 Quality Evaluation

In this subsection, we briefly explain the three different quality experiments we performed and draw inferences. These experiments measure cost as defined in Section 3, which is the average number of queries that needs to be answered before the user arrives at a desired tuple (i.e., effort).

Cost versus varying attribute probability: The intrinsic assumption in our decision tree modeling is the user's inability to answer all the questions. This experiment infers the influence of the probability of an attribute in determining cost.

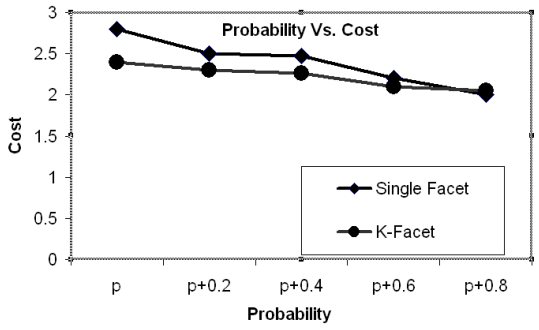


Figure 3: Change Of cost with varying probability

As shown in Figure 3, we compare the cost of the Single-Facet search with the k-Facets based search algorithm by varying the uncertainty model. In our evaluation we set the value of k top = 2. We varied the probability of each attribute in increments of 0.2 in this experiment. As the graph suggests, with higher probability, the cost decreases for both the algorithms. This observation corroborates our basic intuition of considering probability of the attributes in the decision tree construction.

Cost versus varying database size: In this set of experiments, we vary the database size (auto database) and compare the costs of the Single-Facet and the k-Facets based search algorithms. As can be seen from Figure 4, the cost is more for the Single-Facet algorithm. Also, in both cases, costs increase with increasing database size. The reason being, with an increase in the number of tuples, more questions are needed to distinguish them.

Algorithm Name	Data Size 10000	Data Size 12000	Data Size 15000	Data Size 20000	Data Size 25000
Single Facet Based Method	3.52	3.82	4.36	4.78	5.02
K-Facet (K=2) Based Method	2.15	2.5	2.77	2.93	3.08
Ambiguous Method	3.4	3.76	4.3	4.5	4.89
ID3 Classification	7.6	9.2	9.96	11.23	12.78
Categorical PCA	5.2	5.9	6.9	7.7	8.3

Figure 4: Change of cost with varying database size

Comparing against existing techniques: We compared the cost of facets suggested by our methods with that suggested using the *Indg()* method developed in [2] (named as Ambigous method in Figure 4, PCA for categorical data and the $ID3$ classification algorithm. Note that these three algorithms assume all values are known to the user and so do not have any provision for handling *uncertainty* in user response. Hence, these techniques are principally different from our facet selection algorithms. Howeve, from Figure 4, it is clear that the Indg() based method clearly outperforms the ID3 and PCA. Since, both Single-Facet and K-Facet algorithms are richer than Indg() and also show better performance, we can therefore claim that our techniques are better than existing facet (attribute) selection methods.

4.2 Performance Evaluation

We implemented the scalable Rainforest [15] framework to construct the decision tree. We vary two parameters (number of tuples and number of attributes) and measure the average node creation time. As seen from the Figure 6 and Figure 5, average node creation time increases with the increase of dataset size/ width. We point out that that the objective of our decision tree is to distinguish each tuple (in contrast to identifying a class of tuples). Hence, the depth of this tree is much larger than the normal decision trees used for classification problems. Consequently, this leads to a proportional increase in creation time.

4.3 Experiments on Faceted Search in Conjunction with Ranking Function

In this section, we explain our experiments on facet selection algorithms in conjunction with ranking functions. We assume the presence of a "black box" ranking function which simply contributes skewness towards the preference

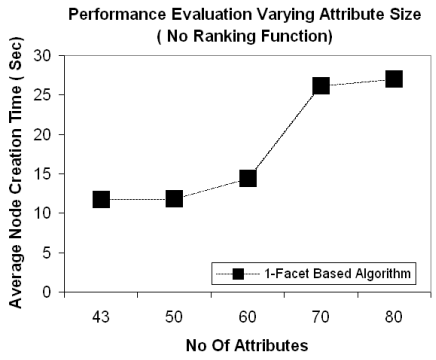


Figure 5: Change of average node creation time with varying number of attributes

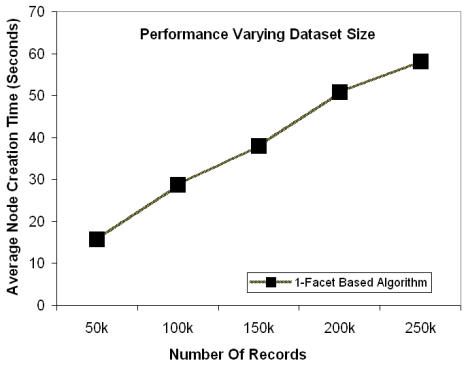


Figure 6: Change of average node creation time with varying number of tuples

of tuples. Consequently, the solution cost is computed as described in Section 3.

Ranking Function: Design of an efficient and effective ranking function is an orthogonal research problem and is not our focus here. For practical purposes, however, we implement a simple ranking function where a tuple t gets a score equal to the square of its Euclidian distance from the centroid of the residual database partition. We further normalize this squared distance to a non-uniform probability distribution over the selected tuples, such that $S(Q,t)$ represents the probability that tuple t is preferred by the user, and that $\sum_{t \text{ selected by } Q} S(Q,t) = 1$.

4.3.1 Quality Evaluation

In this experiment, we compared the cost of our Single Interface algorithm with the existing rank based attribute selection technique [18] on IMDB data. As seen from the Figure 7, our Single Interface facet selection technique performs better than existing approach.

4.3.2 Performance Evaluation

We measure performance is in terms of the average node creation time. We varied the database size and observed the performance of our face selection algorithm using Single Result Interface. As seen in Figure 8, average node creation time increases with the increase in database size. The reason is more tuples need to be processed to make attribute selection decision as size of the database increases.

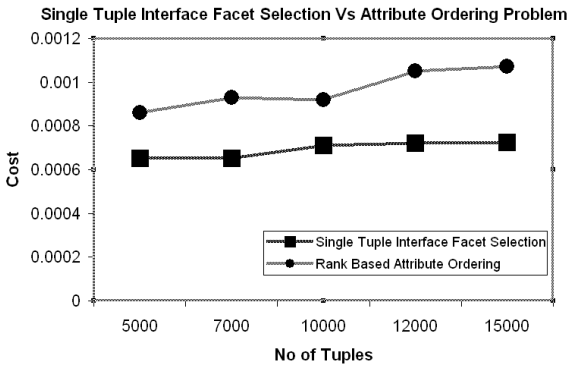


Figure 7: Comparison of Cost between Facet Selection and Attribute Ordering Problem

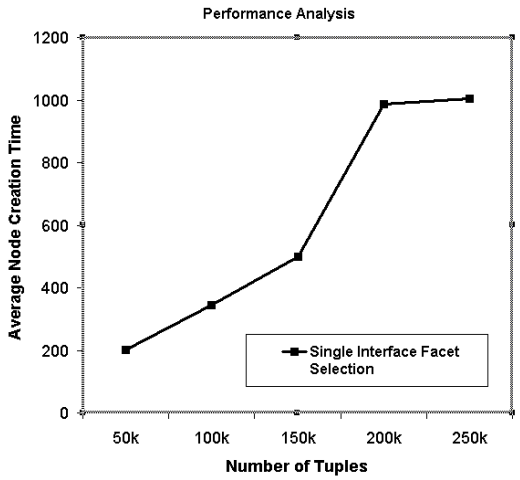


Figure 8: Average Node Creation Time Varying Dataset Size

5. RELATED WORK

The traditional design goal of faceted search interfaces [3, 4, 6, 24, 25] is to offer users a flexible navigational structure, targeted towards text and/or image data. There have been recent efforts at creating a faceted search interface over structured database, e.g., [16], as well as heterogeneous collections [23]. The former is typically designed for specific applications by domain experts. In our work, we aim to propose a domain independent solution for automatically generating facets. In [23], the focus is on computing correlated facets and using them to aggregate and present related information to the user. This appears to be different from our problem, where the focus is minimum effort drill-down.

Our work bears resemblance to the problem of generating automatic categorization of query results [1]. Our developed approach differs from this prior work along several key dimensions: (a) our proposed approach considers uncertainty models, (b) our approach is decision-tree based and depends on user interaction, and (c) our algorithms can work in conjunction with available ranking functions.

Decision trees and classical *Information Value Theory* [10] are widely studied class of techniques in machine learning [19]. However such models require explicit knowledge of

each of the user decision models which is not present in our model. A recent work [2] uses decision trees for fast tuple identification in databases. Our proposed decision tree model captures user inability to answer certain attributes as well as the ability to incorporate ranking functions, which marks the intrinsic difference between our approach and [2].

Dimensionality reduction techniques aim at mapping high dimensional data to lower dimensional data, while preserving some metrics such as distances to the best possible extent (e.g., (PCA) [17]). We have attempted a mapping of the key ideas of PCA to categorical data, and have compared it against other approaches for selecting facets.

Ranked retrieval in structured databases is an active research area [7, 12, 13, 8]. Recent research effort address the problem of keyword-based search techniques in databases combined with the power of aggregation in Online Analytical Processing(OLAP) systems [5]. This ranking metric is based upon "interestingness" of attributes which is different from our effort-based strategy.

In [18], algorithms were described that automatically select attributes of the results of a ranking query. As discussed in this paper, while such attribute selection algorithms can be used for faceted search, they do not necessarily achieve our minimum effort goals.

Selecting the next facet based on a ranking function has connections with automatic query expansion (AQE) studies in IR ([21, 22]. At some level automatic facet selection may be viewed as a similar problem, however while AQE techniques are largely empirical and target text collections, we make several new and important contributions involving structured data, black box ranking functions, as well as scalable algorithms based on modern top-k concepts.

Our fixed k-facets interface design problem has similarities with the classical problem of computing minimum and approximate keys and functional dependencies of database relations (see [14, 11]). Most problem variants are NP-complete, and popular algorithms are based on level-wise methods from data mining ([11]). However, in our case the problem is complicated by the fact that attributes are associated with uncertainties, thus such deterministic procedures appear difficult to generalize to the probabilistic case.

6. CONCLUSION

In this paper we tackle the problem of building faceted search interfaces over enterprise data warehouses for providing a much needed minimal effort entity (tuples) navigation solution. Our proposed technique shows facets based on their ability to rapidly drill down to the most promising tuples, as well as the ability of the user to provide desired values for them. We also provide solutions that can consider bias over the tuple introduced by any ranking function. We provide scalable and efficient implementations of our solutions and present results that show the efficiency and robustness of our solutions. Future directions include extending the techniques to work with multi-table databases and developing faceted interfaces that span both structured and unstructured data sources.

7. REFERENCES

[1] K. Chakrabarti, S. Chaudhuri and S. Hwang. *Automatic Categorization Of Query Results.* SIGMOD 2004.

[2] V. T.Chakravarthy, V. Pandit, S. Roy, P. Awasthi and M. Mohania. *Decision Trees for Entity Identification: Approximation Algorithms and Hardness Results.* PODS 2007.

[3] J. English, M. Hearst, R. Sinha, K. Swearingen and P. Yee. *Hierarchical Faceted Metadata in Site Search Interfaces. CHI Conference Companion* 2002.

[4] W. Dakka, P. G. Ipeirotis and K. R. Wood. *Faceted Browsing over Large Databases of Text-Annotated Objects.* ICDE 2007.

[5] P. Wu, Y. Sismanis and B. Reinwald. *Towards Keyword-Driven Analytical Processing.* SIGMOD 2007.

[6] E. Stoica, M. Hearst and M. Richardson. *Automating Creation of Hierarchical Faceted Metadata Structures. In the proceedings of NAACL-HLT* 2007.

[7] S. Chaudhuri, G. Das, V. Hristidis and G. Weikum. *Probabilistic information retrieval approach for ranking of database query results. ACM Trans. Database Syst*, 31(3): 1134–1168.

[8] S. Agrawal, S. Chaudhuri and G. Das. *DBXplorer: enabling keyword search over relational databases.* SIGMOD 2002.

[9] J. Han and M. Kamber. *Data Mining: Concepts and Techniques.* The Morgan Kaufmann Series 2006.

[10] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall Series 2003.

[11] Y. Huhtala, J. Krkkinen, P. Porkka, and H. Toivonen. *Efficient discovery of functional and approximate dependencies using partitions.* ICDE 1998.

[12] V. Hristidis, Y. Papakonstantinou. *DISCOVER: Keyword Search in Relational Databases.* VLDB 2002.

[13] B. Aditya, G. Bhalotia, S. Chakrabarti, A. Hulgeri, C. Nakhe, P. S. Sudarshan. *BANKS: Browsing and Keyword Searching in Relational Databases.* ICDE 2003.

[14] C. L. Lucchesi and S. L. Osborn. *Candidate Keys for Relations. J. Comput. Syst. Sci.*, 17(2): 1978.

[15] J. Gehrke, R. Ramakrishnan and V. Ganti. *RainForest - A Framework for Fast Decision Tree Construction of Large Datasets. DMKD* 2000.

[16] *http://www.l3s.de/growbag/demonstrators.php.*

[17] J. Shlens. *A Tutorial on Principal Component Analysis.* Institute for Nonlinear Science, UCSD, 2005.

[18] G. Das, V. Hristidis, N. Kapoor, S. Sudarshan. *Ordering the Atributes of Query Results.* SIGMOD 2006.

[19] Tom Mitchell. *Machine Learning.* McGraw Hill 1997.

[20] S. Agrawal, S. Chaudhuri, G. Das, A. Gionis. *Automated Ranking of Database Query Results.* CIDR 2003.

[21] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison Wesley 1999.

[22] E. N. Efthimiadis. *Query Expansion.* Annual Review of Information Systems and Technology 1996.

[23] O. Ben-Yitzhak et al. *Beyond Basic Faceted Search.* WSDM 2008.

[24] W. Dakka, P. G. Ipeirotis, and K. R. Wood. *Automatic construction of multifaceted browsing interfaces.* CIKM 2005.

[25] I. Martin and J. Jose. *A personalised information retrieval tool.* SIGIR 2003.