

ReDRIVE: Result-Driven Database Exploration through Recommendations

Marina Drosou^{*}
Computer Science Department
University of Ioannina, Greece
mdrosou@cs.uoi.gr

Evaggelia Pitoura
Computer Science Department
University of Ioannina, Greece
pitoura@cs.uoi.gr

ABSTRACT

Typically, users interact with database systems by formulating queries. However, many times users do not have a clear understanding of their information needs or the exact content of the database, thus, their queries are of an exploratory nature. In this paper, we propose assisting users in database exploration by recommending to them additional items that are highly related with the items in the result of their original query. Such items are computed based on the most interesting sets of attribute values (or faSets) that appear in the result of the original user query. The interestingness of a faSet is defined based on its frequency both in the query result and in the database instance. Database frequency estimations rely on a novel approach that employs an ϵ -tolerance closed rare faSets representation. We report evaluation results of the efficiency and effectiveness of our approach on both real and synthetic datasets.

Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—*Query formulation, Search process*

General Terms

Algorithms, Experimentation, Design, Performance

1. INTRODUCTION

Typically, users interact with a database system by formulating queries. This interaction mode assumes that users are to some extent familiar with the content of the database and also have a clear understanding of their information needs. However, as databases get larger and become accessible to a more diverse and less technically-oriented audience, exploration or recommendation style database interactions seem attractive and useful.

A step towards this direction is offered by facet queries that provide a form of navigational search, where users restrict their results by selecting interesting facets of the original results (e.g., [11]). With facet search, users start with a general query and progressively narrow its results down to

^{*}Supported by the research program “HRAKLEITOS II” co-funded by the European Union and National Sources.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIKM’11, October 24–28, 2011, Glasgow, Scotland, UK.
Copyright 2011 ACM 978-1-4503-0717-8/11/10 ...\$10.00.

a specific item. Other related research includes addressing the *many-* or *empty-* answers problems. Approaches to the many-answers problem range from reformulating the original query so as to restrict the size of its result (for example, by adding additional constraints to it (e.g., [15]) to automatically ranking the query results and presenting to the user only the top- k most highly ranked among them (e.g., [7]). The empty-answers problem is commonly handled by relaxing the original query (e.g., [12]).

In this paper, we propose a novel exploration mode of interaction: we present to the users additional items which, although not part of the answer of their original query, may be of interest to them. This way users see information that they may be unaware that exists. For instance, when asking for movies directed by F.F. Coppola, we guide exploration by recommending movies by other directors that have directed movies similar to those of F.F. Coppola, i.e., with similar characteristics, such as, genre or production year. We also consider expanding the original query with additional attributes, by finding correlations with other relations. For example, when asking for the title of a movie, we also look into its genre or other characteristics.

The computation of recommended results is based on the most interesting sets of (attribute, value) pairs, called faSets, that appear in the result of the original user query. The *interestingness* of a faSet expresses how unexpected it is to see this faSet in the result. The computation of interestingness is based on the frequency of the faSet both in the user query result and in the database instance. Since computing the frequencies of faSets in the database instance on-line has prohibitively high cost, we opt to maintain statistics that allow us to *estimate* those frequencies when needed. More specifically, we propose a novel approach that is based on storing an ϵ -tolerance closed rare faSets representation as a summary of such frequencies and exploit these summaries to estimate the interestingness of the faSets that appear in the result of any given user query.

We also present a two-phase algorithm for computing the top- k faSets. In the first phase, the algorithm uses the pre-computed statistics to set a frequency threshold that is then used to run a frequent itemset based algorithm on the result of the query. We evaluate the performance of our approach using both real and synthetic datasets.

The rest of this paper is organized as follows. Sec. 2 presents the overall framework, Sec. 3 its implementation and Sec. 4 an experimental evaluation. Finally, related work is presented in Sec. 5 and conclusions are offered in Sec. 6.

2. THE ReDRIVE FRAMEWORK

Let \mathcal{D} be a relational database with n relations $\mathcal{R} = \{R_1, \dots, R_n\}$ and \mathcal{A} be the set of all attributes in \mathcal{R} . Without loss of generality, we assume that relation and attribute names are distinct. To locate items of interest, users pose queries.

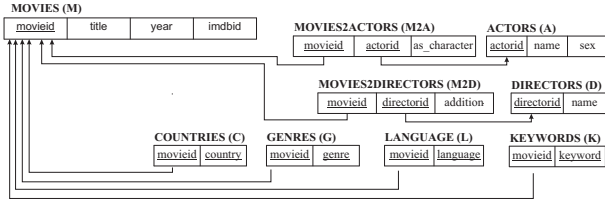


Figure 1: Movies database schema.

In particular, we consider Select-Project-Join (SPJ) queries of the following form:

```
SELECT proj(Q)
FROM rel(Q)
WHERE sel(Q) AND join(Q)
```

where $rel(Q)$ is a set of relations, $sel(Q)$ is a conjunction of selection conditions, $join(Q)$ is a set of join conditions among the relations in $rel(Q)$ and $proj(Q)$ is the set of projected attributes. For simplicity, we shall focus on *equality* conditions, i.e., $sel(Q) = (A_1 = a_1) \wedge \dots \wedge (A_m = a_m)$, $m \geq 1$, where $A_i \in \mathcal{A}$ and $a_i \in domain(A_i)$. The *result set*, $Res(Q)$, of a query Q is a relation with schema $proj(Q)$.

Since users must specify in their queries the conditions that the searched items need to satisfy, they must have a somewhat clear understanding of the information they are seeking. In this paper, we propose an exploratory way of discovering interesting information based on identifying potentially interesting pieces of information based on the initial result set and then using these pieces to explore the database further by recommending additional results to the users.

2.1 Interesting FaSets

Let us first define pieces of information in the result set:

DEFINITION 1 (FACET AND m-FASET). A facet condition, or simply facet, is a condition of the form $(A_i = a_i)$, where $A_i \in \mathcal{A}$ and $a_i \in domain(A_i)$. An m -set of facets or m -faSet, $m \geq 1$, is a set of m facet conditions on m different attributes.

We shall also use the term faSet when the size of the m -faSet is not of interest.

For a faSet f , we use $Att(f)$ to denote its attributes. Let t be a tuple from a set of tuples S with schema R ; we say that t satisfies a faSet f , where $Att(f) \subseteq R$, if $t[A_i] = a_i$, for all facets $(A_i = a_i) \in f$. We call the percentage of tuples in S that satisfy f , *support* of f in S . In the following, we use the term faSet to mean both the conditions and the list of the associated values appearing in the conditions.

Example: Consider the movies database in Fig. 1 and the query and its corresponding result set in Fig. 2. Then $\{G.genre = \text{"Drama"}\}$ or simply $\{\text{"Drama"}\}$ is a 1-faSet and $\{M.year = \text{"1972"}, G.genre = \text{"Drama"}\}$ or simply $\{\text{"1972"}, \text{"Drama"}\}$ is a 2-faSet.

We are looking for interesting pieces of information at the granularity of a faSet: this may be the value of a single attribute (i.e., a 1-faSet) or the values of m attributes (i.e., an m -faSet).

Example: Consider the example in Fig. 2, where a user poses a query to retrieve movies directed by F.F. Coppola. $\{\text{"Drama"}\}$ is a 1-faSet in the result that is likely to interest the user, since it is associated with many of the movies directed by F.F. Coppola. The same holds for the 2-faSet $\{\text{"1983"}, \text{"Drama"}\}$.

To define faSet relevance formally, we take an IR-based approach and rank faSets in decreasing order of their odds of being relevant to a user information need. For a user information need u_Q expressed through a query Q , let R_{u_Q} be the set of tuples that are relevant to u_Q and $\overline{R_{u_Q}}$ be the

```
SELECT D.name, M.title, M.year, G.genre
FROM D, M2D, M, Gs
WHERE D.name = "F. F. Coppola"
AND D.directorid = M2D.directorid
AND M2D.movieid = M.movieid
AND M.movieid = G.genreid
```

D.name	M.title	M.year	G.genre
F. F. Coppola	The Godfather III	1990	Drama
F. F. Coppola	The Rainmake	1997	Crime
F. F. Coppola	The Godfather	1972	Drama
F. F. Coppola	Rumble Fish	1983	Drama
F. F. Coppola	The Conversation	1974	Thriller
F. F. Coppola	The Outsiders	1983	Drama
F. F. Coppola	Supernova	2000	Thriller
F. F. Coppola	Apocalypse Now	1979	Drama

Figure 2: Example query and result set.

set of tuples that are not relevant to u_Q . Then, the relevance score of a faSet f for u_Q is defined as:

$$\frac{p(R_{u_Q}|f)}{p(\overline{R_{u_Q}}|f)}$$

where $p(R_{u_Q}|f)$ (resp. $p(\overline{R_{u_Q}}|f)$) is the probability that a tuple satisfying f is relevant (resp. not relevant) to u_Q . Using the Bayes rule we get:

$$\frac{p(f|R_{u_Q})p(R_{u_Q})}{p(f|\overline{R_{u_Q}})p(\overline{R_{u_Q}})}$$

Since $p(R_{u_Q})$ and $p(\overline{R_{u_Q}})$ have the same value for all faSets, and thus do not affect their ranking, they can be ignored.

We make the assumption that all relevant to u_Q results are those that appear in $Res(Q)$, thus $p(f|R_{u_Q})$ is equal with the probability that f is satisfied by a tuple in the result set, written $p(f|Res(Q))$. Similarly, $p(f|\overline{R_{u_Q}})$ is the probability that f is satisfied by a tuple that is not relevant, that is, a tuple that does not belong to the result set. We make the logical assumption that the result set is small in comparison with the size of the database, and approximate the non-relevant tuples with all tuples in the database, that is, all tuples in the global relation, denoted by \mathcal{D} , with schema \mathcal{A} . Based on the above motivation, we provide the following definition for the relevance of a faSet:

DEFINITION 2 (INTERESTINGNESS SCORE). Let Q be a query and f be a faSet with $Att(f) \subseteq proj(Q)$. The *interestingness score*, $score(f, Q)$, of f for Q is defined as:

$$score(f, Q) = \frac{p(f|Res(Q))}{p(f|\mathcal{D})}$$

The term $p(f|Res(Q))$ is estimated by the support of f in $Res(Q)$, that is, the percentage of tuples in the result set that satisfy f . The term $p(f|\mathcal{D})$ is a global measure that does not depend on the query. It serves as an indication of how frequent the faSet is in the whole dataset, i.e., it measures the discriminative power of f .

Example: In the example in Fig. 2, $\{\text{"Drama"}\}$ appears more frequently than $\{\text{"Thriller"}\}$ in the result set. However, if $\{\text{"Thriller"}\}$ appears only a handful of times in the database, then it would be considered more interesting than $\{\text{"Drama"}\}$.

In general, a faSet stands out when it appears more frequently in $Res(Q)$ than expected. Clearly, the $sel(Q)$ part of a query is also a faSet. Therefore, another way of interpreting the interestingness score of f for Q is as the confidence of the association rule: $sel(Q) \rightarrow f$. High confidence indicates a strong dependency of the faSet f on the selection conditions of Q .

Finally, note that in particular, for a faSet f with $Att(f) \subseteq Att(sel(Q))$, that is, for a faSet that includes only attributes whose values are specified in the selection conditions, it holds that $score(f, Q) \geq 1$, since $p(f|Res(Q)) = 1$.

Attribute Expansion: Def. 2 provides a means of ranking the various faSets that appear in the result set of a query Q and discovering the most interesting among them. However, there may be interesting faSets that include attributes

not in $proj(Q)$ and thus do not appear in $Res(Q)$. For example take a query Q that just returns the titles of movies directed by F.F. Coppola. All faSets appear only once in the result set of Q . However, including for instance the relation “Countries” in $rel(Q)$ (and modifying $join(Q)$ accordingly) may disclose interesting information, e.g., that many of the movies directed by F.F. Coppola are related to Romania.

To this end, we extend the definition of interestingness to include faSets with attributes not in $proj(Q)$, by introducing an extended query Q' with the same $sel(Q')$ as the original query Q but with additional attributes in $proj(Q')$ and additional relations in $rel(Q')$.

DEFINITION 3 (EXTENDED INTERESTINGNESS SCORE). Let Q be a query and f be a faSet with $Att(f) \subseteq \mathcal{A}$. The interestingness score of f for Q is equal to:

$$score(f, Q) = \frac{p(f|Res(Q'))}{p(f|\mathcal{D})}$$

where Q' is an SPJ query with $proj(Q') = proj(Q) \cup Att(f)$, $rel(Q') = rel(Q) \cup \{R' \mid A_i \in R', \text{ for } A_i \in Att(f)\}$, $sel(Q') = sel(Q)$ and $join(Q') = join(Q) \wedge (\text{joins with } \{R' \mid A_i \in R', \text{ for } A_i \in Att(f)\})$.

2.2 Exploratory Queries

Besides locating interesting faSets, we also use interesting faSets to discover additional pieces of data that are potentially related to the user needs. In particular, we aim at constructing exploratory queries that retrieve results strongly correlated with those of the original user query Q by replacing the selection conditions, $sel(Q)$, of Q with related ones. Recall that a high interestingness score for f means that the confidence of $sel(Q) \rightarrow f$ is high, indicating replacing $sel(Q)$ with f , since $sel(Q)$ seems to impose f .

For example, by replacing $sel(Q)$ of Q in Fig. 2 with its interesting faSet $\{“Drama”\}$, we get the exploratory query:

```
SELECT D.name
FROM D, M2D, M, G
WHERE G.genre = 'Drama'
AND D.directorid = M2D.directorid
AND M2D.movieid = M.movieid
AND M.movieid = G.movieid
```

which retrieves other directors that have also directed drama movies, which is an interesting value appearing in the original query result set.

DEFINITION 4 (EXPLORATORY QUERY). Let Q be a user query and f be an interesting faSet for Q . The exploratory query \hat{Q} that uses f is an SPJ query with $proj(\hat{Q}) = Attr(sel(Q))$, $rel(\hat{Q}) = rel(Q) \cup \{R' \mid A_i \in R', \text{ for } A_i \in Att(f)\}$, $sel(\hat{Q}) = f \wedge \neg sel(Q)$ and $join(\hat{Q}) = join(Q) \wedge (\text{joins with } \{R' \mid A_i \in R', \text{ for } A_i \in Att(f)\})$.

Then, interesting faSets for the exploratory \hat{Q} are recommended to the user. Clearly, one can use the interesting faSets in the results of an exploratory query to construct other exploratory queries. This way, users may start with an initial query Q and gradually discover other interesting information in the database through results attained by applying exploratory queries progressively.

Framework Overview: In summary, ReDRIVE database exploration works as follows. Given a query Q , the top- k most interesting faSets for Q are computed and presented to the users. Such faSets may be either interesting pieces (sub-tuples) of the tuples in the result set of Q or extended tuples that include additional attributes not in the original result. Interesting faSets are further used to construct exploratory queries that lead to discovering additional information related to the initial user query. This process may be repeated for each exploratory query.

3. TOP-K FASETS COMPUTATION

In this section, we present algorithms for finding interesting faSets. In particular, first, we present an approach to maintaining statistics for estimating $p(f|\mathcal{D})$ for each faSet f . Then, we present a two-phase algorithm for computing the top- k most interesting faSets for a query Q . Proofs omitted can be found in [9].

3.1 Estimation of $p(f|\mathcal{D})$

To compute the interestingness of a faSet f for a query Q according to Def. 2 (resp. Def. 3), we need to compute two quantities: $p(f|Res(Q))$ (resp. $p(f|Res(Q'))$) and $p(f|\mathcal{D})$. Whereas $p(f|Res(Q))$ (resp. $p(f|Res(Q'))$) is different for each Q , and thus needs to be computed on-line, $p(f|\mathcal{D})$ is the same for all user queries. The straightforward approach would be to also compute $p(f|\mathcal{D})$ on-line for each query Q , by counting for each examined faSet f the number of database tuples that satisfy it. Given the large number of such faSets and the size of the database, the cost of such computations may become prohibitively expensive. We propose pre-computing and storing some form of information that will allow us to estimate $p(f|\mathcal{D})$, thus avoiding such on-line computations. Next, we discuss available options.

Let m_{max} be the maximum number of projected attributes of any user query, i.e., $m_{max} = |\mathcal{A}|$. An exhaustive approach would be to generate all possible faSets of size up to m_{max} and pre-compute their support in \mathcal{D} . Such an approach, however, is infeasible even for small databases due to the exponential number of possible faSets.

A first approach is to pre-compute and store the support of all 1-faSets in \mathcal{D} . Then, assuming that facet conditions are satisfied independently from each other, the support of a higher-order m -faSet is equal to:

$$p(f|\mathcal{D}) = p(\{A_1 = a_1, \dots, A_m = a_m\}|\mathcal{D}) = \prod_{i=1}^m p(\{A_i = a_i\}|\mathcal{D})$$

This approach requires maintaining information for a relatively small number of faSets, i.e., $\sum_{A_i \in \mathcal{A}} |domain(A_i)|$ faSets. However, the independence assumption is unrealistic in real-world applications.

Now, let us assume that we maintain the support of all faSets up to size ℓ . In this case, we can attain a more accurate estimation of the support of a high-order m -faSet f , $m > \ell$, using a more sophisticated method such as Iterative Proportional Fitting (IPF) [4] used in [15]. The supports of low-order faSets provide some knowledge for the distribution of the supports of high-order faSets. IPF is based on the Principle of Maximum Entropy, which states that, since there is no reason to bias the estimated distribution towards any specific form, then the estimation should be as close to the uniform distribution as possible. The maximum entropy criterion smooths the estimated distribution. This may lead to the loss of interesting information. Consider for example that the faSets $\{G.genre = “Sci-Fi”\}$, $\{M.year = “2000”\}$, $\{M.year = “2005”\}$ have similar supports, while the supports of $\{G.genre = “Sci-Fi”, M.year = “2000”\}$ and $\{G.genre = “Sci-Fi”, M.year = “2005”\}$ differ a lot. IPF (for $\ell = 1$) will estimate similar values for these two faSets.

Maintaining ϵ -Tolerance Closed Rare FaSets.

We propose a different form of statistics aiming at capturing fluctuations in the support of related faSets. To do this, we extend the notion of δ -tolerance frequent itemsets [8] and define ϵ -tolerance closed rare faSets.

Preliminaries: In data mining, the term *itemset* refers to a set of items. An itemset is said to be *frequent* in a dataset if its frequency is above a specific threshold. Otherwise it is called *rare*. Clearly, an m -faSet is an itemset whose items are facets. An itemset $\{A_1 = a_1, \dots, A_m = a_m\}$ appears

in exactly the same set of tuples that satisfy the m -faSet $f = \{A_1 = a_1, \dots, A_m = a_m\}$. Therefore, we say that a faSet f is *frequent* (resp. *rare* (RF)) for a set of tuples S if its support in S is above (resp. below) a specific threshold. Also, we define a faSet f to be *closed rare* (CRF) for S if it is rare and has no proper subset f' , $f' \subset f$, such that, f' has the same support as f in S . A faSet f is *minimal rare* (MRF) for S , if it is rare and has no subset f' such that f' is rare for S .

Statistics based on ϵ -tolerance: Maintaining the support of a number of representative faSets can assist us in estimating the support of a given faSet f . Generally, faSets that appear frequently in the database \mathcal{D} are not expected to be interesting, even if they appear often in the result of user queries, since this is expected. Therefore, it is useful to maintain information about the support of rare faSets in \mathcal{D} . We use $\text{count}(f, S)$ to denote the absolute number of appearances of a faSet f in a set of tuples S .

If we maintain the MRFs, we can derive all corresponding RFs but not their actual support, while if we keep the CRFs we can retrieve these supports as well. However, the number of CRFs is in practice very large, since any RF that has a distinct support is a CRF. Thus, our goal is to store a tunable amount of rare faSets from which we will be able to retrieve a bounded estimation of the support of a given faSet in the database. This is achieved by relaxing the definition of CRFs to maintain only those RFs whose support differs from their rare subsets based on a threshold ϵ . In particular, we define ϵ -tolerance closed rare faSets (ϵ -CRFs) as follows:

DEFINITION 5 (ϵ -CRF). *A faSet f is called ϵ -CRF for a set of tuples S , if and only if, it is rare for S and it has no proper immediate rare subset f' , i.e., $|f'| = |f| - 1$, such that, $\text{count}(f', S) < (1 + \epsilon) \text{count}(f, S)$, where $\epsilon \geq 0$.*

Intuitively, a rare faSet f is an ϵ -CRF if, even if we increase its count by a constant ϵ , all its subsets still have a larger frequency than f . This means that f has a different frequency from all its subsets and cannot be estimated (or represented) by any of them.

Let us assume that a set of ϵ -CRFs is maintained for some value of ϵ . We denote this set C . An RF f either belongs to C or not. If $f \in C$, then the support of f is stored and its count is readily available. If not, then, according to Def. 5, there is some subset of f that belongs to C whose support is close to that of f . We use $C(f)$ to denote the faSet in C that is the most suitable one to estimate the count of f , i.e., the largest subset of f in C . The following lemma holds:

LEMMA 1. *Let C be a set of ϵ -CRFs for a set of tuples S and f be a faSet, $f \notin C$. Then, there exists f' , $f' \in C$, such that, $\text{count}(f', S) \leq \phi \text{count}(f, S)$, where $\phi = (1 + \epsilon)^{(|f| - |f'|)}$.*

To provide estimations, each ϵ -CRF is stored along with its *frequency extension* defined as follows:

DEFINITION 6 (FREQUENCY EXTENSION). *Let C be a set of ϵ -CRFs for a set of tuples S and f be a faSet in C . Let also $\mathcal{X}(f)$ be the set of all RFs represented in C by f . Then, $X_i(f) = \{x \mid x \in \mathcal{X}(f) \wedge |x| - |f| = i\}$, $1 \leq i \leq m$, where $m = \max\{i \mid X_i(f) \neq \emptyset\}$. The frequency extension of f for i , $1 \leq i \leq m$, is defined as:*

$$\text{ext}(f, i) = \frac{\sum_{x \in X_i(f)} \frac{\text{count}(x, S)}{\text{count}(f, S)}}{|X_i(f)|}$$

Intuitively, the frequency extension of f for i is the average count difference between f and all the faSets that f represents whose size difference from f is equal to i . Given a faSet f , the estimation of $p(f|\mathcal{D})$, denoted $\tilde{p}(f|\mathcal{D})$, is:

$$\tilde{p}(f|\mathcal{D}) = \text{count}(C(f), S) \cdot \text{ext}(C(f), |f| - |C(f)|)$$

It holds that:

LEMMA 2. *Let f be an ϵ -CRF. Then, $\forall i$, it holds that $\frac{1}{\phi} \leq \text{ext}(f, i) \leq 1$, where $\phi = (1 + \epsilon)^i$.*

It can be shown that the estimation error is bounded by ϕ , i.e., by ϵ . More specifically, let f be an RF and $|f| - |C(f)| = i$. The estimation error for $p(f|\mathcal{D})$ is bounded as follows:

$$\frac{1}{\phi} - 1 \leq \frac{\tilde{p}(f|\mathcal{D}) - p(f|\mathcal{D})}{p(f|\mathcal{D})} \leq \phi - 1$$

3.2 The Two-Phase Algorithm

Given a query Q , our goal is to locate the k faSets with the highest interestingness scores. Clearly, the brute-force method of generating all possible faSets in $\text{Res}(Q)$ and computing their score is exponential on the number of distinct values that appear in $\text{Res}(Q)$. Applying an a-priori approach for generating and pruning faSets is not applicable either, since *score* is neither an upwards nor a downwards closed measure. Recall that, a measure d is *upwards closed* if for any two sets S and S' , $S \subseteq S' \Rightarrow d(S) \leq d(S')$ and *downwards closed* if $S \subseteq S' \Rightarrow d(S) \geq d(S')$.

PROPOSITION 1. *Let Q be a query and f a faSet. $\text{score}(f, Q)$ is neither an upwards nor a downwards closed measure.*

This implies that we cannot employ any subset or superset relations among the faSets of $\text{Res}(Q)$ to prune the search space.

As a baseline approach to reduce the number of examined faSets of $\text{Res}(Q)$, we consider only the most frequent faSets of $\text{Res}(Q)$, motivated by the fact that faSets that appear in $\text{Res}(Q)$ frequently are likely to be highly interesting to the user. To this end, we apply an adaptation of a frequent itemset mining algorithm to generate all frequent faSets of $\text{Res}(Q)$, that is, all faSets with support larger than some pre-specified threshold minsupp_f . Then, for each frequent faSet f , we use the maintained statistics to estimate $p(f|\mathcal{D})$ and compute $\text{score}(f, Q)$.

This baseline approach has the problem of being highly dependent on minsupp_f . A large value of minsupp_f may lead to losing some less frequent in the result but very rarely appearing in the database faSets, whereas a small value may result in a very large number of candidate faSets being examined. Therefore, we propose a Two-Phase Algorithm (TPA), described next, that addresses this issue by setting minsupp_f to an appropriate value so that all top- k faSets are located without generating redundant candidates. TPA assumes that the maintained statistics are based on keeping rare faSets of the database \mathcal{D} . Let minsupp_r be the support threshold of the maintained rare faSets.

In the first phase of the algorithm, all facet conditions, or 1-faSets, that appear in $\text{Res}(Q)$ are located. TPA checks which rare faSets of \mathcal{D} , according to the maintained statistics, contain only facet conditions from $\text{Res}(Q)$. Let X be the set of faSets. Then, in one pass of $\text{Res}(Q)$, all faSets of $\text{Res}(Q)$ that are supersets of some faSet in X are generated and their support in $\text{Res}(Q)$ is measured. For each of the located faSets, $\text{score}(f, Q)$ is computed. Let s be the k^{th} highest score among them. TPA sets minsupp_f equal to $s \times \text{minsupp}_r$ and proceeds to the second phase where it executes a frequent itemset mining algorithm with threshold equal to minsupp_f . Any faSet in $\text{Res}(Q)$ less frequent than minsupp_f has score smaller than the k^{th} faSet located in the first phase and thus can be safely ignored. To see this, let f be a faSet examined in the second phase of the algorithm. Since the score of f has not been computed in the first phase, then $p(f|\mathcal{D}) > \text{minsupp}_r$. Therefore, for $\text{score}(f, Q) > s$ to hold, it must be that $p(f|\text{Res}(Q)) > s \times p(f|\mathcal{D})$, i.e., $p(f|\text{Res}(Q)) > s \times \text{minsupp}_r$. TPA is shown in Alg. 1.

Algorithm 1 Two-Phase Algorithm (TPA).

Input: $Q, Res(Q), k, C, minsupp_r$ of C .**Output:** The top- k interesting faSets for Q .

```
1:  $S \leftarrow \emptyset, A \leftarrow$  all 1-faSets of  $Res(Q)$ 
2: for all faSets  $f \in C$  do
3:   if all 1-faSets  $g \subseteq f$  are contained in  $A$  then
4:      $f.score = score(f, Q), S \leftarrow S \cup \{f\}$ 
5:   end if
6: end for
7: for all tuples  $t \in Res(Q)$  do
8:   generate all faSets  $f \subseteq t$ , s.t.  $\exists g \in S$  with  $g \subset f$ 
9:   for all such faSets  $f$  do
10:     $f.score = score(f, Q), S \leftarrow S \cup \{f\}$ 
11:   end for
12: end for
13:  $minsupp_f \leftarrow (k^{th} \text{ highest score in } S) \times minsupp_r$ 
14:  $candidates \leftarrow frequentFaSetMiner(Res(Q), minsupp_f)$ 
15: for all faSets  $f$  in  $candidates$  do
16:    $f.score = score(f, Q), S \leftarrow S \cup \{f\}$ 
17: end for
18: return The  $k$  faSets in  $S$  with the highest scores
```

4. EXPERIMENTAL RESULTS

In this section, we present experimental results of the deployment of our approach. We use two real datasets: (i) “AUTOS”, a single-relation database consisting of 41 characteristics for 15,191 used cars from Yahoo!Auto [2] and (ii) “MOVIES”, a database with 13 relations whose sizes range from around 10,000 to almost 1,000,000 tuples, containing information extracted from the Internet Movie Database [1]. We use also synthetic datasets consisting of a single relation with 5 attributes and 1,000 tuples, taking values generated using a zipf distribution from a 5-value domain.

Statistics Generation: First, we evaluate the proposed method for maintaining statistics based on ϵ -CRFs in terms of (i) storage, measured as the number of stored faSets and (ii) generation time. We report results for all intermediate computation steps, i.e., locating MRFs, RFs and CRFs. We base our implementation for locating MRFs and RFs on the MRG-Exp and Arima algorithms [18] and use an adapted version of the CFI2TCFI algorithm [8] for producing ϵ -CRFs. For a given $minsupp_r$, all MRFs are located. However, locating all RFs is inefficient due to the exponential nature of algorithms such as Apriori. To overcome this, we use a *random walk* based approach [10]: instead of producing all RFs, we produce only a subset of them discovered by random walks initiated at the MRFs.

Table 1(a) reports the number of produced faSets for different values of $minsupp_r$ and ϵ . In the reported results, we kept the number of random walks fixed (equal to 20). As ϵ increases, an ϵ -CRF represents faSets with larger support differences and, thus, the number of maintained statistics decreases. As $minsupp_r$ increases, more faSets of the database are considered to be rare and, thus, the number of maintained faSets should also increase. However, this is not always the case because of the random walks technique employed to retrieve RFs. Note also that the number of ϵ -CRFs is much smaller than the number of RFs, even for values of ϵ as low as 0.5. For comparison, we also report the number of 2-faSets (Table 1(b) with id attributes excluded) that could alternatively be used for estimating frequencies, as discussed in Sec. 3, which is considerably large for the real databases. Finally, in terms of execution time, the main overhead was induced by the stage of generating RFs which can take up many minutes, while all other stages require a couple of seconds for all datasets (see [9] for details).

Estimation Accuracy: Next, we evaluate the accuracy of estimating the support of rare faSets using the support of the stored ϵ -CRFs. For this, we employ our synthetic datasets and probe our statistics to retrieve estimations for

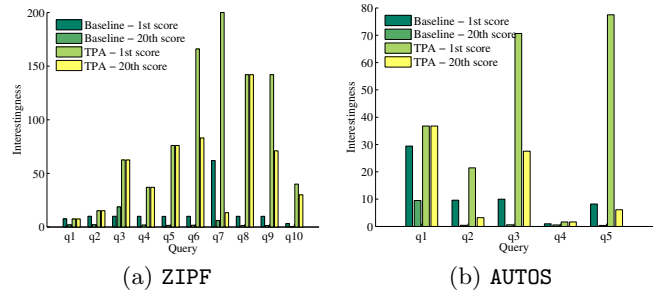


Figure 3: Baseline vs. TPA.

the frequency of 10 rare faSets for each possible size. We observed that, even though we do not maintain the complete set of ϵ -CRFs, because of our random walks approach, the estimation error remains low and below the theoretical bound. For example, for $\epsilon = 0.9$, and $minsupp_r = 10\%$, the absolute error was 3.0 to 4.0 on average. The same holds for the real datasets (see [9] for details).

We also experimented using IPF. This approach turned out to be very inefficient for estimating the frequency of rare faSets. This was mainly due to two reasons. Take for example rare 3-faSets. When they consist of rare 2-faSets, their estimated support was too small (most often equal to zero) due to the maximum entropy principle that tends to consider faSet co-occurrences independent. On the other hand, when rare 3-faSets consist of frequent 2-faSets, their support was over-estimated (often at a ten-fold order).

Top- k FaSet Discovery: We compare the baseline and the Two-Phase (TPA) algorithms. For the synthetic datasets, we generate random queries, while for the real databases we use predefined queries selected so that their result sets include various combinations of rare and frequent faSets. Fig. 3 shows the 1st and 20th highest ranked interestingness score retrieved. For TPA we set $k = 20$ and for the baseline approach we start with a high $minsupp_f$ and gradually decrease it until we get at least 20 results. TPA retrieves more interesting faSets, mainly due to the first phase where rare faSets in $Res(Q)$ are examined. For the reported results ϵ was equal to 0.5. In general, the value of ϵ did not affect the interestingness scores of the top- k results considerably. In most cases TPA located k results during phase one and, thus, phase two was not executed.

Exploring the two Real Databases: We experimented with the exploration of the real databases. Due to space limitations, we just present two of our observations about the acquired results. For example, take a query in the AUTOS database about car models with navigation systems which is a query whose result set includes many rare faSets, all having a high interestingness score, e.g., the car models “Land Rover Discovery II HSE7” and “Mercedes-Benz G55 AMG”. Expanding the query towards the “state” attribute reveals interesting faSets not present in the original result, such as the “Land Rover Range Rover” model in VA and the “Cadillac” model in DE, suggesting that such combinations are highly related with navigation systems but only in specific states. As another example take the query in the MOVIES database for the countries and genres of movies directed by F.F. Coppola. FaSet {“Switzerland”, “Sci-Fi”} is retrieved as the most interesting, since this is extremely rare in the database, and it is very interesting that it was located by the user query. Another highly ranked faSet for this query is {“Romania”} (and its supersets).

5. RELATED WORK

In this paper, we have proposed a novel database exploration model. A common exploration technique is *faceted search* (e.g., [14, 11]), where query results are classified into different multiple categories, or facets, and the user refines

Table 1: Number of FaSets.

minsupp _r	# MRFs	# RFs	# CRFs	# ϵ -CRFs					# 1-faSets	# 2-faSets
				$\epsilon = 0.1$	$\epsilon = 0.3$	$\epsilon = 0.5$	$\epsilon = 0.7$	$\epsilon = 0.9$		
ZIPF										
5%	186	2983	2283	2283	2177	1319	778	243	25	250
10%	145	3402	2639	2639	2505	1541	881	254		
20%	54	3238	2547	2547	2443	1556	876	154		
AUTOS										
5%	2106	2960	2737	2715	2683	2617	2547	2446	17664	851954
10%	2014	2856	2606	2591	2561	2486	2437	2338		
20%	1960	2654	2421	2398	2365	2278	2231	2126		
MOVIES										
5%	11533	11809	11787	11787	11783	11757	11730	11630	67094	561223
10%	11537	11820	11803	11803	11798	11768	11734	11621		
20%	11538	11826	11802	11802	11799	11776	11750	11643		

(a)

(b)

these results by selecting one or more facet condition. Our approach is different in that we do not tackle refinement, instead, our goal is to discover other interesting results related to the results of the original query. There is also some relation to *query reformulation*, where a query is relaxed or restricted when its results are too few or too many respectively, using term rewriting or query expansion to increase recall and precision (e.g., [15]). Again, our aim is locating interesting results that are highly related to the results of the original query. Besides restricting the query, another common method of addressing the too-many answers problem is ranking its results and presenting only the top- k ones to the user. This line of research is extensive; the work most related to ours is research on *automatically ranking* the results [7, 3]. Besides addressing a different problem, our approach is also different in that the granularity of ranking is at the level of faSets as opposed to whole result tuples. We also propose a novel method for frequency estimation. In terms of the interestingness score, similar measures have been used in the literature, such as unexpectedness [21] and χ^2 [5].

Yet another method of exploring results relies on *why queries* that consider the presence of unexpected tuples in the result and *why not* queries that consider the absence of expected tuples in the result. For example, ConQueR [19] proposes posing follow-up queries for *why not* by relaxing the original query. In our approach, we find interesting faSets in the result based on their frequency and other faSets highly correlated with them. Another line of research considers how to “output” a query whose execution will yield results equivalent to a given result set [20, 16]. Our work differs in that we do not aim at constructing queries to match given result sets but rather guiding the users towards novel results.

Finally, in some respect, exploration queries may be seen as *recommendations*. In a previous position paper [17], we have discussed various approaches for making recommendations in relational databases. Extending database queries with recommendations has been studied in two recent works, namely [13] and [6]. In [13], a general framework and a related engine are proposed for the declarative specification of the recommendation process. Recommendations in [6] are based on the past behavior of similar users, whereas we consider only the content of the database and the result.

6. CONCLUSIONS

In this paper, we introduced ReDRIVE, a novel database exploration framework for recommending to users items which may be of interest to them although not part of the results of their original query. The computation of such additional results is based on identifying the most interesting sets of (attribute, value) pairs, or faSets, that appear in the result of the original user query. The computation of interestingness is based on the frequency of the faSet in the query result and in the database instance. We also proposed a frequency estimation method based on storing an ϵ -CRF representation

and a two-phase algorithm for computing the top- k faSets. There are many directions for future work, such as, extending our work to more general types of facet conditions and considering that a history of previous database queries and results exists.

7. REFERENCES

- [1] The Internet Movie Database. <http://www.imdb.com>.
- [2] Yahoo!Auto. <http://autos.yahoo.com>.
- [3] S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis. Automated ranking of database query results. In *CIDR*, 2003.
- [4] Y. M. Bishop, S. E. Fienberg, and P. W. Holland. *Discrete Multivariate Analysis: Theory and Practice*. Springer, 2007.
- [5] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In *SIGMOD*, 1997.
- [6] G. Chatzopoulou, M. Eirinaki, and N. Polyzotis. Query recommendations for interactive database exploration. In *SSDBM*, 2009.
- [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), 2006.
- [8] J. Cheng, Y. Ke, and W. Ng. delta-tolerance closed frequent itemsets. In *ICDM*, 2006.
- [9] M. Drosou and E. Pitoura. ReDRIVE: Result-driven database exploration through recommendations, TR 2011-10. *Computer Science Dept., Univ. of Ioannina*, 2011.
- [10] D. Gunopulos, R. Khardon, H. Mannila, S. Saluja, H. Toivonen, and R. S. Sharm. Discovering all most specific sentences. *ACM Trans. Database Syst.*, 28(2), 2003.
- [11] A. Kashyap, V. Hristidis, and M. Petropoulos. Facetor: cost-driven exploration of faceted query results. In *CIKM*, 2010.
- [12] N. Koudas, C. Li, A. K. H. Tung, and R. Vernica. Relaxing join and selection queries. In *VLDB*, 2006.
- [13] G. Koutrika, B. Bercovitz, and H. Garcia-Molina. Flexrecs: expressing and combining flexible recommendations. In *SIGMOD*, 2009.
- [14] S. B. Roy, H. Wang, G. Das, U. Nambiar, and M. K. Mohania. Minimum-effort driven dynamic faceted search in structured databases. In *CIKM*, 2008.
- [15] N. Sarkas, N. Bansal, G. Das, and N. Koudas. Measure-driven keyword-query expansion. *PVLDB*, 2(1), 2009.
- [16] A. D. Sarma, A. G. Parameswaran, H. Garcia-Molina, and J. Widom. Synthesizing view definitions from data. In *ICDT*, 2010.
- [17] K. Stefanidis, M. Drosou, and E. Pitoura. “You May Also Like” results in relational databases. In *PersDB*, 2009.
- [18] L. Szathmary, A. Napoli, and P. Valtchev. Towards rare itemset mining. In *ICTAI (1)*, 2007.
- [19] Q. T. Tran and C.-Y. Chan. How to conquer why-not questions. In *SIGMOD*, 2010.
- [20] Q. T. Tran, C.-Y. Chan, and S. Parthasarathy. Query by output. In *SIGMOD*, 2009.
- [21] K. Wang, Y. Jiang, and L. V. S. Lakshmanan. Mining unexpected rules by pushing user dynamics. In *KDD*, 2003.