Finding All Occurring Sets of Interest

Taneli Mielikäinen

HIIT Basic Research Unit
Department of Computer Science
University of Helsinki, Finland
Taneli.Mielikainen@cs.helsinki.fi

Abstract. In this paper we examine the problem of mining all occurring sets of interest. We define what they are, sketch some applications, describe streaming algorithms for the problem and analyze their computational complexity. We also study alternative representations for the occurring sets of interest and evaluate some of them experimentally.

1 Introduction

Finding all sets that occur frequently in a given data set has been a very popular research topic in data mining for years [1,2,3,4,5]. This *frequent set mining problem* can be stated as follows: given a finite set R of *attributes*, a sequence $d = d_1, \ldots, d_n$ of its subsets $d_i \in 2^R, 1 \leq i \leq n$, and a threshold value $\sigma \in [0,1]$, find all σ-frequent sets in d, i.e., find all subsets X of R such that the frequency

$$fr\,(X) = fr\,(X, d) = \frac{|\{i : X \subseteq d_i, 1 \leq i \leq n\}|}{n}$$

of X is at least σ. The collection of σ-frequent sets in d is denoted by

$$\mathcal{F}_{\sigma,d} = \{X \subseteq R : fr\,(X, d) \geq \sigma\}\,.$$

The set of different sets in d is denoted by $\mathcal{D} = \{d_i : 1 \leq i \leq n\}$ and the number of (exact) occurrences of set X in d is denoted by $occ\,(X, d) = |\{i : X = d_i\}|$. The sequence d is sometimes called a *database*. Then the elements d_i of d are called *transactions* or *rows*.

One practical reason why the frequencies of only the σ-frequent sets are computed instead of the frequencies of the whole set collection 2^R, is that 2^R contains $2^{|R|}$ subsets which is too large to be practical already for very small numbers of attributes. Also, the frequent sets are presumed to contain the most relevant sets in 2^R. However, it is not always easy to find a good minimum frequency threshold σ such that $\mathcal{F}_{\sigma,d}$ would contain most of the relevant sets and only few irrelevant ones. It might be even possible that such threshold does not exist at all. This is the case, for example, when some combinations of attributes are are not relevant no matter of their frequencies.

These observations suggest to study a task complementary to the frequent set mining, namely mining all sets occurring in d that are contained in some of the given subsets of attributes. The problem can be formulated more precisely as follows: given a finite set R of attributes, a sequence $d = d_1, \ldots, d_n$ of its subsets, and collection $\mathcal{S}$ of subsets of R, find all *occurring sets of interest* in d, i.e., find all subsets of R that are contained in at least one set in $\mathcal{S}$. Analogously to the set of frequent sets, the set of occurring sets of interest is denoted by

$$\mathcal{I}_{\mathcal{S},d} = \{X \subseteq Y \cap Z : Y \in \mathcal{S}, Z \in \mathcal{D}\}.$$

The set $\mathcal{I}_{\mathcal{S},d}$ can be computed by first generating all subsets of the sets in $\mathcal{S}$ and then counting their frequencies in d. As the collection $\mathcal{F}_{\sigma,d}$, also the collection $\mathcal{I}_{\mathcal{S},d}$ can be very redundant. The redundancy of frequent sets has motivated several studies of *condensed (or concise) representations* of set collections, i.e., subsets of the set collection (and their frequencies) from which all the other sets (and their frequencies) in the collection can be inferred. Some of the condensed representations, e.g. closed sets [6,7,8,9], disjunction-free sets [10] and their generalizations [11,12,13,14], represent both the frequent sets and their exact frequencies while some other allow approximate representations of frequencies [15,16,17,18,19,20,21] or determine just the collection of frequent sets without the actual frequencies [22,23,24,25]. These representations are usually based on the frequencies of the frequent sets and they can be significantly smaller than the collection of all frequent sets. The condensed representations for frequent sets can be adapted to the occurring sets of interest.

In this paper we shall use two condensed representations called *maximal sets* and *closed sets*. The maximal sets suffice to determine uniquely the collection of frequent sets.

Definition 1. *A set $X \in \mathcal{C}$ is maximal in a set collection $\mathcal{C}$ iff the collection does not contain any of its supersets, i.e., iff $Y \supset X \Rightarrow Y \notin \mathcal{C}$. The set of maximal sets in a collection $\mathcal{C}$ is denoted by $max(\mathcal{C})$.*

If we are interested also in the actual frequencies of the frequent sets, we can determine maximal sets using each frequency in $\{fr(X,d) : X \in \mathcal{F}_{\sigma,d}\}$ as the minimum frequency threshold. These sets together form the collection of closed sets in $\mathcal{F}_{\sigma,d}$.

Definition 2. *A set $X \in \mathcal{C}$ is closed in a set collection $\mathcal{C}$ (w.r.t. d) iff it has no supersets in $\mathcal{S}$ with the same frequency, i.e., iff $X \subset Y \in \mathcal{C} \Rightarrow fr(X,d) > fr(Y,d)$.*

From the above mentioned condensed representations one can derive another application of the occurring sets of interest, in addition of being an alternative to minimum frequency thresholds for the set collections: the occurring sets of interest can be used to refine already computed coarse condensed representations. For example, the closed frequent sets can be computed from the maximal frequent sets and the data set. This kind of resource-aware mining with additional information might be useful e.g. in ubiquitous computing.

We have developed some streaming algorithms for mining all occurring sets of interest, i.e., algorithms that find all occurring sets of interest without storing the sequence d. (For a short introduction to data streams, see e.g. [26].) Thus the collection of the occurring sets of interest can be used also as a summary of a data stream by choosing the set $\mathcal{S}$ to be a collection of few small random subsets of R and maintaining the occurring sets of interest for the set $\mathcal{S}$. This approach does not require any data set dependent parameters as the minimum frequency threshold in frequent set mining.

The rest of the paper is organized as follows. In Section 2 we consider the computational problem of finding all occurring closed sets of interest. In Section 3 we examine alternative representations for the occurring sets of interest. Section 4 concludes the paper.

2 Mining Closed Occurring Sets of Interest

In this section we consider the problem of computing all closed (occurring) sets of interest. We describe several algorithms and analyze their computational complexity. Also, we propose a new data structure that might have some interest of its own. Let us first show a useful lemma about closed sets:

Lemma 1. *The set $X \subseteq R$ is closed in $2^R = \{Y \subseteq R\}$ iff $X = \bigcap_{i \in I} d_i$ for some $I \subseteq [n] = \{1, \ldots, n\}$.*

Proof. By Definition 2, $X \subseteq R$ is closed in 2^R iff $fr\,(X, d) > fr\,(Y, d)$ for all $Y \supset X, Y \subseteq R$. If $X = \bigcap_{i \in I} d_i$ for some $I \subseteq [n]$ then for all of its supersets $Y \subseteq R$ there is $d_i, i \in I$, such that $Y \nsubseteq d_i$, i.e., $fr\,(X, d) > fr\,(Y, d)$. On the other hand, if $X \neq \bigcap_{i \in I} d_i$ for any $I \subseteq [n]$, then $X \subset \bigcap_{i \in I} d_i = Y$ for some $I \subseteq [n]$ and thus there is $Y \supset X, Y \subseteq R$, such that $fr\,(X, d) = fr\,(Y, d)$. □

Applying Lemma 1 we describe an algorithm that computes all the closed sets exhaustively and intersects them by the sets in $\mathcal{S}$:

```
INTERSECT-EXHAUSTIVE($\mathcal{S}, d$)
$\mathcal{I} \leftarrow \emptyset$
for each $I \subseteq \{1, \ldots, n\}$
   do $X \leftarrow \bigcap_{i \in I} d_i$
      for each $Y$ in $\mathcal{S}$
         do $Z \leftarrow X \cap Y$
            if $Z \notin \mathcal{I}$
               then $\mathcal{I} \leftarrow \mathcal{I} \cup Z$
                    $supp\,[Z] \leftarrow 0$
            if $supp\,[Z] < |I|$
               then $supp\,[Z] \leftarrow |I|$
for each $X$ in $\mathcal{I}$
   do $fr\,(X) \leftarrow supp\,[X]\,/n$
return $(\mathcal{I}, fr\,(\mathcal{I}))$
```

Let m denote $\min\{\max_{1 \leq i \leq n} |d_i|, \max_{Y \in \mathcal{S}} |Y|\}$. Then we can show that the algorithm Intersect-Exhaustive has the following time complexity:

Theorem 1. *Algorithm* Intersect-Exhaustive *finds all closed sets of interest and their frequencies in time* $\mathcal{O}\left(m2^n\left|\mathcal{S}\right|\right)$.

Proof. By Lemma 1, the algorithm computes all closed sets in 2^R and projects them by $\mathcal{S}$. Thus the set $\mathcal{I}$ the algorithm produces is equal to $\mathcal{I}_{\mathcal{S},d}$.

The algorithm computes $2^n\left|\mathcal{S}\right|$ intersections, the number of terms in the intersection is $n+1$ in the worst case and the number of comparisons needed to intersect two sets is $|R|$ in the worst case. Thus all intersections can be computed in time $\mathcal{O}\left(|R|n2^n\left|\mathcal{S}\right|\right)$.

However, by computing all intersections of k sets before the intersections of $k+1$ sets and memorizing the intersections of k sets, each intersection can be computed in time $|R|$. Furthermore, the intersection between sets X and Y can be computed in time $\mathcal{O}\left(\min\left\{|X|,|Y|\right\}\right)$ by testing which elements of the smaller set are contained in the larger one. □

Unfortunately, the algorithm Intersect-Exhaustive is too slow. Moreover, it has to memorize the whole sequence d as the intersections of subsets have to be computed. (In fact, already computing all pairwise intersections $d_i \cap d_j, 1 \leq i < j \leq n$, would require this.) Note that because the sets in $\mathcal{S}$ should reduce the huge set of all subsets of R to a smaller collection of the occurring sets of interest, storing the sequence d itself should not be necessary.

It turns out that the computation of the intersections can reorganized to be more efficient. This can be done by noticing that all possible combinations of intersections producing some closed set X do not have to be computed: it is enough to produce each closed set once and to be able to compute its frequency. This observation can be formulated as an incremental streaming output-efficient algorithm as follows:

```
Intersect-Incremental($d, \mathcal{S}$)
$\mathcal{I} \leftarrow \mathcal{S}$
for each $Y$ in $\mathcal{I}$
    do $lastVisited[Y] \leftarrow 0$
for $i \leftarrow 1$ to $n$
    do for each $Y$ in $\mathcal{I}$
           do $Z \leftarrow d_i \cap Y$
              if $Z \notin \mathcal{I}$
                then $\mathcal{I} \leftarrow \mathcal{I} \cup Z$
                     $supp[Z] \leftarrow 1$
                     $lastVisited[Y] \leftarrow i$
              if $lastVisited[Y] < i$
                then $supp[Z] \leftarrow supp[Z] + 1$
for each $X$ in $\mathcal{I}$
    do if $supp[X] = 0$
         then $\mathcal{I} \leftarrow \mathcal{I} \setminus \{X\}$
         else $fr(X) \leftarrow supp[X]/n$
return $(\mathcal{I}, fr(\mathcal{I}))$
```

Theorem 2. *The algorithm* Intersect-Incremental *finds all closed sets of interest and their frequencies in time* $\mathcal{O}\left(nm\left(|\mathcal{I}_{\mathcal{S},d}| + |\mathcal{S}|\right)\right)$.

Proof. Let $\mathcal{I}_0, \mathcal{I}_1, \ldots, \mathcal{I}_n = \mathcal{I}$ denote the solutions found by the algorithm Intersect-Incremental for prefixes of d of lengths $0, 1, \ldots n$, respectively. Clearly, $\mathcal{I}_0 \subseteq \mathcal{I}_1 \subseteq \ldots \subseteq \mathcal{I}_n$. We first show that $\mathcal{I} = \mathcal{I}_{\mathcal{S},d} \cup \mathcal{S}$ by induction on the length of the sequence:

1. If $|d| = 0$, then there are no occurring sets in d and $\mathcal{I}_0 = \mathcal{S}$.
2. Assume that for some $k < n$, $\mathcal{I}_i = \mathcal{I}_{\mathcal{S},d_1 \ldots d_i} \cup \mathcal{S}$ holds for all $0 \leq i \leq k$. Then

$$\begin{aligned}
\mathcal{I}_{k+1} &= \mathcal{I}_k \cup \{d_{k+1} \cap X : X \in \mathcal{I}_k\} \cup \{d_{k+1} \cap Y : Y \in \mathcal{S}\} \\
&= \left\{\bigcap_{i \in I} d_i \cap Y : Y \in \mathcal{S}, I \subseteq [k]\right\} \cup \left\{d_{i+1} \cap \bigcap_{i \in I} d_i \cap Y : Y \in \mathcal{S}, I \subseteq [k]\right\} \\
&= \left\{Y \cap \bigcap_{i \in I} d_i : Y \in \mathcal{S}, I \subseteq [k+1]\right\} = \mathcal{I}_{\mathcal{S},d_1 \ldots d_{k+1}} \cup \mathcal{S}.
\end{aligned}$$

Thus $\mathcal{I}_i = \mathcal{I}_{\mathcal{S},d_1 \ldots d_i} \cup \mathcal{S}$ holds for all $0 \leq i \leq n$. In particular, $\mathcal{I} = \mathcal{I}_n = \mathcal{I}_{\mathcal{S},d} \cup \mathcal{S}$ as claimed.

The counts of the sets in $\mathcal{S} \setminus \mathcal{I}_{\mathcal{S},d}$ remain zero and they can be removed from the collection $\mathcal{I}_{\mathcal{S},d}$ in time $\mathcal{O}(|\mathcal{I}|)$. The frequencies of the closed occurring sets of interest are correct because each $d_i, 1 \leq i \leq n$, increases the count of each $X \in \mathcal{I}_{\mathcal{S},d}$ contained in d_i exactly once.

As each set $d_i, 1 \leq i \leq n$, can intersect at most $|\mathcal{I}|$ sets in $\mathcal{I}_{\mathcal{S},d} \cup \mathcal{S}$ and one intersection can be computed in time $\mathcal{O}(m)$, the set $\mathcal{I} \setminus \mathcal{S} = \mathcal{I}_{\mathcal{S},d}$ and the frequencies of the sets $X \in \mathcal{I}_{\mathcal{S},d}$ can be computed in time $\mathcal{O}(nm\,|\mathcal{I}|) = \mathcal{O}\left(nm\left(|\mathcal{I}_{\mathcal{S},d}| + |\mathcal{S}|\right)\right)$. □

One problem with the algorithm Intersect-Incremental is that it can compute the empty intersection several times for each set d_i. We attempt to avoid this problem by introducing a data structure called *skewers* that allows efficient computation of non-empty intersections.

Definition 3 (Skewers). *A skewers data structure* $\mathcal{V}$ *represents a collection of sets. Each set* X *in the skewers consists a set of attributes* $\mathcal{V}_X^X$*, a counter* $\mathcal{V}_{supp}^X$ *and the time of last visit* $\mathcal{V}_{time}^X$*.*

The sets in $\mathcal{V}$ *are in ascending order in their cardinality. In addition there is a skewer* $\mathcal{V}^A$ *for each attribute* $A \in R$ *consisting the sets containing* A *in the order conforming the global order of sets.*

The skewers $\mathcal{V}$ *supports operations insertion, deletion and location of a set* X *in time* $\mathcal{O}(|X| \log |\mathcal{V}|)$*. Next and previous set in each skewer* $\mathcal{V}^A$ *can be found in constant time.*

The intersection algorithm for skewers incrementally intersect the sets in the skewers data structure by scanning over the skewers. The scan over the sets is implemented by a priority queue.

```
Intersect-Skewers($d, \mathcal{S}$)
$\mathcal{I} \leftarrow \emptyset$
$\mathcal{V} \leftarrow \emptyset$
for each $Y \in \mathcal{S}$
    do Insert-Skewers $(Y)$
for $i \leftarrow 1$ to $n$
    do $\mathcal{Q} \leftarrow \emptyset$
        for each $A$ in $d_i$
            do Insert-Queue $\left(\mathcal{Q}, \mathcal{V}^A_{head}\right)$
        while $\mathcal{Q} \neq \emptyset$
            do $X \leftarrow$ Extract-Queue $(\mathcal{Q})$
                $Y \leftarrow X \cap d_i$
                if $\mathcal{V}^Y = $ nil
                   then Insert-Skewers $(Y)$
                        $\mathcal{I} \leftarrow \mathcal{I} \cup \{Y\}$
                if $\mathcal{V}^Y_{time} < i$
                   then $\mathcal{V}^Y_{supp} \leftarrow \mathcal{V}^Y_{supp} + 1$
                        $\mathcal{V}^Y_{time} \leftarrow i$
                for each $A$ in $X$
                    do Insert-Queue $\left(\mathcal{Q}, \text{Next-Skewer}\left(\mathcal{V}^A\right)\right)$
for each $X$ in $\mathcal{I}$
    do $fr\,(X) \leftarrow \mathcal{V}^X_{supp}/n$
return $(\mathcal{I}, fr\,(\mathcal{I}))$
```

Theorem 3. *The algorithm* Intersect-Skewers *finds all closed sets of interest and their frequencies in time*

$$\mathcal{O}\left(\sum_{i=1}^{n}\left(|d_i| + \log|\mathcal{V}|\right)\sum_{A\in d_i}\left|\mathcal{V}^A\right|\right).$$

Proof. All closed sets of interest and their frequencies are found as the algorithm Intersect-Skewers incrementally constructs the collection of all closed sets of interest similarly to the algorithm Intersect-Incremental.

The number of sets intersected by d_i is bounded by $\sum_{A\in d_i}\left|\mathcal{V}^A\right|$. Each intersection can be computed in time $|d_i|$ and the set corresponding to the intersection can be found in time $\log|\mathcal{V}|$. Thus, the combined time complexity is $\mathcal{O}\left(\sum_{i=1}^{n}\left(|d_i| + \log|\mathcal{V}|\right)\sum_{A\in d_i}\left|\mathcal{V}^A\right|\right)$ as claimed. □

The time bound is quite pessimistic as the bound does not take into account e.g. that the same set is not added into the priority queue more than once per scan. Furthermore, if all data d can be stored, then the computation can be made more efficient by replacing the sequence d by the set $\mathcal{D}$ with number of occurrences $occ\,(X, d)$ for each $X \in \mathcal{D}$.

The algorithms can be adapted to mine the closed frequent sets instead of the closed sets of interest by maintaining upper bounds of frequencies for the closed sets and removing a closed set when it is clear that the set is infrequent.

3 Simplified Databases

It is not clear whether the closed sets of interest should be mined explicitly. For example, the closed sets are not very good as an index structure for frequency queries by subsets of R.

It is easy to see that the number of the closed sets in 2^R is at least as large as $|\mathcal{D}|$ due to the fact that each set in $\mathcal{D}$ is a closed set. Usually the number of closed sets is even higher because also all intersections $\bigcap_{X \in \mathcal{C}} X, \mathcal{C} \subseteq \mathcal{D}$, are closed sets. However, if we are considering the closed sets of interest instead of all closed sets in 2^R, then the number of different rows in the database d is not necessarily smaller: The set collection $\mathcal{S}$ can be chosen to be the collection of the maximal frequent sets. Then the set of the closed occurring sets of interest is the collection of the closed frequent sets. The collection of the closed frequent sets can be much smaller than the original database d or even smaller than the number $|\mathcal{D}|$ of different rows in d.

However, also the database d can be condensed. The simplest approach is to replace the database by the set $\mathcal{D}$ and the number of occurrences $occ(X, d)$ for each $X \in \mathcal{D}$. The number of sets in $\mathcal{D}$ can be further reduced by removing all attributes that are not present in $\mathcal{S}$. If the number of sets in $\mathcal{S}$ is very small, it can be worthwhile to store the projection of the database $\mathcal{D}$ onto Y for each $Y \in \mathcal{S}$ separately.

We tested with few data sets how the sizes of different representations of the set collection and the database relate to each other. The experiments were done using IPUMS Census data set from UCI KDD Repository[1] and Ilmo data set from the course enrollment system of Department of Computer Science, University of Helsinki. IPUMS Census consists of 88443 rows (88211 different rows) and 39954 attributes. Ilmo consists of 3505 rows (1903 different rows) and 97 attributes. We simulated different set collections $\mathcal{S}$ by computing the maximal σ-frequent sets for several minimum frequency thresholds σ. Thus in our experiments the sets in $\mathcal{S}$ correspond to the maximal σ-frequent sets.

The results are shown in Table 1 and Table 2. The column "pruned rows" corresponds to the number of different rows in d after removing the attributes that do not occur in $\mathcal{S}$. The column "projections" corresponds to the combined number of different rows in each projection of $\mathcal{D}$ onto $Y \in \mathcal{S}$. It can be seen from the tables that all of the representations have their good sides. The representation of database based on separate projections to all sets in $\mathcal{S}$ do nicely with high minimum frequency thresholds but eventually, as the minimum frequency threshold decreases, the collection of pruned database rows succeeds to be the smallest one.

In general, the small representations of the databases and occurring sets of interest seem to have some nontrivial computational problems:

1. Given a data set d and a collection $\mathcal{S}$ of subsets of R, find the smallest data set d' that agrees with d when projected onto any $Y \in \mathcal{S}$.

[1] `http://kdd.ics.uci.edu`

σ	$\mathcal{S} := max\left(\mathcal{F}_{\sigma,d}\right)$	$cl\left(\mathcal{I}_{\mathcal{S},d}\right) = cl\left(\mathcal{F}_{\sigma,d}\right)$	$\mathcal{I}_{\mathcal{S},d} = \mathcal{F}_{\sigma,d}$	pruned rows	projections
0.40	41	1517	1517	11934	595
0.38	56	2111	2111	13490	847
0.36	66	2795	2795	15424	1154
0.34	82	3975	3975	15993	1636
0.32	107	5361	5361	17777	2400
0.30	126	8205	8205	22626	3083
0.28	152	11443	11443	22763	4179
0.26	198	17503	17503	22763	5774
0.24	272	23903	23903	27429	8450
0.22	387	53203	53203	31488	12935
0.20	578	86879	86879	39730	22616
0.18	789	250441	250441	40128	37435
0.16	1082	524683	524683	44534	63784

Table 1. IPUMS Census data set

σ	$\mathcal{S} := max\left(\mathcal{F}_{\sigma,d}\right)$	$cl\left(\mathcal{I}_{\mathcal{S},d}\right) = cl\left(\mathcal{F}_{\sigma,d}\right)$	$\mathcal{I}_{\mathcal{S},d} = \mathcal{F}_{\sigma,d}$	pruned rows	projections
0.040	102	286	286	1546	978
0.038	113	355	355	1546	1328
0.036	132	430	430	1546	1878
0.034	123	512	512	1581	1713
0.032	123	594	594	1620	1958
0.030	157	691	691	1651	2645
0.028	190	847	847	1651	3519
0.026	226	1090	1095	1693	4814
0.024	260	1363	1378	1693	6192
0.022	292	1741	1771	1693	8380
0.020	359	2264	2348	1699	11605
0.018	436	3017	3226	1714	15109
0.016	565	4078	4519	1723	22044

Table 2. Ilmo data set

2. Given a data set d and a collection $\mathcal{S}$ of subsets of R, find a collection of data sets $d^1, \ldots, d^k$ with the smallest combined number of different rows and a mapping $f : \mathcal{S} \to [k]$ such that $d^{f(Y)}$ agrees with d when projected onto any $Y \in \mathcal{S}$.

The second problem can be attempted to solve by the following heuristic:

1. Compute projections d^Y of the database d onto each $Y \in \mathcal{S}$.
2. Find the pair of projections d^Y and d^Z such that $\Delta = \left|d^Y\right| + \left|d^Z\right| - \left|d^{Y \cup Z}\right|$ is largest.
3. If $\Delta > 0$, then replace d^Y and d^Z by $d^{Y \cup Z}$ and go to step 2.

The algorithm can be implemented in time $\mathcal{O}\left(|R|\,|\mathcal{D}|\,|\mathcal{S}|^2\right)$ where $R = \bigcup_{Y \in \mathcal{S}} Y$. The solution found by the heuristic is at least as good as projecting the database the separately onto each set in $\mathcal{S}$.

4 Conclusions

In this paper we have studied the problem of finding sets X contained in a set d_i of a data set d such that $X \subseteq Y$ for some $Y \in \mathcal{S}$. We have proposed streaming algorithms for the problem and discussed about other possible representations for the occurring sets of interest. There are some interesting open problems:

- How the closed sets of interest with $\mathcal{S}$ as a collection of random projections could be used as a useful summary for a data stream?
- Can the algorithms used to speed up the frequent set mining in conjunction with some efficient implementation of a maximal set mining algorithm?
- Where the skewers data structure could be applied?
- Is there some clear relationship between the overlap of sets in $\mathcal{S}$, the size of the collection $\mathcal{I}_{\mathcal{S},d}$ and the size of d?
- How a database can be transformed into a smallest form that can be used to answer to certain queries efficiently and (approximately) correctly?

References

1. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast discovery of association rules. In Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R., eds.: Advances in Knowledge Discovery and Data Mining. AAAI/MIT Press (1996) 307–328
2. Goethals, B.: Survey on frequent pattern mining. Manuscript (2003)
3. Hand, D.J.: Pattern detection and discovery. In Hand, D., Adams, N., Bolton, R., eds.: Pattern Detection and Discovery. Volume 2447 of LNAI., Springer-Verlag (2002) 1–12
4. Hipp, J., Güntzer, U., Nakhaeizadeh, G.: Algorithms for association rule mining – a general survey and comparison. SIGKDD Explorations **1** (2000) 58–64
5. Mannila, H.: Local and global methods in data mining: Basic techniques and open problems. In Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R., eds.: Automata, Languages and Programming. Volume 2380 of LNCS., Springer-Verlag (2002) 57–68
6. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Discovering frequent closed itemsets for association rules. In Beeri, C., Buneman, P., eds.: Database Theory - ICDT'99. Volume 1540 of LNCS., Springer-Verlag (1999) 398–416
7. Pei, J., Han, J., Mao, T.: CLOSET: An efficient algorithm for mining frequent closed itemsets. In Gunopulos, D., Rastogi, R., eds.: ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery. (2000) 21–30
8. Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L.: Computing iceberg concept lattices with Titanic. Data & Knowledge Engineering **42** (2002) 189–222
9. Zaki, M.J., Hsiao, C.J.: CHARM: An efficient algorithms for closed itemset mining. In Grossman, R., Han, J., Kumar, V., Mannila, H., Motwani, R., eds.: Proceedings of the Second SIAM International Conference on Data Mining, SIAM (2002)
10. Bykowski, A., Rigotti, C.: A condensed representation to find frequent patterns. In: Proceedings of the Twenteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, ACM (2001)

11. Calders, T., Goethals, B.: Mining all non-derivable frequent itemsets. In Elomaa, T., Mannila, H., Toivonen, H., eds.: Principles of Data Mining and Knowledge Discovery. Volume 2431 of LNAI., Springer-Verlag (2002) 74–865
12. Calders, T., Goethals, B.: Minimal *k*-free representations of frequent sets. In Lavrac, N., Gamberger, D., Todorovski, L., Blockeel, H., eds.: Principles of Knowledge Discovery and Data Mining. LNAI, Springer-Verlag (2003)
13. Kryszkiewicz, M.: Concise representation of frequent patterns based on disjunction-free generators. In Cercone, N., Lin, T.Y., Wu, X., eds.: Proceedings of the 2001 IEEE International Conference on Data Mining, IEEE Computer Society (2001) 305–312
14. Kryszkiewicz, M., Gajek, M.: Concise representation of frequent patterns based on generalized disjunction-free generators. In Chen, M.S., Yu, P., Liu, B., eds.: Advances in Knowledge Discovery and Data Mining. Volume 2336 of LNAI., Springer-Verlag (2002) 159 – 171
15. Bouliçaut, J.F., Bykowski, A.: Frequent closures as a concise representation for binary data mining. In Terano, T., Liu, H., Chen, A.L.P., eds.: Knowledge Discovery and Data Mining. Volume 1805 of LNAI., Springer-Verlag (2000) 62–73
16. Bouliçaut, J.F., Bykowski, A., Rigotti, C.: Free-sets: a condensed representation of Boolean data for the approximation of frequency queries. Data Mining and Knowledge Discovery **7** (2003) 5–22
17. Geerts, F., Goethals, B., Mielikäinen, T.: What you store is what you get (extended abstract). In: 2nd International Workshop on Knowledge Discovery in Inductive Databases. (2003)
18. Mielikäinen, T.: Frequency-based views to pattern collections. In: IFIP/SIAM Workshop on Discrete Mathematics and Data Mining. (2003)
19. Mielikäinen, T., Mannila, H.: The pattern ordering problem. In Lavrac, N., Gamberger, D., Todorovski, L., Blockeel, H., eds.: Principles of Knowledge Discovery and Data Mining. LNAI, Springer-Verlag (2003)
20. Pavlov, D., Mannila, H., Smyth, P.: Beyond independence: probabilistic methods for query approximation on binary transaction data. IEEE Transactions on Data and Knowledge Engineering (2003) To appear.
21. Pei, J., Dong, G., Zou, W., Han, J.: On computing condensed pattern bases. In: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan, IEEE Computer Society (2002) 378–385
22. Bayardo Jr., R.J.: Efficiently mining long patterns from databases. In Laura M. Haas, A.T., ed.: SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, ACM (1998) 85–93
23. Boros, E., Gurvich, V., Khachiyan, L., Makino, K.: On the complexity of generating maximal frequent and minimal infrequent sets. In Alt, H., Ferreira, A., eds.: STACS 2002. Volume 2285 of LNCS., Springer-Verlag (2002) 133–141
24. Gouda, K., Zaki, M.J.: Efficiently mining maximal frequent itemsets. In Cercone, N., Lin, T.Y., Wu, X., eds.: Proceedings of the 2001 IEEE International Conference on Data Mining. IEEE Computer Society (2001) 163–170
25. Gunopulos, D., Khardon, R., Mannila, H., Saluja, S., Toivonen, H., Sharma, R.S.: Discovering all most specific sentences. ACM Transactions on Database Systems **28** (2003) 140–174
26. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In Popa, L., ed.: Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, ACM (2002) 1–16