Comprehensive Log Compression with Frequent Patterns

Kimmo Hätönen[1], Jean François Boulicaut[2], Mika Klemettinen[1], Markus Miettinen[1], and Cyrille Masson[2]

[1] Nokia Research Center
P.O.Box 407, FIN-00045 Nokia Group, Finland
{kimmo.hatonen,mika.klemettinen,markus.miettinen}@nokia.com
[2] INSA de Lyon, LIRIS CNRS FRE 2672
F-69621 Villeurbanne, France
{Jean-Francois.Boulicaut,Cyrille.Masson}@insa-lyon.fr

Abstract. In this paper we present a comprehensive log compression (CLC) method that uses frequent patterns and their condensed representations to identify repetitive information from large log files generated by communications networks. We also show how the identified information can be used to separate and filter out frequently occurring events that hide other, unique or only a few times occurring events. The identification can be done without any prior knowledge about the domain or the events. For example, no pre-defined patterns or value combinations are needed. This separation makes it easier for a human observer to perceive and analyse large amounts of log data. The applicability of the CLC method is demonstrated with real-world examples from data communication networks.

1 Introduction

In the near future telecommunication networks will deploy an open packet-based infrastructure which has been originally developed for data communication networks. The monitoring of this new packet-based infrastructure will be a challenge for operators. The old networks will remain up and running for still some time. At the same time the rollout of the new infrastructure will take place introducing many new information sources, between which the information needed in, e.g., security monitoring and fault analysis will be scattered. These sources can include different kinds of event logs, e.g., firewall logs, operating systems' system logs and different application server logs to name a few. The problem is becoming worse every day as operators are adding new tools for logging and monitoring their networks. As the requirements for the quality of service perceived by customers gain more importance, the operators are starting to seriously utilise information that is hidden in these logs. Their interest towards analysing their own processes and operation of their network increases concurrently.

Data mining and knowledge discovery methods are a promising alternative for operators to gain more out of their data. Based on our experience, however,

Y. Kambayashi, M. Mohania, W. Wöß (Eds.): DaWaK 2003, LNCS 2737, pp. 360–370, 2003.

simple-minded use of discovery algorithms in the network analysis poses problems with the amount of generated information and its relevance. In the KDD process [6, 10, 9], it is often reasonable or even necessary to constrain the discovery using background knowledge. If no constraints are applied, the discovered result set of, say, association rules [1, 2] might become huge and contain mostly trivial and uninteresting rules. Also, association and episode rule mining techniques can only capture frequently recurring events according to some frequency and confidence thresholds. This is needed to restrict the search space and thus for computation tractability. Clearly, the thresholds that can be used are not necessarily the ones that denote objective interestingness from the user point of view. Indeed, rare combinations can be extremely interesting. When considering our previously unknown domains, an explicit background knowledge is missing, e.g., about the possible or reasonable values of attributes and their relationships.

When it is difficult or impossible to define and maintain a priori knowledge about the system, there is still a possibility to use *meta information* that can be extracted from the logs. Meta information characterizes different types of log entries and log entry combinations. It can not only be used to help an expert in filtering and browsing the logs manually but also to automatically identify and filter out insignificant log entries. It is possible to reduce the size of an analysed data set to a fraction of its original size without losing any critical information.

One type of meta information are frequent patterns. They capture the common value combinations that occur in the logs. Furthermore, such meta information can be condensed by means of, e.g., the closed frequent itemsets [12, 3]. Closed sets form natural inclusion graphs between different covering sets. This type of presentation is quite understandable for an expert and can be used to create hierarchical views. These condensed representations can be extracted directly from highly correlated and/or dense data, i.e., in contexts where the approaches that compute the whole collection of the frequent patterns FS are intractable [12, 3, 17, 13]. They can also be used to regenerate efficiently the whole FS collection, possibly partially and on the fly.

We propose here our *Comprehensive Log Compression* (CLC) method. It is based on the computation of frequent pattern condensed representations and we use this presentation as an entry point to the data. The method provides a way to dynamically characterize and combine log data entries before they are shown to a human observer. It finds frequently occurring patterns from dense log data and links patterns to the data as a data directory. It is also possible to separate recurring data and analyse it separately. In most cases, this reduces the amount of data needed to be evaluated by an expert to a fraction of the original volume.

This type of representation is general w.r.t. different log types. Frequent sets can be generated from most of the logs that have structure and contain repeating symbolic values in their fields, e.g., in Web Usage Mining applications [11, 16]. The main difference between the proposed method and those applications is the objective setting of the mining task. Most of the web usage applications try to identify and somehow validate common access patterns in web sites. These patterns are then used to do some sort of optimization of the site. The proposed

```
...
777;11May2000; 0:00:23;a_daemon;B1;12.12.123.12;tcp;;
778;11May2000; 0:00:31;a_daemon;B1;12.12.123.12;tcp;;
779;11May2000; 0:00:32;1234;B1;255.255.255.255;udp;;
781;11May2000; 0:00:43;a_daemon;B1;12.12.123.12;tcp;;
782;11May2000; 0:00:51;a_daemon;B1;12.12.123.12;tcp;;
...
```

Fig. 1. An example of a firewall log

method, however, doesn't say anything about semantic correctness or relations between the found frequent patterns. It only summarizes the most frequent value combinations in entries. This gives either a human expert or computationally more intensive algorithms a change to continue with data, which doesn't contain too common and trivial entries. Based on our experience with real-life log data, e.g., large application and firewall logs, the original data set of tens of thousands of rows can often be represented by just a couple of identified patterns and the exceptions not matching these patterns.

2 Log Data and Log Data Analysis

A log data consists of entries that represent a specific condition or an event that has occurred somewhere in the system. The entries have several fields, which are called variables from now on. The structure of entries might change over time from entry to another, although some variables are common to all of them. Each variable has a set of possible values called a value space. Values of one value space can be considered as binary attributes. Variable value spaces are separated. A small example of a log data is given in Figure 1. It shows a sample from a log file produced by CheckPoint's Firewall-1.

In a data set a value range in a variable value space might be very large or very limited. For example, there may be only few firewalls in an enterprise, but every IP address in the internet might try to contact the enterprise domain. There are also several variables that have such a large value space but contain only a fraction of the possible values. Therefore, it is unpractical and almost impossible to fix the size of the value spaces as a priori knowledge.

A log file may be very large. During one day, there might accumulate millions of lines into a log file. A solution to browse the data is either to search for patterns that are known to be interesting with high probability or to filter out patterns that most probably are uninteresting. A system can assist in this but the evaluation of interestingness is left for an expert. To be able to make the evaluation an expert has to check the found log entries by hand. He has to return to the original log file and iteratively check all those probably interesting entries and their surroundings. Many of the most dangerous attacks are new and unseen for an enterprise defense system. Therefore, when the data exploration is limited only to known patterns it may be impossible to find the new attacks.

Comprehensive Log Compression (CLC) is an operation where meta information is extracted from the log entries and used to summarize redundant entries

```
{Proto:tcp, Service:a_daemon, Src:B1}   11161
{Proto:tcp, SPort:, Src:B1} 11161
{Proto:tcp, SPort:, Service:a_daemon}   11161
{SPort:, Service:a_daemon, Src:B1}  11161
...
{Destination:123.12.123.12, SPort:, Service:a_daemon, Src:B1}   10283
{Destination:123.12.123.12, Proto:tcp, Service:a_daemon, Src:B1}    10283
{Destination:123.12.123.12, Proto:tcp, SPort:, Src:B1}  10283
{Destination:123.12.123.12, Proto:tcp, SPort:, Service:a_daemon}    10283
{Proto:tcp, SPort:, Service:a_daemon, Src:B1}   11161
...
{Destination:123.12.123.12, Proto:tcp, SPort:, Service:a_daemon, Src:B1}    10283
```

Fig. 2. A sample of frequent sets extracted from a firewall log

without losing any important information. By combining log entries with their frequencies and identifying recurring patterns, we are able to separate correlating entries from infrequent ones and display them with accompanying information. Thus, an expert has a more covering overview of the logged system and he can identify interesting phenomena and concentrate on his analysis.

The summary has to be understandable for an expert and must contain all the relevant information that is available in the original log. Presentation has also to provide a mechanism to move back and forth between the summary and the original logs.

Summarization can be done by finding correlating value combinations from large amount of log entries. Due to the nature of the logging mechanism, there are always several value combinations that are common to a large number of the entries. When these patterns are combined with information about how uncorrelating values are changing w.r.t. to these correlating patterns it gives a comprehensive description of the contents of the logs. In many cases it is possible to detect such patterns by browsing the log data but unfortunately it is also tedious. E.g., a clever attack against a firewall cluster of an enterprise is scattered over all of its firewalls and executed slowly from several different IP addresses using all the possible protocols alternately.

Figure 2 provides a sample of frequent sets extracted from the data introduced in Figure 1. In Figure 2, the last pattern, which contains five attributes, has five subpatterns out of which four have the same frequency as the longer pattern and only one has larger frequency. In fact, many frequent patterns have the same frequency and it is the key idea of the frequent closed set mining technique to consider only some representative patterns, i.e., the frequent closed itemsets (see next section for a formalization). Figure 3 gives a sample of frequent closed sets that correspond to the frequent patterns shown in Figure 2.

An example of the results of applying the CLC method to a firewall log data set can be seen in Table 1. It shows three patterns with highest coverage values found from the firewall log introduced in Figure 1. If the supports of these patterns are combined, then 91% of the data in the log is covered. The blank fields in the figure are intentionally left empty in the original log data. The fields marked with '*' can have varying values. For example, in the pattern 1 the field

```
{Proto:tcp, SPort:, Service:a_daemon, Src:B1}   11161
{Destination:123.12.123.12, Proto:tcp, SPort:, Service:a_daemon, Src:B1}    10283
{Destination:123.12.123.13, Proto:tcp, SPort:, Service:a_daemon, Src:B1}    878
```

Fig. 3. A sample of closed sets extracted from a firewall log

Table 1. The three most frequent patterns found from a firewall log

No	Destination	Proto	SPort	Service	Src	Count
1.	*	tcp		A_daemon	B1	11161
2.	255.255.255.255	udp		1234	*	1437
3.	123.12.123.12	udp		B-dgm	*	1607

'Destination' gets two different values on lines matched by it, as it is shown in Figure 3.

3 Formalization

The definition of a *LOG pattern domain* is made of the definition of a language of patterns $\mathcal{L}$, evaluation functions that assign a description to each pattern in a given log $\mathbf{r}$, and languages for primitive constraints that specify the desired patterns. We introduce some notations that are used for defining the LOG pattern domain. A so-called *log* contains the data in a form of *log entries* and patterns are the so-called *itemsets*, which are sets of $(field, value)$ pairs of log entries.

Definition 1 (Log). *Assume that* $\texttt{Items}$ *is a finite set of* $(field, value)$ *pairs denoted by field name combined with value, e.g.,* $\texttt{Items} = \{\texttt{A} : a_i, \texttt{B} : b_j, \texttt{C} : c_k, \ldots\}$. *A* log entry t *is a subset of* $\texttt{Items}$. *A* log $\mathbf{r}$ *is a finite and non empty multiset* $\mathbf{r} = \{e_1, e_2, \ldots, e_n\}$ *of log entries.*

Definition 2 (Itemsets). *An* itemset *is a subset of* $\texttt{Items}$. *The language of patterns for itemsets is* $\mathcal{L} = 2^{\texttt{Items}}$.

Definition 3 (Constraint). *If* $\mathcal{T}$ *denotes the set of all logs and* $2^{\texttt{Items}}$ *the set of all itemsets, an itemset constraint* $\mathcal{C}$ *is a predicate over* $2^{\texttt{Items}} \times \mathcal{T}$. *An itemset* $S \in 2^{\texttt{Items}}$ satisfies *a constraint* $\mathcal{C}$ *in the database* $\mathbf{r} \in \mathcal{T}$ *iff* $\mathcal{C}(S, \mathbf{r}) = true$. *When it is clear from the context, we write* $\mathcal{C}(S)$.

Evaluation functions return information about the properties of a given itemset in a given log. These functions provide an expert information about the events and conditions in the network. They also form a basis for summary creation. They are used to select the proper entry points to the log data.

Definition 4 (Support for Itemsets). *A log entry* e supports *an itemset* S *if every item in* S *belongs to* e, *i.e.,* $S \subseteq e$. *The* support *(denoted* $\text{support}(S, \mathbf{r})$*) of an itemset* S *is the multiset of all log entries of* $\mathbf{r}$ *that supports* S *(e.g.,* $\text{support}(\emptyset) = \mathbf{r}$*).*

Definition 5 (Frequency). *The* frequency *of an itemset S in a log $\mathbf{r}$ is defined by $\mathcal{F}(S, \mathbf{r}) = |\text{support}(S)|$ where $|.|$ denotes the cardinality of the multiset.*

Definition 6 (Coverage). *The* coverage *of an itemset S in a log $\mathbf{r}$ is defined by $\mathcal{C}ov(S, \mathbf{r}) = \mathcal{F}(S, \mathbf{r}) \cdot |S|$, where $|.|$ denotes the cardinality of the itemset S.*

Definition 7 (Perfectness). *The perfectness of an itemset S in a log $\mathbf{r}$ is defined by $\mathcal{P}erf(S, \mathbf{r}) = \mathcal{C}ov(S, \mathbf{r}) / \sum_{i=0}^{\mathcal{F}(S,\mathbf{r})} |e_i|$, where $\forall e_i : e_i \in \text{support}(S, \mathbf{r})$ and $|e_i|$ denotes to the cardinality of log entry e_i. Please, notice that if the cardinality of all the log entries is constant it applies then $\mathcal{P}erf(S, \mathbf{r}) = \mathcal{C}ov(S, \mathbf{r}) / (\mathcal{F}(S, \mathbf{r}) \cdot |e|)$, where e is an arbitrary log entry.*

Primitive constraints are a tool set that is used to create and control summaries. For instance, the summaries are composed by using the frequent (closed) sets, i.e., sets that satisfy a conjunction of a minimal frequency constraint and the closeness constraint plus the original data.

Definition 8 (Minimal Frequency). *Given an itemset S, a log $\mathbf{r}$, and a frequency threshold $\gamma \in [1, |\mathbf{r}|]$, $\mathcal{C}_{\text{minfreq}}(S, \mathbf{r}) \equiv \mathcal{F}(S, \mathbf{r}) \geq \gamma$. Itemsets that satisfy $\mathcal{C}_{\text{minfreq}}$ are called* γ-frequent *or frequent in $\mathbf{r}$.*

Definition 9 (Minimal Perfectness). *Given an itemset S, a log $\mathbf{r}$, and a perfectness threshold $\pi \in [0, 1]$, $\mathcal{C}_{\text{minperf}}(S, \mathbf{r}) \equiv \mathcal{P}erf(S, \mathbf{r}) \geq \pi$. Itemsets that satisfy $\mathcal{C}_{\text{minperf}}$ are called* π-perfect *or perfect in $\mathbf{r}$.*

Definition 10 (Closures, Closed Itemsets and Constraint $\mathcal{C}_{close}$). *The closure of an itemset S in $\mathbf{r}$ (denoted by $\texttt{closure}(S, \mathbf{r})$) is the maximal (for set inclusion) superset of S which has the same support than S. In other terms, the closure of S is the set of items that are common to all the log entries which support S. A closed itemset is an itemset that is equal to its closure in $\mathbf{r}$, i.e., we define $\mathcal{C}_{close}(S, \mathbf{r}) \equiv \texttt{closure}(S, \mathbf{r}) = S$. Closed itemsets are maximal sets of items that are supported by a multiset of log entries.*

If we consider the equivalence class that group all the itemsets that have the same closure (and thus the same frequency), the closed sets are the maximal elements of each equivalence class. Thus, when the collection of the frequent itemsets FS is available, a simple post-processing technique can be applied to compute only the frequent closed itemsets. When the data is sparse, it is possible to compute FS, e.g., by using Apriori-like algorithms [2]. However, the number of frequent itemsets can be extremely large, especially in dense logs that contain many highly correlated field values. In that case, computing FS might not be feasible while the frequent closed sets CFS can often be computed for the same frequency threshold or even a lower one. $CFS = \{\phi \in \mathcal{L} \mid \mathcal{C}_{\text{minfreq}}(\phi, \mathbf{r}) \wedge \mathcal{C}_{close}(\phi, \mathbf{r}) \text{ satisfied}\}$. On one hand, FS can be efficiently derived from CFS without scanning the data again [12, 3]. On the other hand, CFS is a compact representation of the information about every frequent set and its frequency and thus fulfills the needs for CLC. Several algorithms can compute efficiently the frequent closed sets. In this work, we compute the frequent closed sets by

computing the frequent free sets and providing their closures [4, 5]. This is efficient since the freeness property is anti-monotonic, i.e., a key property for an efficient processing of the search space.

For a user, displaying of the adequate information is the most important phase of the CLC method. This phase gets the original log file and a condensed set of frequent patterns as input. An objective of the method is to select the most informative patterns as starting points for navigating the condensed set of patterns and data. As it has been shown [12], the frequent closed sets give rise to a lattice structure, ordered by set inclusion. These inclusion relations between patterns can be used as navigational links.

What are the most informative patterns depends on the application and a task in hand. There are at least three possible measures that can be used to sort the patterns: *frequency*, i.e., on how many lines the pattern exists in a data set; *perfectness*, i.e., how big part of the line has been fixed in the pattern; and *coverage* of the pattern, i.e., how large part of the database is covered by the pattern. Coverage is a measure, which balances the trade-off between patterns that are short but whose frequency is high and patterns that are long but whose frequency is lower. Selection of the most informative patterns can also be based on the optimality w.r.t. coverage. It is possible that an expert wishes to see only n most covering patterns or most covering patterns that together cover more than $m\%$ of the data. Examples of optimality constraints are considered in [14, 15].

An interesting issue is the treatment of the patterns, whose perfectness is close to zero. It is often the case that the support of such a small pattern is almost entirely covered by supports of larger patterns, subset of which the small pattern is. The most interesting property of this kind of lines is the possibility to find those rare and exceptional entries that are not covered by any of the frequent patterns.

In the domain that we are working on, log entries of telecommunication applications, we have found out that coverage and perfectness are very good measures to find good and informative starting points for pattern and data browsing. This is probably because of the fact that if there are too many fields that have not fixed values, then the meaning of the entry is not clear and those patterns are not understandable for an expert. On the other hand, in those logs there are a lot of repeating patterns, whose coverage is high and perfectness is close to 100 percent.

4 Experiments

Our experiments were done with two separate log sets. The first of them was a firewall log that was divided into several files so that each file contained entries logged during one day. From this collection we selected logs of four days with which we executed the CLC method with different frequency thresholds. The purpose of this test was to find out how large a portion of the original log it is possible to cover with the patterns found and what the optimal value for the

Table 2. Summary of the CLC experiments with firewall data

	Firewall days																	
Sup	Day 1					Day 2					Day 3					Day 4				
	Freq	Clsd	Sel	Lines	%	Freq	Clsd	Sel	Lines	%	Freq	Clsd	Sel	Lines	%	Freq	Clsd	Sel	Lines	%
100	8655	48	5	5162	96.3	9151	54	5	15366	98.6	10572	82	7	12287	97.1	8001	37	4	4902	97.3
50	9213	55	6	5224	97.5	9771	66	7	15457	99.2	11880	95	11	12427	98.2	8315	42	5	4911	97.5
10	11381	74	12	5347	99.8	12580	88	12	15537	99.7	19897	155	19	12552	99.2	10079	58	8	4999	99.2
5	13013	82	13	5351	99.9	14346	104	14	15569	99.9	22887	208	20	12573	99.3	12183	69	10	5036	99.9
Tot				5358					15588					12656					5039	

frequency threshold would be. In Table 2, a summary of the experiment results is presented.

Table 2 shows, for each firewall daily log file, the number of frequent sets (*Freq*), closed sets (*Clsd*) derived from those, selected closed sets (*Sel*), the number of lines that the selected sets cover (*Lines*) and how big part of the log these lines are covering (*%*). The tests were executed with several frequency thresholds (*Sup*). The pattern selection was based on the coverage of each pattern.

As can be seen from the result, already with the rather high frequency threshold of 50 lines, the coverage percentage is high. With this threshold there were, e.g., only 229 (1.8%) lines not covered in the log file of day 3. This was basically because there was an exceptionally well distributed port scan during that day. Those entries were so fragmented that they escaped from the CLC algorithm, but were clearly visible when all the other information was taken away.

In Table 2, we also show the sizes of the different representations compared to each other. As can be seen, the reduction from the number of frequent sets to the number of closed sets is remarkable. However, by selecting the most covering patterns, it is possible to reduce the number of shown patterns to very few without losing the descriptive power of the representation.

Another data set that was used to test our method was an application log of a large software system. The log contains information about the execution of different application modules. The main purpose of the log is to provide information for system operation, maintenance and debugging. The log entries provide a continuous flow of data, not occasional bursts, which are typical for firewall entries. The interesting thing in the flow are the possible error messages that are rare and often hidden in the mass.

The size of the application log was more than 105 000 lines, which were collected during a period of 42 days. From these entries, with the frequency threshold of 1000 lines (about 1%), the CLC method was able to identify 13 interesting patterns that covered 91.5% of the data. When the frequency threshold was still lowered to 50 lines, the coverage rose up to 95.8%. With that threshold value, there were 33 patterns found. The resulting patterns, however, started to be so fragmented that they were not very useful anymore.

These experiments show the usefulness of the condensed representation of the frequent itemsets by means of the frequent closed itemsets. In a data set like a firewall log, it is possible to select only a few most covering of the found frequent closed sets and cover the majority of the data. After this bulk has been

removed from the log it is much easier for any human expert to inspect the rest of the log, even manually.

Notice also that the computation of our results has been easy. This is partly because of our test data sets reported here are not very large; the largest set being a little over 100 000 lines. However, in a real environment of a large corporation, the daily firewall logs might contain millions of lines and much more variables. The amount of data — the number of lines and the number of variables — will continue to grow in the future, when the number of service types, different services and their use will grow. The scalability of the algorithms that compute the frequent closed sets is quite good compared to the APRIORI approach: fewer data scans are needed and the search space can be drastically reduced in the case of dense data [12, 3, 5]. In particular, we have done preliminary testing with `ac-miner` designed by A. Bykowski [5]. It discovers free sets, from which it is straightforward to compute closed sets. These tests have shown promising results w.r.t. execution times. This approach seems to scale up more easily than the search for a whole set of frequent sets.

Also, other condensed representations have been recently proposed like the δ-free sets, the $\vee$-free sets or the Non Derivable Itemsets [5, 7, 8]. They could be used in even more difficult contexts (very dense and highly-correlated data). Notice however, that from the end user point of view, these representations do not have the intuitive semantics of the closed itemsets.

5 Conclusions and Future Work

The Comprehensive Log Compression (CLC) method provides a powerful tool for any analysis that inspects data with lot of redundancy. Only very little a priori knowledge is needed to perform the analysis: knowledge structures: only a minimum frequency threshold for the discovery of closed sets and e.g., the number of displayed patterns, to guide the selection of the most covering patterns.

The method provides a mechanism to separate different information types from each other. The CLC method identifies frequent repetitive patterns from a log database and can be used to emphasize either the normal course of actions or exceptional log entries or events in the normal course of actions. This is especially useful in getting knowledge out of previously unknown domains or in analyzing logs that are used to record unstructured and unclassified information.

In the future we are interested in generalizing and testing the described method with frequent episodes: how to utilize relations between selected closed sets. Other interesting issues concern the theoretical foundations of the CLC method as well as ways to utilize this method in different real world applications.

Acknowledgements

The authors have partly been supported by the Nokia Foundation and the *consortium on discovering knowledge with Inductive Queries* (*cInQ*), a project

funded by the Future and Emerging Technologies arm of the IST Programme (Contract no. IST-2000-26469).

References

[1] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *SIGMOD'93*, pages 207–216, Washington, USA, May 1993. ACM Press. 361

[2] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press, 1996. 361, 365

[3] Jean-François Boulicaut and Artur Bykowski. Frequent closures as a concise representation for binary data mining. In *PAKDD'00*, volume 1805 of *LNAI*, pages 62–73, Kyoto, JP, April 2000. Springer-Verlag. 361, 365, 368

[4] Jean-François Boulicaut, Artur Bykowski, and Christophe Rigotti. Approximation of frequency queries by mean of free-sets. In *PKDD'00*, volume 1910 of *LNAI*, pages 75–85, Lyon, F, September 2000. Springer-Verlag. 366

[5] Jean-François Boulicaut, Artur Bykowski, and Christophe Rigotti. Free-sets: a condensed representation of boolean data for the approximation of frequency queries. *Data Mining and Knowledge Discovery journal*, 7(1):5–22, 2003. 366, 368

[6] Ronald J. Brachman and Tej Anand. The process of knowledge discovery in databases: A first sketch. In *Advances in Knowledge Discovery and Data Mining*, July 1994. 361

[7] Artur Bykowski and Christophe Rigotti. A condensed representation to find frequent patterns. In *PODS'01*, pages 267 – 273. ACM Press, May 2001. 368

[8] Toon Calders and Bart Goethals. Mining all non derivable frequent itemsets. In *PKDD'02*, volume 2431 of *LNAI*, pages 74–83, Helsinki, FIN, August 2002. Springer-Verlag. 368

[9] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27 – 34, November 1996. 361

[10] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery: An overview. In *Advances in Knowledge Discovery and Data Mining*, pages 1 – 34. AAAI Press, Menlo Park, CA, 1996. 361

[11] R. Kosala and H. Blockeel. Web mining research: A survey. *SIGKDD: SIGKDD Explorations: Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining, ACM*, 2(1):1–15, 2000. 361

[12] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25–46, January 1999. 361, 365, 366, 368

[13] Jian Pei, Jiawei Han, and Runying Mao. CLOSET an efficient algorithm for mining frequent closed itemsets. In *SIGMOD Workshop DMKD'00*, Dallas, USA, May 2000. 361

[14] Tobias Scheffer. Finding association rules that trade support optimally against confidence. In *PKDD'01*, volume 2168 of *LNCS*, pages 424–435, Freiburg, D, September 2001. Springer-Verlag. 366

[15] Jun Sese and Shinichi Morishita. Answering the most correlated N association rules efficiently. In *PKDD'02*, volume 2431 of *LNAI*, pages 410–422, Helsinki, FIN, August 2002. Springer-Verlag. 366

[16] Jaideep Srivastava, Robert Cooley, Mukund Deshpande, and Pang-Ning Tan. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explorations*, 1(2):12–23, 2000. 361

[17] Mohammed Javeed Zaki. Generating non-redundant association rules. In *SIGKDD'00*, pages 34–43, Boston, USA, August 2000. ACM Press. 361