Summarising Data by Clustering Items

Michael Mampaey and Jilles Vreeken

Department of Mathematics and Computer Science
Universiteit Antwerpen
{michael.mampaey,jilles.vreeken}@ua.ac.be

Abstract. For a book, the title and abstract provide a good first impression of what to expect from it. For a database, getting a first impression is not so straightforward. While low-order statistics only provide limited insight, mining the data quickly provides too much detail. In this paper we propose a middle ground, and introduce a parameter-free method for constructing high-quality summaries for binary data. Our method builds a summary by grouping items that strongly correlate, and uses the Minimum Description Length principle to identify the best grouping —without requiring a distance measure between items.
Besides offering a practical overview of which attributes interact most strongly, these summaries are also easily-queried surrogates for the data. Experiments show that our method discovers high-quality results: correlated attributes are correctly grouped and the supports of frequent itemsets are closely approximated.

1 Introduction

When handling a book, and wondering about its contents, we can simply start reading it from A to Z. In practice, however, to get a good first impression we usually first refer to the summary. For a book, this can be anything from the title, the abstract, up to simply paging through it. The common denominator here is that a summary quickly provides high-quality and high-level information about the book. A summary may already contain exactly what we were looking for, but in general we expect to get enough insight to judge what the book contains and whether we need to read it further.

When handling a transaction database, and wondering about its content and whether (or how) we should analyse it, it is quite hard to get a good first impression. Of course, one can inspect the schema of the database, and the attribute labels will also convey some information. However, these do not provide an overview of what is *in* the database. To this end, basic statistics can help to a limited extent, e.g. first order statistics tell us which items occur often, and which do not. For binary transaction databases, however, further basic statistics are not readily available. Ironically, this means that while the goal is to get a first impression, we have to analyse the data in detail. For non-trivially sized databases especially, this means investing far more time and effort than we should at this stage of the analysis.

When analysing data, a good first impression of the data is important, as mining data is essentially an iterative process [9], where each step provides extra insight, which allows us to extract increasingly more knowledge. A good summary allows us to make a well-informed decision on what basic assumptions to make and how to mine the data.

Here, we propose a simple and parameter-free method for providing high-quality summary overviews for binary transaction data. The outcome provides insight into which attributes are most correlated and in what value-configurations these occur. They are probabilistic models of the data that can be queried fast and accurately, allowing them to be used instead of the data. Further, by showing which attributes interact most strongly, these summaries can provide insight for selecting and constructing features. In short, like a proper summary, they provide a good first impression and can be used as a surrogate.

To the best of our knowledge, there currently do not exist light-weight data analysis methods that can be easily used for summary purposes. Instead, for binary data the standard approach is to mine for frequent itemsets first, the result of which quickly grows up to many times the size of the original data. Resulting, many proposals exist that focus on summarising sets of frequent patterns. That is, to choose groups of representative itemsets such that the information in the complete pattern set is maintained as well as possible. Here, we do not summarise the outcome of an analysis, i.e. a set of patterns, but instead provide a summary which can be used to decide how to further analyse the data.

Existing proposals for data summarisation, such as Krimp [17] and Summarization [3], provide highly detailed results. Although this has obvious merit, analysing these summaries consequently also requires significant effort. Our method shares the approach of using compression to find a good summary. However, we do not aim at finding a group of descriptive itemsets. Instead, we view the data symmetrically with regard to 0s and 1s and aim to optimally group those items that interact most strongly. In this regard, our approach is also related to selecting low-entropy sets [10], itemsets that identify strong interactions in the data. An existing proposal to this end, LESS [11], requires a collection of low-entropy sets as input, and the resulting model cannot easily be queried. For a more complete discussion of related work, please refer to Section 5.

The method we propose in this paper groups attributes that interact strongly, i.e. that have low entropy. We identify the best grouping through the Minimum Description Length principle; no parameter needs to be set by the user. No distance measure between attributes is required, and the similarity between clusters is easily calculated. Experiments show that our method discovers high-quality results: correlated attributes are correctly grouped, representative features are identified, and the supports of itemsets are closely approximated.

The roadmap of this paper is as follows. First, we introduce notation and formalise the problem. In Section 3 we introduce our method for finding good summarisations and how information can be extracted from these in Section 4. Related work is discussed in Section 5. We experimentally evaluate our method in Section 6. We round up with a discussion in Section 7 and conclude in Section 8.

2 MDL for Attribute Clustering

In this section we formally introduce our method. We start by covering the preliminaries and notation, then define what an attribute clustering is, and how to use MDL to identify good clusterings.

2.1 Preliminaries

We denote the set of all items by $\mathcal{I} = \{I_1, \ldots, I_n\}$. A dataset $\mathcal{D}$ is a bag of transactions t. A transaction is a binary vector of length n. An *item* is a binary attribute, that is, a pair $(I = v)$, where $I \in \mathcal{I}$ and $v \in \{0, 1\}$. Then, an *itemset* is simply a pair $(X = v)$, where $X \subseteq \mathcal{I}$ is a set of items, and $v \in \{0, 1\}^{|X|}$ is a binary vector of length $|X|$. Sometimes we will also refer to a set of attributes as an itemset. A transaction t is said to contain an itemset $X = v$, denoted as $X \subset t$, if for all items $x_i \in X$ it holds that $t_i = v_i$. The *support* of $X = v$ is the number of transactions in $\mathcal{D}$ that contain $X = v$, i.e. $\mathit{supp}(X) = |\{t \in \mathcal{D} \mid X \subset t\}|$. The *frequency* of $X = v$ is defined as its support, divided by the size of $\mathcal{D}$, i.e. $\mathit{freq}(X = v) = \mathit{supp}(X = v)/|\mathcal{D}|$. The entropy of an itemset X over $\mathcal{D}$ is defined as $H(X) = -\sum_v \mathit{freq}(X = v) \log \mathit{freq}(X = v)$, where the logarithm is to base 2.

2.2 Definitions

The summaries we use are based on attribute clusterings. Therefore, we first formally introduce the concept of an attribute clustering.

Definition 1. *An attribute clustering $\mathcal{A} = \{A_1, \ldots, A_k\}$ of a set of items $\mathcal{I}$ is a partition of $\mathcal{I}$, where*

1. *each cluster is not empty: $\forall A_i \in \mathcal{A} : A_i \neq \emptyset$,*
2. *all clusters are pairwise disjoint: $\forall i \neq j : A_i \cap A_j = \emptyset$,*
3. *every item belongs to a cluster: $\bigcup_i A_i = \mathcal{I}$.*

Next, we must define what the *best* attribute clustering is. For this, we use the *Minimum Description Length* principle (MDL). This principle [7] can be roughly described as follows. Given a set of models $\mathcal{M}$ for $\mathcal{D}$, the best model $M \in \mathcal{M}$ is the one that minimises

$$L(\mathcal{D} \mid M) + L(M) ,$$

where $L(M)$ is the length, in bits, of the description of the model, and $L(\mathcal{D} \mid M)$ is the length of the description of the data when encoded by the model. To use MDL, we need to define how to encode a model, and how it describes the data.

First, let us determine how to describe the attribute clustering. To begin with, we must state how many items there are, and then, for each item we describe to which cluster it belongs. In this description there is some redundancy, since any permutation of the cluster labels yields an equivalent partition. Taking this into account, the description of the partition requires $\log n + n \log k - \log k!$ bits.

CT_i			
a b c	$code(v)$	$freq(v)$	$L(code(v))$
1 1 1	0	50%	1
1 1 0	10	25%	2
1 0 1	110	12.5%	3
0 1 0	1110	6.25%	4
0 0 0	1111	6.25%	4

Fig. 1. Example of a code table CT_i for the cluster $A_i = \{a, b, c\}$. The frequencies are not actually part of the code table, they are merely included as illustration. Moreover, the specific codes are examples—in our computations we are not interested in materialised codes, only in their lengths, $L(code(v))$.

Secondly, we use *code tables* to describe the distribution of each cluster. Let $A_i \in \mathcal{A}$ be an attribute cluster, then the code table CT_i for A_i describes which itemset values occur in the data with respect to this cluster, together with their codes. That is, a code table for a cluster A_i is a two-column table with value-assignments $v \in \{0,1\}^{|A_i|}$ for A_i on the left-hand side, and the corresponding codes on right-hand side. Figure 1 shows an example of a code table.

The values v can best be described as strings of bits, so their length is simply $|A_i|$. A well-known result from information theory [4] states that the code lengths for $A_i = v$ are optimal when $L(code(v)) = -\log freq(v)$. Note that we are not interested in actual materialised codes (e.g. computed through Huffman coding [4]), but only in their lengths. If a certain v has a frequency of 0, that is, it does not occur in the data, then we do not record it in the code table. Hence, the description length of the code table of a cluster A_i can be computed as $L(CT_i) = \sum_{v; freq(v)\neq 0} |A_i| - \log freq(v)$.

Finally, we need to define how to compute $L(\mathcal{D} \mid \mathcal{A})$, the length of the encoded description of database $\mathcal{D}$ given the clustering $\mathcal{A}$. Each transaction $t \in \mathcal{D}$ is partitioned according to $\mathcal{A}$, and encoded using the optimal codes found in the code tables. Since an itemset $X = v$ is used $|\mathcal{D}| \cdot freq(X = v)$ times, the total encoded size of $\mathcal{D}$ with respect to a single cluster A_i can be written as $L(\mathcal{D}_{A_i} \mid \mathcal{A}) = -\sum_t \log freq(t) = -|\mathcal{D}| \cdot \sum_v freq(v) \log freq(v) = |\mathcal{D}| \cdot H(A_i)$.

Putting this all together, we define the total encoded size $L(\mathcal{A}, \mathcal{D})$ as follows.

Definition 2. *The total description length of a clustering $\mathcal{A} = \{A_i\}_{i=1}^k$ of size k for a dataset $\mathcal{D}$ is:*

$$L(\mathcal{A}, \mathcal{D}) = L(\mathcal{D} \mid \mathcal{A}) + L(\mathcal{A}),$$

where

$$\begin{cases} L(\mathcal{D} \mid \mathcal{A}) &= |\mathcal{D}| \cdot \sum_{i=1}^k H(A_i) \\ L(\mathcal{A}) &= \log n + n \log k - \log k! + \sum_{i=1}^k L(CT_i) \\ L(CT_i) &= \sum_{v; freq(v)\neq 0} |A_i| + L(code(v)) \\ L(code(v)) &= -\log freq(v) \end{cases}$$

2.3 Problem Definition

Our goal is to discover an optimal summary of a transaction database. A summary consists of a partitioning of the attributes of a binary transaction database; it must be optimal in the sense that the attribute groups should be relatively independent, while the individual clusters should exhibit strong correlations on the data, as clusters with a lot of structure can be described succinctly.

Formally, the problem we address is as follows. Given a transaction database $\mathcal{D}$ over a set of binary attributes $\mathcal{I}$, find the attribute clustering $\mathcal{A}$ that minimises

$$L(\mathcal{A}, \mathcal{D}) = L(\mathcal{D} \mid \mathcal{A}) + L(\mathcal{A})\,.$$

With this problem statement we let MDL decide what the optimal number of attribute clusters is, by choosing the clustering that minimises the number of bits required to describe the model and the data. This also ensures that two unrelated groups of attributes will not be combined into one, as it will be far cheaper to describe the two groups separately.

The search space we have to consider for our problem is rather large. The total number of possible partitions of a set of n elements is known as the Bell number B_n, which is at least $\Omega(2^n)$. Therefore we cannot simply enumerate and test all possible partitions for a non-trivial dataset. We must traverse the search space and exploit its structure to arrive at a good clustering. The *refinement* of partitions naturally structures the search space into a lattice. A partition $\mathcal{A}$ refines a partition $\mathcal{A}'$ if for all $A \in \mathcal{A}$ there exists $A' \in \mathcal{A}'$ such that $A \subseteq A'$. The minimal clustering with respect to refinement contains a cluster for each individual item. We will call this the *independence clustering*. The transitive reduction of the refinement relation corresponds to merging two clusters, and this is how we will traverse the search space. Note that $L(\mathcal{A}, \mathcal{D})$ is not (anti-)monotonic with respect to refinement, otherwise the best clustering would simply be $\{\mathcal{I}\}$ or $\{\{I\} \mid I \in \mathcal{I}\}$, respectively. Further, this means there is no structure we can exploit to efficiently find the optimal clustering.

2.4 Measuring Cluster Similarity

Instead of requiring the user to specify a distance metric for individual items, we can derive a similarity measure between clusters from our definition of $L(\mathcal{A}, \mathcal{D})$.

Let $\mathcal{A}$ be an attribute clustering and let $\mathcal{A}'$ be the result of merging the clusters A_i and A_j in $\mathcal{A}$. In other words, $\mathcal{A}$ is a refinement of $\mathcal{A}'$. Then the difference of description lengths defines a similarity measure between A_i and A_j.

Definition 3. *The similarity of two clusters A_i and A_j in $\mathcal{A}$ is defined as*

$$CS_{\mathcal{D}}(A_i, A_j) = L(\mathcal{A}, \mathcal{D}) - L(\mathcal{A}', \mathcal{D}),$$

where $\mathcal{A}' = \mathcal{A} \setminus \{A_i, A_j\} \cup \{A_i \cup \mathcal{A}_j\}$.

If A_i and A_j are very similar, i.e. have a low joint entropy, then this merger improves the total description length, meaning $CS_{\mathcal{D}}(A_i, A_j)$ will be positive, otherwise it is negative. Note that this similarity is local, in that it is not influenced by the other clusters in $\mathcal{A}$. This is further supported by the following lemma, which allows us to compute cluster similarity without having to compute the entire cluster description length. For the sake of exposition, we here ignore the cluster description term $\log n + n \log k - \log k!$, which is not a dominating term for the total description length.

Lemma 1. *Let $\mathcal{A}$ be an attribute clustering of $\mathcal{I}$, with $A_i, A_j \in \mathcal{A}$. Then*

$$CS_{\mathcal{D}}(A_i, A_j) = |\mathcal{D}| \cdot I(A_i, A_j) + \Delta L(CT),$$

where $I(A_i, A_j) = H(A_i) + H(A_j) - H(A_i A_j)$ is the mutual information between A_i and A_j, and $\Delta L(CT) = L(CT_i) + L(CT_j) - L(CT_{ij})$.

Lemma 1 shows that we can decompose cluster similarity into a mutual information term, and a term expressing the difference in code table size. Both of these are high when the attributes in A_i and A_j are highly correlated.

3 Mining Attribute Clusterings

As detailed above, the search space we have to consider is extremely large, and there exists no structure we can exploit to find the optimum. Hence, we have to settle for heuristics. In this section we introduce our algorithm, which finds a good attribute clustering $\mathcal{A}$ with a low description length $L(\mathcal{A}, \mathcal{D})$.

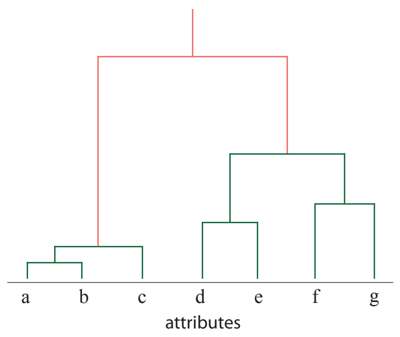
Since we do not employ a distance metric between the attributes, the problem is not as easy as simply applying an existing clustering algorithm such as k-means [13]. Instead, we use a greedy bottom-up clustering algorithm, which iteratively merges clusters by selecting those two clusters whose union has the shortest description. This results in a hierarchy of clusters, which can be represented visually as a dendrogram, as shown on the left hand side of Figure 2. At the bottom we have the independence distribution and at the top the joint empirical distribution of the data. An advantage of this approach is that we can so visualise how the clusters were formed.

The pseudo-code is given in Algorithm 1. We start by placing each item in its own cluster (line 1), which corresponds to the independence model. Then, we iteratively find the two clusters with the highest similarity (4), an merge them (5). In other words, in each iteration the algorithm tries to reduce the total description length as much as possible. If a merge reduces the lowest description length seen yet, we remember it (6-7), and finally return the best clustering (10).

The graph on the right hand side of Figure 2 shows how the description length behaves during the course of the algorithm on the *Pen Digits* dataset. Starting at $k = n$, the description length $L(\mathcal{A}, \mathcal{D})$ gradually decreases as similar clusters are being merged. This indicates that there is some definite structure present in the data. It continues to decrease until $k = 5$, which yields the best clustering found for this dataset. After this, the description length of the code tables increases dramatically, which implies that no more structure is present.

Algorithm 1 AttributeClustering

```
Input: A transactional dataset $\mathcal{D}$ over a set of items $\mathcal{I}$.
Output: A clustering of the items $\mathcal{A} = \cup_{i=1}^{k} A_i$.
 1. $\mathcal{A} \leftarrow \{\{I\} \mid I \in \mathcal{I}\}$
 2. $\mathcal{A}_{min} \leftarrow \mathcal{A}$
 3. while $|\mathcal{A}| > 1$ do
 4.    $A_i, A_j \leftarrow \text{argmax}_{i,j} CS_{\mathcal{D}}(A_i, A_j)$
 5.    $\mathcal{A} \leftarrow \mathcal{A} \setminus \{A_i, A_j\} \cup \{A_i \cup A_j\}$
 6.    if $L(\mathcal{A}, \mathcal{D}) < L(\mathcal{A}_{min}, \mathcal{D})$ then
 7.       $\mathcal{A}_{min} \leftarrow \mathcal{A}$
 8.    end if
 9. end while
10. return $\mathcal{A}_{min}$
```



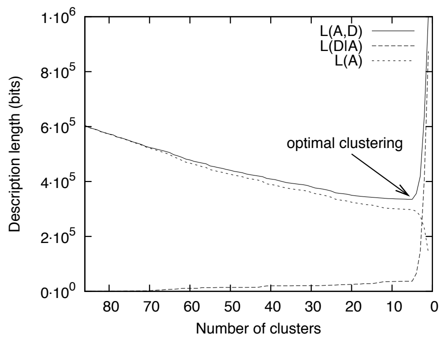


Fig. 2. (left) Example dendrogram. Merges that save bits are depicted in green (dark grey), merges that cost bits are red (light grey). Here the optimal $k = 2$. (right) Evolution of the encoded length $L(\mathcal{A}, \mathcal{D})$ with respect to the number of clusters k, on the *Pen Digits* dataset. The optimum is at $k = 5$.

3.1 Convexity

Figure 2 seems to suggest that the description length evolves convexly with respect to k. That is, there is a single local minimum, and once $L(\mathcal{A}, \mathcal{D})$ starts to increase, there are no more steps in which the description length decreases. Naturally, the question arises whether this is the case in general; if so, the algorithm can terminate as soon as a local minimum is detected. Intuitively, we would expect that if the currently best cluster merge increases the total description length, then all other merges are even worse, and we expect the same from all future merges. However, the following example shows that this is not the case.

Consider a dataset $\mathcal{D}$ with $\mathcal{I} = \{a, b, c, d\}$. Let us assume that for the transactions of $\mathcal{D}$ it holds that $d = a \oplus b \oplus c$, where $\oplus$ denotes exclusive or. Now, using this dependency, let $\mathcal{D}$ contain a transaction for every $v \in \{0, 1\}^3$ as values for abc. Then, every pair of clusters whose union contains up to three items (e.g. $A_i = ab$ and $A_j = d$) is independent. It is clear that as the algorithm starts to merge clusters, the entropy remains constant, and the code tables become more complex and thus $L(\mathcal{A}, \mathcal{D})$ increases. Only at the last step, when the two last clusters are merged, the dependency is recognised and the entropy drops, leading to a decrease of the total encoded size. Hence, the total description length $L(\mathcal{A}, \mathcal{D})$ is non-convex with respect to to cluster merges.

Note, however, that the gain in encoded length depends on the number of transactions in the database. In the above example, if every unique transaction occurs 20 times, the complete clustering would be preferred over the independence model. However, if there are fewer transactions (say, every transaction occurs four times), then while the dependencies are the same, the algorithm decides that the best clustering corresponds to the independence model (i.e. there is no significant structure). Intuitively this can be explained by the fact that if there are only few samples then the observed dependencies might be coincidental, but if many transactions follow it, the dependencies are truly present. This is one of the nice properties we get from using MDL to identify the best model.

While this example shows that in general we should not stop the algorithm at a local minimum, it is a very synthetic example with a strong requirement on the number of transactions. For instance, if we generalise the XOR example to 20 attributes, the minimum number of transactions for it to be detectable is already larger than 20 million. Furthermore, in none of our experiments with real data did we encounter a local minimum which was not also a global minimum. Therefore, we can say that in practice it is acceptable to stop the algorithm at a local minimum.

3.2 Algorithmic Complexity

Naturally, a summarisation method should be fast, because of our goal to get a quick overview of the data. For complex data mining algorithms on the other hand, it is often found acceptable that they are exponential. Here we show that our algorithm is polynomial in the number of attributes.

In the first iteration, we compute the description length for each singleton cluster $\{I\}$, and then determine which clusters to merge. To do this, we must compute $O(n^2)$ cluster similarities, where $n = |\mathcal{I}|$. Since we might need some of the similarities later on, we store them in a heap, such that we can easily retrieve the maximum. Now say that in a subsequent iteration k we have just merged A_i and A_j into A_{ij}. Then we delete $2k-1$ similarities from the heap, and compute and insert $k-1$ new similarities, i.e. between A_{ij} and the remaining clusters. Since heap insertion and deletion is logarithmic, maintaing the similarities in one iteration takes $O(k \log k)$ time. The computation of the similarities $CS_{\mathcal{D}}(A_i, A_j)$ requires collecting all nonzero frequencies $\mathit{freq}(A_{ij} = v)$, and we do this by simply iterating over all transactions t and computing $A_{ij} \cap t$, which takes $O(n|\mathcal{D}|)$ time. In total, the time complexity of our algorithm is $O(n^2 \log n \times n|\mathcal{D}|)$.

The biggest cost in terms of storage are the cluster similarities, and hence the memory complexity is $O(n^2)$.

4 Querying a Summary

Besides providing a general overview of which attributes interact most strongly, and in which value-assignments they typically occur, our summaries can also be used as surrogates for the data. That is, we can query a summary. For binary data, a query comes down to calculating marginals: counting how often a particular value-assignment occurs, in other words, determining supports.

The frequency of an itemset (or conjunctive query) can be estimated from an attribute clustering, by assuming that the clusters are independent. By MDL, we know this is a safe assumption: if two clusters A_i and A_j were dependent, it would have been far cheaper to combine them into a single cluster A_{ij}.

Let $\mathcal{A} = \{A_i\}_{i=1}^k$ be a clustering of $\mathcal{I}$, and let $X \subset \mathcal{I}$ be an itemset. Then the frequency of X can be estimated as

$$\hat{freq}(X) = \prod_{i=1}^{k} freq(X \cap A_i)$$

As an example, let $\mathcal{I} = \{a, b, c, d, e, f\}$ and $\mathcal{A} = \{abc, de, f\}$, and let $X = \{a, b, e, f\}$, then $\hat{freq}(X) = freq(ab) \cdot freq(e) \cdot freq(f)$.

As each CT_i implicitly contains the frequencies of the value-assignments for A_i, we can use our clustering models as very efficient surrogates for $\mathcal{D}$.

5 Related Work

The main goal of this proposal is to offer a good first impression of the data. For numerical data, averages and correlations can easily be computed, and more importantly, are informative. For binary transaction data such informative statistics are not readily available. As such, our work can be seen as to provide an informative 'average' for binary data; for those attributes that interact strongly, it shows how often the value-assignments occur.

Most existing techniques for summarisation are aimed at giving a succinct representation of a given collection of itemsets. Well-known examples include closed itemsets [15] and non-derivable itemsets [2], which both provide a lossless reduction of the complete collection. A lossy approach that provides a succinct summary of the patterns was proposed by Yan et al. [20]. Experiments show our method provides better frequency estimates, while requiring fewer 'profiles'. Wang and Karypis gave a method [19] for directly mining a summary of the frequent pattern collection for a given *minsup* threshold. Please refer to [8] for a more complete overview of pattern mining and summarisation techniques.

For summarising data fewer proposals exist. Chandola and Kumar [3] propose to induce k transaction templates such that the database can be reconstructed with minimal loss of information. Alternatively, the Krimp algorithm [17] selects those itemsets that provide the best lossless compression of the database, i.e. the best description. While it only considers the 1s in the data, it provides high-quality and detailed results, which are consequently not as small and easily interpreted as our summaries. Though the KRIMP code tables can generate data virtually indistinguishable from the original [18], they are not probabilistic models and cannot be queried directly, they are no surrogate for the data.

Most related to our method are low-entropy sets [10], itemsets for which the entropy of the data is below a given threshold. As entropy is strongly monotonically increasing, typically very many low-entropy sets are discovered even for low thresholds. Heikinheimo et al. introduced a filtering proposal [11], LESS, to select those low-entropy sets that together describe the data well. Here, instead of filtering, we discover itemsets with low entropy directly on the data.

Orthogonal to our approach, the maximally informative k-itemsets (miki's) by Knobbe and Ho [12] are k items (or patterns) that together split the data optimally, found through exhaustive search. Bringmann and Zimmermann [1] proposed a greedy alternative to this exhaustive method that can consider larger sets of items. We group items together that correlate strongly, so the correlations between groups are weak. As future work, we plan to investigate whether good approximate miki's can be extracted from our summaries.

As our approach employs clustering, the work in this field is not unrelated. However, clustering is foremost concerned with grouping rows together, typically requiring a distance measure between objects. Bi-clustering [16] is a type of clustering in which clusters are detected over both attributes and rows. In our setup we only group attributes, not rows, and do not require a distance measure between items.

6 Experiments

In this section we experimentally evaluate our method and validate the quality of the returned summaries.

6.1 Setup

We implemented our algorithm in C++, and provide the source code for research purposes[1]. All experiments were executed on an quad-core Intel Xeon machine with 6GB of memory, running Linux.

We evaluate our method on three synthetic datasets, as well as on seven publicly available real-world datasets. Their basic characteristics are given in Table 1. The *Independent* data has independent attributes with random frequencies. In *Markov* each item is a copy of the previous one with a random probability. The *DAG* dataset is generated according to a directed acyclic graph among the items. An item depends on a small amount of preceding items, the probabilities in the corresponding contingency table are generated at random.

The *Accidents*, *BMS-Webview-1*, *Chess*, *Connect*, and *Mushroom* datasets were obtained from the FIMI dataset repository[2] and the *Pen Digits* data was

[1] `http://www.adrem.ua.ac.be/implementations/`
[2] `http://fimi.cs.helsinki.fi/data/`

Table 1. Results of our Attribute Clustering algorithm for 3 synthetic and 7 real datasets. As basic statistics per dataset, shown are the number of binary attributes, and the number of transactions. For the result of our method, shown are the number of identified groups, the attained compression ratio relative to the independence model, and the wall-clock time used to generate the summary.

	Basic Statistics		*Attribute Clustering*		
Dataset	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{D}\rvert$	k	$\frac{L(\mathcal{A},\mathcal{D})}{L(\mathcal{I},\mathcal{D})}$	*time*
Independent	50	20000	50	100%	3 s
Markov	50	20000	14	89.6%	5 s
DAG	50	20000	12	95.7%	6 s
Accidents	468	340183	199	64.7%	165 m
BMS-Webview-1	497	59602	150	89.6%	434 s
Chess	75	3196	9	40.8%	2 s
Connect	129	67557	7	43.4%	182 s
DNA Amplification	391	4950	52	42.0%	22 s
Mushroom	119	8124	5	37.9%	14 s
Pen Digits	86	10992	5	55.4%	9 s

obtained from the LUCS-KDD data library[3]. Further, we use the *DNA Amplification* database, which contains data on DNA copy number amplifications. Such copies activate oncogenes and are hallmarks of nearly all advanced tumors [14]. Amplified genes represent targets for therapy, diagnostics and prognostics.

6.2 Evaluation

In Table 1 we are interested in k, the number of clusters our algorithm finds, and what the total compressed size $L(\mathcal{A},\mathcal{D})$ is, relative to the independence clustering $L(\mathcal{I},\mathcal{D})$. A low number of clusters and a short description length indicate that our algorithm models structure present in the data. The algorithm correctly detects 50 clusters in the *Independent* data, even though there might seem some accidental dependencies present due to the randomness of the data generation. For the other datasets we see that the number of clusters k is much lower than the number of items $|\mathcal{I}|$. As such, it is perfectly feasible to inspect these clusters by hand. Many of the datasets are highly structured, which can be seen from the strong compression ratios the clusterings achieve.

In Table 2 we test whether the clustering that our algorithm finds actually reflects true structure in the data, rather then just finding some random artifacts. For each dataset $\mathcal{D}$ we create 1000 swap randomised datasets $\mathcal{D}_S$ [6], and run our algorithm. These datasets have the same row and column margins as the original data, but are random otherwise. Only patterns depending on the margins are therefore retained. We see that in all cases all structure disappears,

[3] `http://www.csc.liv.ac.uk/~frans/KDD/`

Table 2. Results for the randomisation experiments. The second and third columns are the averaged results of our algorithm on 1000 swap randomised datasets (100 for *BMS-Webview-1* and 20 for *Accidents*). The number of swaps is equal to the number of ones in the data as suggested in [6]. The fourth column is the average total description length for 1000 random k-partitions.

	Swap Randomisation		*Random k-partition*
Dataset	k_{swap}	$\frac{L(\mathcal{A}_s,\mathcal{D}_s)}{L(\mathcal{I},\mathcal{D}_s)}$	$\frac{L(\mathcal{A}_r,\mathcal{D})}{L(\mathcal{I},\mathcal{D})}$
Independent	49.95 ± 0.22	$100.0\% \pm 0.0$	$100.0\% \pm 0.0$
Markov	49.93 ± 0.88	$100.0\% \pm 0.0$	$99.5\% \pm 0.0$
DAG	49.92 ± 0.59	$100.0\% \pm 0.0$	$100.4\% \pm 1.1$
Accidents	432.7 ± 28.6	$99.9\% \pm 0.2$	$99.7\% \pm 0.3$
BMS-Webview-1	339.1 ± 4.32	$99.6\% \pm 0.0$	100.0 ± 0.0
Chess	73.36 ± 2.00	$99.9\% \pm 0.1$	$94.5\% \pm 3.6$
Connect	100.8 ± 3.64	$100.0\% \pm 0.2$	$90.6\% \pm 2.0$
DNA Amplification	348.9 ± 3.18	$99.8\% \pm 0.0$	$104.9\% \pm 0.0$
Mushroom	114.8 ± 2.13	$100.0\% \pm 0.0$	$67.7\% \pm 2.3$
Pen Digits	80.47 ± 7.56	$100.0\% \pm 0.0$	$102.6\% \pm 3.5$

and the average number of clusters our algorithm returns is very close to the number of attributes, i.e. the best clustering is close to the independence clustering. Furthermore, we also see that the average description length is basically the same as for the independence clustering. For each dataset, we also created 1000 random partitions, $\mathcal{A}_r$, of k groups. The last column in Table 2 shows the average description length compared to $L(\mathcal{I}, \mathcal{D})$. We see that while for several of the datasets random partitions can still compress the data better than the independence clustering and hence model *some* structure, the gain is much lower than the compression gain that our algorithm attains.

Next, we investigate the actual clusterings discovered by our attribute clustering algorithm in closer detail.

For the synthetic data, we see that the embedded structures are correctly recovered. For *Independent* the algorithm of course returns the independence clustering. The items in the *Markov* dataset form a Markov chain, and the clusters found by our algorithm contain adjacent items, i.e. they are Markov chains themselves. Interestingly, when regarding its dendrogram (not shown), we see that the chains are split up at exactly those places where the dependency between items is low, i.e. the copy probability is close to 50%. Likewise, in the *DAG* dataset, which has attribute dependencies forming a directed acyclic graph, the clusters contain items which form tightly linked groups in the graph.

The *DNA Amplification* dataset is an approximately *banded* dataset [5], that is, the majority of the ones form a staircase pattern, and are located in blocks along the diagonal. In Figure 3, a submatrix of the data is plotted, along with the attribute clustering our algorithm finds. The clustering clearly distinguishes the blocks in the data. In turn, these blocks correspond to related oncogenes.

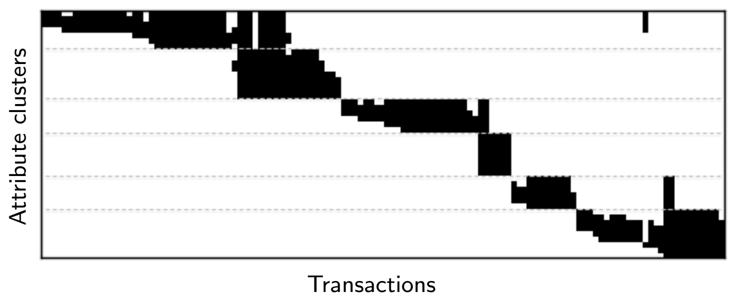


Fig. 3. A (transposed) submatrix of the *DNA Amplification* data and the corresponding discovered attribute clusters, separated by the dotted lines.

The *Connect* dataset contains all legal 8-ply positions of the well-known Connect Four game. The game has 7 columns and 6 rows, and for each of the 42 squares, an attribute describes whether it is blank, or which one of the two players has positioned a chip there. Furthermore, a class label describes which player can win or whether the game will result in a draw. The dataset we use is binary, and contains an item for each possible attribute-value pair, as well as an item for each class label. First of all, we see that all attribute-value pairs (items) originating from a single attribute (i.e. location) are grouped into the same cluster. Furthermore, our algorithm discovers 7 clusters. Each one of these clusters correctly corresponds to a column in the game, i.e. the structure found by our algorithm reflects the physical structure of the game. The class label is placed in the cluster of the middle column; this makes a lot of sense since any horizontal or diagonal line of four must pass through the middle column, and hence this column is key for winning the game.

6.3 Estimating Itemset Frequencies

In this subsection we investigate how well our summaries can be used to estimate itemset frequencies. For each dataset we first mine up to the top-10 000 closed frequent itemsets. Then, for each itemset in this collection, we estimate its frequency according to our model and compute both its absolute and relative error. For comparison the same is done for the independence model, which is equivalent to the singleton clustering. As can be seen from the results in Table 3, the models returned by our algorithm allow for very good frequency estimates; for most datasets the average absolute error is less than 1% and much better than that for the independence model. While for the *BMS-Webview-1*, *DNA Amplification* and *Pen Digits* datasets the average relative error seems rather high (50%), this is explained by the fact that the frequencies for the top-10 000 closed itemsets for these datasets are very low, as can be read from the first column.

In Figure 4 we plot the cumulative probability of the absolute errors for *Connect* and *Mushroom*. For every $\epsilon \in [0, 1]$ we determine the probability $\delta = p(err > \epsilon)$ that the absolute estimation error $|freq(X) - \hat{freq}(X)|$ is greater than ϵ.

Table 3. Results for frequency estimation of the top-10 000 closed frequent itemsets. Depicted are the average frequency in the original data, the average absolute and relative errors of the frequency estimates using our model (third and fourth column) and using the independence model (fifth and last column).

		Attribute Clustering		*Independence Model*	
Dataset	$\overline{freq}$	$\lvert freq - \widehat{freq} \rvert$	$\frac{\lvert freq-\widehat{freq}\rvert}{freq}$	$\lvert freq - \widehat{freq} \rvert$	$\frac{\lvert freq-\widehat{freq}\rvert}{freq}$
Independent	29.0%	0.15%	0.54%	0.15%	0.54%
Markov	15.7%	0.30%	2.02%	1.36%	8.47%
DAG	20.9%	0.50%	2.51%	0.92%	4.69%
Accidents	55.8%	1.47%	2.74%	2.89%	5.38%
BMS-Webview-1	0.1%	0.09%	83.1%	0.10%	91.14%
Chess	81.2%	0.93%	1.16%	1.47%	1.83%
Connect	88.8%	0.38%	0.45%	2.56%	2.95%
DNA Amplification	0.5%	0.08%	53.24%	0.46%	92.25%
Mushroom	12.5%	1.30%	13.55%	5.48%	48.46%
Pen Digits	6.1%	2.89%	51.52%	3.86%	67.52%

For both datasets we see that the best clustering outperforms the independence clustering. For instance, in the *Mushroom* dataset we see that probability of an absolute error larger than 5% is about 50% for the independence model, while for our clustering method this is only 1%.

Lastly, we compare the frequency estimation capabilities of our attribute clusterings with the profile-based summarisation approach by Yan et al. [20]. In short, a profile is a submatrix of the data, in which the items are assumed to be independent. A collection of profiles can be overlapping, and summarises a given set of patterns, rather than being a global model for the data. Even though summarisation with profiles is different from our clustering approach, we can compare the quality of the frequency estimates. We mimic the experiments in [20]

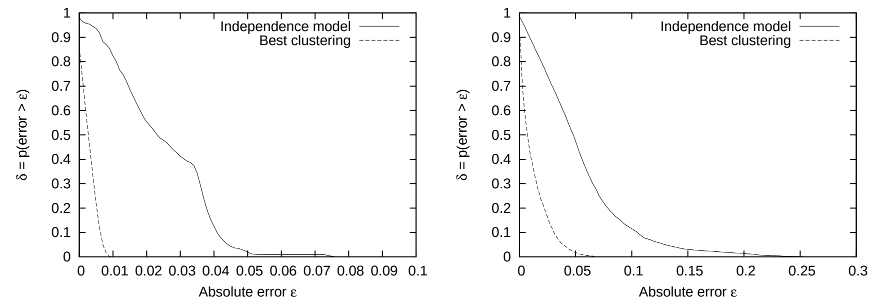


Fig. 4. Probability of an estimation error larger than ϵ for *Connect* (left) and *Mushroom* (right) on the top-10 000 closed frequent itemsets.

on *Mushroom* and *BMS-Webview-1* by comparing the average relative error, also called restoration error. The collection of itemsets contains all frequent closed itemsets for a minsup threshold of 25% and 0.1% respectively. On *Mushroom* we attain a restoration error of 2.31%, which is lower than the results reported in [20] for any number of profiles. For *BMS-Webview-1* our restoration error is 70.4%, which is on par with Yan et al.'s results when using about 100 profiles. Their results improve when increasing the number of profiles; however, the best scores require over a thousand profiles, a number at which it rapidly becomes infeasible to inspect the profile-based summary.

7 Discussion

The experiments show our method discovers high-quality summaries. The high compression ratios for real data, and the inability to compress swap-randomised data, show our models capture the significant structure of the data. The summaries are good surrogates for the data, which can be queried quickly and accurately to approximate the frequencies of itemsets. Inspection of the models showed that correlated attributes are correctly grouped, providing necessary insight when constructing background knowledge to effectively mine the data [9]. Also, this information could be used to select or construct features. Further research into this matter is required, however.

Even while our current implementation is crude, the summaries considered were constructed fast. The implementation can be trivially parallelised, and optimised by using *tid*-lists. We especially regard the development of fast *approximate* summarisation techniques for databases with many transaction and/or items as an important topic for future research, in particular as many data mining techniques cannot consider such datasets directly, but could be made to consider the summary surrogate. Another important open problem is the generation of summaries for data consisting of both numeric and binary attributes.

8 Conclusions

In this paper we introduced a method for getting a good first impression of a binary transaction dataset. Our parameter-free method builds such summaries by grouping items that strongly correlate, and uses the Minimum Description Length principle to identify the best grouping—without requiring a distance measure between items. The result offers an overview of which attributes interact most strongly, and in what value-instantiations these typically occur. Further, as they consider the data symmetrically with regard to 0/1 and form probabilistic models for it, these summaries are good surrogates for the data that can be queried efficiently.

Experiments showed that our method provides high-quality results that correctly identify groups of correlated items, and can be used to obtain close approximations of itemset frequencies.

Acknowledgements

Michael Mampaey is supported by the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

References

1. B. Bringmann and A. Zimmermann. The chosen few: On identifying valuable patterns. In *Proceedings of ICDM'07*, pages 63–72, 2007.
2. T. Calders and B. Goethals. Mining all non-derivable frequent itemsets. In *Proceedings of ECML PKDD'02*, pages 74–85, 2002.
3. V. Chandola and V. Kumar. Summarization – compressing data into an informative representation. In *Proceedings of ICDM'05*, pages 98–105, 2005.
4. T. M. Cover and J. A. Thomas. *Elements of Information Theory, 2nd ed.* John Wiley and Sons, 2006.
5. G. C. Garriga, E. Junttila, and H. Mannila. Banded structure in binary matrices. In *Proceedings of KDD'08*, pages 292–300, 2008.
6. A. Gionis, H. Mannila, T. Mielikäinen, and P. Tsaparas. Assessing data mining results via swap randomization. *TKDD*, 1(3), 2007.
7. P. D. Grünwald. *The Minimum Description Length Principle.* MIT Press, 2007.
8. J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: Current status and future directions. *Data Mining and Knowledge Discovery*, 15(1):55–86, 2007.
9. S. Hanhijärvi, M. Ojala, N. Vuokko, K. Puolamäki, N. Tatti, and H. Mannila. Tell me something I don't know: randomization strategies for iterative data mining. In *Proceedings of KDD'09*, pages 379–388. ACM, 2009.
10. H. Heikinheimo, E. Hinkkanen, H. Mannila, T. Mielikäinen, and J. K. Seppänen. Finding low-entropy sets and trees from binary data. In *Proceedings of KDD'07*, pages 350–359, 2007.
11. H. Heikinheimo, J. Vreeken, A. Siebes, and H. Mannila. Low-entropy set selection. In *Proceedings of SDM'09*, pages 569–579, 2009.
12. A. J. Knobbe and E. K. Y. Ho. Maximally informative k-itemsets and their efficient discovery. In *Proceedings of KDD'06*, pages 237–244, 2006.
13. J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Symposium on Mathematical Statistics and Probability*, 1967.
14. S. Myllykangas, J. Himberg, T. Böhling, B. Nagy, J. Hollmén, and S. Knuutila. DNA copy number amplification profiling of human neoplasms. *Oncogene*, 25(55):7324–7332, 2006.
15. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proceedings of ICDT'99*, pages 398–416, 1999.
16. R. Pensa, C. Robardet, and J-F. Boulicaut. A bi-clustering framework for categorical data. In *Proceedings of ECML PKDD'05*, pages 643–650, 2005.
17. A. Siebes, J. Vreeken, and M. van Leeuwen. Item sets that compress. In *Proceedings of SDM'06*, pages 393–404, 2006.
18. J. Vreeken, M. van Leeuwen, and A. Siebes. Preserving privacy through data generation. In *Proceedings of ICDM'07*, pages 685–690, 2007.
19. J. Wang and G. Karypis. SUMMARY: Efficiently summarizing transactions for clustering. In *Proceedings of ICDM'04*, pages 241–248, 2004.
20. X. Yan, H. Cheng, J. Han, and D. Xin. Summarizing itemset patterns: A profile-based approach. In *Proceedings of KDD'05*, pages 314–323, 2005.