Maximally Informative *k*-Itemsets and their Efficient Discovery

Arno J. Knobbe[1,2], Eric K.Y. Ho[1]

[1]Kiminkii, P.O. box 171, NL-3990 DD, Houten, The Netherlands

[2]Utrecht University, P.O. box 80 089, NL-3508 TB Utrecht, The Netherlands

+31 30 253 99 67

a.knobbe@kiminkii.com, e.ho@kiminkii.com

ABSTRACT

In this paper we present a new approach to mining binary data. We treat each binary feature (item) as a means of distinguishing two sets of examples. Our interest is in selecting from the total set of items an itemset of specified size, such that the database is partitioned with as uniform a distribution over the parts as possible. To achieve this goal, we propose the use of *joint entropy* as a quality measure for itemsets, and refer to optimal itemsets of cardinality *k* as *maximally informative k-itemsets*. We claim that this approach maximises distinctive power, as well as minimises redundancy within the feature set. A number of algorithms is presented for computing optimal itemsets efficiently.

Categories and Subject Descriptors

F.2 Analysis of Algorithms and Problem Complexity. G.3 Probability and Statistics. H.1.1 Systems and Information Theory. I.2.6 Learning

General Terms

Algorithms, Theory.

Keywords

Maximally informative *k*-itemsets, binary data, feature selection, information theory, joint entropy, subgroup discovery.

1. INTRODUCTION

In this paper we present a novel class of regularities to be mined from binary datasets. If we think of the binary features (items) in such datasets as means of distinguishing examples within the dataset from each other, we could wonder which features are best at doing so. Furthermore, we could wonder whether some features provide any meaningful and additional distinction, given a number of the other features. In other words, we are interested in selecting a small set of binary features that provides as good a distinction of the examples as possible. A subset of binary features that maximises the joint entropy of its features provides exactly that notion. We will refer to such feature sets of cardinality *k* as *maximally informative k-itemsets* (or *miki*'s).

KDD'06, August 20–23, 2006, Philadelphia, Pennsylvania, USA.

As information theory dictates [6], the joint entropy of a set of *k* binary features is optimal when the distribution of possible combinations of binary values is uniform. In other words, the database is partitioned into 2^k parts of equal size. This happens if all selected features are independent and each feature has a probability of $p(x = 1) = 0.5$. A nice property of favouring sets with high joint entropy is that features are only selected if they provide distinctive power that is additional to the remaining features. A feature may be distinctive in isolation (i.e. have an entropy of 1 bit), but is of no use if this information is already conveyed by the remaining features.

Although the definition of miki's is fairly basic, they are of use in a wide array of domains. The general applicability becomes clearer if we interpret binary features in the most general sense: any function $D \rightarrow \{0, 1\}$ that splits a database in two mutually exclusive parts. This includes the obvious binary attributes (items), but also subgroups or patterns (examples are either covered by a pattern, or not). Furthermore, one can think of binary classifiers as binary features. The following potential applications explain our motivation for working on miki's:

- Feature selection [7][10][15][22] in binary databases. A considerable number of Data Mining techniques is hindered by the presence of many attributes, either due to the presence of irrelevant attributes, or for reasons of efficiency. If we can select a small number of attributes, while retaining most of the discriminative power, the results of such techniques may be improved.
- Subgroup discovery [11][13][23]. Subgroup discovery (whether supervised or unsupervised) typically produces an abundance of interesting patterns that is very hard to inspect manually. There may be considerable redundancy because relevant subsets of the data may be described in several (nearly) equivalent ways. Furthermore, some patterns constitute logical combinations of two or more alternative patterns reported by the subgroup discovery algorithm. Selecting patterns by means of miki's will condense the original set of individually interesting patterns to a manageable set of important findings.
- Propositionalisation in Multi-Relational Data Mining (MRDM) [13][16][18]. An important task within MRDM is the translation of multi-relational data to a propositional format. The obvious application of this operation is to allow the deployment of attribute-value

learning systems that could otherwise not be used due to the non-determinate nature of the data. The transformation is typically run in an unsupervised setting, and many binary features (each corresponding to a discovered multi-relational pattern) are produced. Without proper filtering, the resulting binary table will contain too many redundant features.

Maximally informative *k*-itemsets have a number of features that set them apart from frequent itemsets, which have been the dominant class of models for mining binary data [1][8][24]. First of all, miki's are symmetric in their treatment of the values 0 and 1. The two values are merely seen as identifiers for the two distinct groups. In fact, swapping these values in any of the features considered has no effect on the computation. In many applications, this symmetry is desirable. A symmetrical treatment of 0 and 1 also implies that it makes sense to not just have a minimum support threshold of the 1's, as is the case in most frequent itemset algorithms, but also on the 0's (in a sense a maximum support threshold for the 1's). The use of a minimum on the entropy replaces these two thresholds, as the entropy of a feature decreases if either of the values becomes dominant.

The goals of frequent itemsets and miki's are orthogonal in a sense. Whereas frequent itemsets favour items that are positively associated [1], miki's aim at optimising the independence of items within the miki. Frequent itemset mining is sensitive to items with high support, because their addition to any frequent itemset has little effect on the support. Such items are ignored when mining for miki's, because they are frequent and thus provide little information, but also because internal correlation is minimised.

In this paper, we present some basic results concerning miki's and information theory (Section 2). More importantly, we introduce a number of algorithms for computing miki's (Section 3). We start by describing four algorithms for computing exact miki's. As the number of candidate miki's can be very large, efficiency is a major concern. We start by considering an exhaustive algorithm, and improve on that by applying a number of the observations from Section 2. As the scale of the search space becomes prohibitive with larger values of *k*, we also describe a greedy algorithm that computes approximate miki's efficiently, at a small penalty for optimality.

In Section 4, we perform a series of experiments to demonstrate the value of maximally informative *k*-itemsets, and to assess the efficiency of the algorithms presented. We also empirically test the error introduced by the greedy algorithm. The paper finishes with related work and conclusions in Section 5 and 6.

2. PRELIMINARIES

In this section, we will provide definitions for basic concepts related to maximally informative *k*-itemsets, and introduce a number of useful properties. These properties will be exploited by the algorithms introduced in the next section. In the remainder, we will assume that we have a collection of items I. We will use lowercase x_1, x_2,… to indicate items, and capitals X and Y to indicate itemsets. We assume that we can obtain a probability estimate $p(x = a)$ by scanning the database associated with I, and counting the relative number of occurrences of value a.

We start with a number of basic definitions and properties, based on information theoretical notions presented in [6]. We begin by defining the notion of joint entropy of an itemset, a measure for the amount of information conveyed by the itemset. A maximally informative *k*-itemset (or *miki* in short) is then simply the itemset of specified cardinality that maximises this joint entropy. Joint entropy is measured in *bits*.

Definition 1 (joint entropy). Suppose that $X = \{x_1,\ldots,x_k\}$ is an itemset, and $B = (b_1,\ldots,b_k) \in \{0,1\}^k$ is a tuple of binary values. The *joint entropy* of X is defined as

$$H(X) = -\sum_{B\in\{0,1\}^k} p(x_1 = b_1,\ldots,x_k = b_k)\lg p(x_1 = b_1,\ldots,x_k = b_k)$$

Definition 2 (maximally informative *k*-itemset). Suppose that I is a collection of n items. An itemset $X \subseteq I$ of cardinality k is a *maximally informative k-itemset*, *iff* for all itemsets $Y \subseteq I$ of cardinality k,

$$H(Y) \le H(X)$$

Note that the joint entropy of an itemset increases as more items are added to it. Because items are binary features, every item provides at most 1 bit of additional information.

Proposition 1 (monotonicity of joint entropy). Suppose X and Y are two itemsets such that $X \subseteq Y$. Then

$$H(X) \le H(Y)$$

Proposition 2 (unit growth of joint entropy). Suppose X and Y are two itemsets such that $X \subseteq Y$. Then

$$H(Y) \le H(X) + |Y \setminus X|$$

As Proposition 1 shows, the joint entropy is a non-decreasing function of the number of items involved in the itemset. This raises the issue of choosing a good value for the parameter k. In theory larger values of k will give a better distinction between examples. On the other hand, feature *selection* calls for small numbers of items. In many cases, the right value of k will be implied by the application. The problem is very similar to the selection of the right number of clusters in clustering tasks, where this number is often determined by considering increasing values, and stopping when there is a clear drop in improvement. An example of this is given in our experiments in Section 4.

Algorithms for computing miki's of a desired cardinality k will have to consider a large amount of candidate itemsets and compute their joint entropy by scanning the database. Computing the joint entropy of a given itemset essentially comes down to a bucket sort [5] where each bucket corresponds to a cell in a contingency table of k dimensions. This table scan can be performed in $O(kN)$, where N equals the number of records in the database. In theory a total of $\binom{n}{k}$ itemsets will have to be considered. As N is typically large and therefore a table scan is an expensive operation, it is important to have upper bounds on the value of $H(X)$ that are relatively cheap to compute. Such a bound can then be used to discard candidates that are clearly not maximal, without having to scan the data for verification. A simple, moderately tight, upper bound can be obtained by considering the entropy of the individual items in the itemset.

Proposition 3 (independence bound on joint entropy). Suppose that $X = \{x_1,\ldots,x_k\}$ is an itemset. Then

$$H(X) \le \sum_i H(x_i)$$

Although the proof (see [6]) of this proposition is non-trivial, and involves among others Jensen's inequality, the intuition behind it is straightforward. Every item provides a certain amount of information to the joint entropy. If all pairs of items are independent, the joint entropy equals the sum of entropies. However, if items are dependent, they share a certain amount of information, which is ignored when simply adding the individual entropies.

Note that, although our specific interest in this paper is with binary features, Definitions 1, 2 and Proposition 3 can be easily generalised to categorical features [6].

Example 1. Consider the following database consisting of four items. Items A to C all have equal numbers of 1's and 0's, hence $H(A) = 1$, $H(B) = 1$, $H(C) = 1$. $H(D) = -\frac{3}{8}\lg\frac{3}{8} - \frac{5}{8}\lg\frac{5}{8} \approx 0.96$. The itemset $\{A, B, C\}$ is a maximally informative k-itemset of cardinality 3. Its joint entropy equals 2.5 bits which is less than the 3 bits of information provided by the three items separately (Proposition 3).

A	B	C	D
1	1	1	0
1	1	0	0
1	1	1	0
1	0	0	0
0	1	1	0
0	0	0	1
0	0	1	1
0	0	0	1

The fact that mutual information between items is ignored by Proposition 3 suggests that we can obtain a potentially tighter bound by grouping items that share a considerable amount of information, and taking the joint entropy within this group rather than the sum of the entropies. This amounts to treating each such group as a single categorical feature with all binary combinations as possible values. Joint entropies for such (small) subsets of items can for example be obtained cheaply by pre-computing and storing the results in a datastructure for future reference. We continue by presenting a number of results that will be exploited in algorithms presented in the next section. We show that grouping items within the itemset can be used to compute upper bounds for $H(X)$ that are potentially tighter than the independence bound (Proposition 3). Furthermore, bounds become tighter as more items are grouped together.

Definition 3 (partition of itemset). Suppose that $X = \{x_1,\ldots,x_k\}$ is an itemset. A *partition* of X is a set of itemsets $P = \{B_1,\ldots,B_m\}$ such that

$$\forall i, j : i \neq j \Rightarrow B_i \cap B_j = \emptyset,$$

$$\bigcup_{i=1}^{m} B_i = X, \forall i : B_i \neq \emptyset, m \geq 2$$

The itemsets B_i are known as the *blocks* of P.

Definition 4 (joint entropy of partition). Suppose that $P = \{B_1,\ldots,B_m\}$ is a partition of an itemset. The *joint entropy* of P is defined as

$$H(P) = \sum_i H(B_i)$$

Proposition 4 (partitioned bound on joint entropy). Suppose that $P = \{B_1,\ldots,B_m\}$ is a partition of an itemset X. Then

$$H(X) \leq H(P)$$

This proposition shows that $H(P)$ is an upper bound on the joint entropy. If we think of each block B_i in P as a categorical feature with at most $2^{|B_i|}$ values, and apply the categorical version of Proposition 3 (see [6]), we can easily proof that

$$H(X) = H(B_1,\ldots,B_m) \leq \sum_i H(B_i) = H(P)$$

Proposition 5 (independence bound on partitioned joint entropy). Suppose that $P = \{B_1,\ldots,B_m\}$ is a partition of an itemset $X = \{x_1,\ldots,x_k\}$. Then

$$H(P) \leq \sum_i H(x_i)$$

This proposition shows that $H(P)$ is at least as tight as the independence bound (Proposition 3). This follows from

$$H(P) = \sum_i H(B_i) \leq \sum_i \sum_j H(B_{ij}) = \sum_i H(x_i)$$

where B_{ij} refers to the jth item of the ith block B_i.

Propositions 4 and 5 demonstrate that partitions of the itemset at hand can provide a tighter upper bound. We could thus consider all possible partitions and select the lowest value, hoping to avoid unnecessary table scans. Unfortunately, the number of partitions of a k-itemset can become very large, even with reasonably small values of k. This number is known as the *Bell number*, $B(k)$, and satisfies the following recurrence relation [17]:

$$B(k) = \sum_{i=0}^{k-1} \binom{k-1}{i} B(i)$$

where $B(0) = 1$.

Fortunately, the following results show that we need only consider partitions of 2 blocks. This still leaves us with $2^{k-1}-1$ partitions to examine, which is likely to be too expensive. Furthermore, computation of the partitioned joint entropy requires the joint entropy of itemsets of cardinality up to $k-1$. In this paper we will therefore only consider partitions with blocks of up to 2 items, even though this produces sub-optimal bounds.

Definition 5 (inclusion of partitions). Suppose that P and P' are partitions of an itemset. P *includes* P' ($P' \leq P$) *iff*

$$\forall X \in P \exists Q \subseteq P' : X = \bigcup_i Q_i$$

Proposition 6 (anti-monotonicity of partitioned bound). Suppose that P and P' are partitions of an itemset, and $P' \leq P$. Then

$$H(P) \leq H(P')$$

Proposition 7 (2-partitions). Suppose that X is an itemset. The tightest partitioned bound on $H(X)$ can be found among the partitions of X of cardinality 2.

Example 2. The itemset $\{B, C, D\}$ in Example 1 produces 4 partitions:

$$\{\{B\}, \{C\}, \{D\}\}, \{\{B, C\}, \{D\}\},$$

$$\{\{B, D\}, \{C\}\}, \{\{C, D\}, \{B\}\}.$$

The lowest partitioned joint entropy for these partitions is produced by $\{\{B, D\}, \{C\}\}$: 2.41 bits. The joint entropy of $\{B, C, D\}$ equals 2.16 bits. Note that similar items (B and D) are grouped together. Assuming a search algorithm has already considered $\{A, B, C\}$, and is therefore looking for itemsets exceeding 2.5 bits, itemset $\{B, C, D\}$ would then be discarded (Proposition 4). The upper bound on the basis of Proposition 3 (2.95 bits) would not be sufficient to do so.

3. ALGORITHMS

In this section we present a number of algorithms for computing maximally informative k-itemsets. The basic outline of these algorithms is to consider all subsets of size k in lexicographic order, and compute the joint entropy of each, in order to find the maximum. We then proceed to apply a number of the results from Section 2 in order to discard itemsets that can be proven non-maximal, or even prune large portions of the search space.

We will write a k-element subset of I as a list of integers that refer to the elements of I.

$$X = [x_1, \ldots, x_k]$$

where

$$x_1 < \ldots < x_k$$

The algorithms will rely on a simple algorithm for computing the lexicographic successor of a given itemset, presented in [16]. This algorithm (*LexicographicSuccessor*) works as follows (see pseudocode). The first while-loop identifies the last item i in X that can be increased (i.e. replaced by a succeeding item). If no such item can be found, all the subsets have been exhausted. Otherwise the item i is increased, and all elements to the right of i are reset to refer to successive items.

Algorithms 1 to 4 all report a single miki (the last one found), although they can be easily modified to report all miki's, because every algorithm tests all candidate miki's. The fifth algorithm reports a single approximation that may or may not be an actual miki.

Algorithm **LexicographicSuccessor(*X, k, n*)**

$Y \leftarrow X$
$i \leftarrow k$
while $i \geq 1$ **and** $x_i = n - k + i$
    $i \leftarrow i - 1$
if ($i = 0$)
    return "undefined"
else
    for $j \leftarrow i$ **to** k
        $y_j \leftarrow x_i + 1 + j - i$
    return Y

Algorithm 1. The first algorithm (**ExhaustiveMiki**) now simply considers all k-itemsets exhaustively, and reports the maximally informative one (the last one if more than one miki exists). All itemsets are considered by calling **LexicographicSuccessor** repeatedly until a value *undefined* is returned (see pseudocode). The joint entropy is computed by projecting the binary table on the selected items, and counting the different combinations that occur (Definition 1).

Algorithm **ExhaustiveMiki(*k, n*)**

$X \leftarrow [1, \ldots, k]$
$h_{\max} \leftarrow$ JointEntropy(X)
$Y \leftarrow X$
while LexicographicSuccessor(X, n) ≠ "undefined"
    $X \leftarrow$ LexicographicSuccessor(X, n)
    $h \leftarrow$ JointEntropy(X)
    if $h \geq h_{\max}$
        $h_{\max} \leftarrow h$
        $Y \leftarrow X$
return Y

Algorithm 2. A first improvement on the exhaustive algorithm can be obtained by applying Proposition 3. This proposition provides a cheap way of computing an upper bound on the itemset at hand that can potentially be used to discard a large part of the candidates. A single condition needs to be added to the basic algorithm that checks whether the upper bound of the current itemset exceeds the current maximum. If this condition is satisfied, we still need to compute the joint entropy by performing an expensive table scan. Computation of the independence bound (Proposition 3) requires the computation and storage of the entropies of the n individual items, as a preparatory step.

Algorithm 3. Especially when correlated items abound, the independence bound cannot be expected to be very tight. Proposition 4 provides a tighter bound, and thus a potentially faster algorithm. This algorithm computes an upper bound by choosing a partition of the itemset at hand, and computing the joint entropy of this partition. Although it is tempting to consider all possible partitions of an itemset, and pick the lowest value, the discussion in Section 2 shows that this becomes too expensive with large values of k. Instead, we only consider partitions of blocks of at most 2 items. Initial experimentation shows that picking such a partition at random provides the fastest algorithm. Even though this method may not consider the tightest upper bound, it is faster because only a single bound needs to be computed. Partitioned bounds can be computed by looking up, and adding, pre-computed joint entropies of individual items and pairs of items.

Algorithm 4. Assuming Proposition 4 provides a substantial reduction in table scans, we can expect the running time of Algorithm 3 to be dominated by the computation of the upper

bound for each of the $\binom{n}{k}$ itemset. In Algorithm 4, we aim to improve on this by skipping a range of candidates on the basis of the joint entropy of the sub-itemset they have in common. By applying Proposition 2, we know that if an itemset X of size k-l cannot be extended with any l items (at most l bits) to exceed to the current maximum, we can skip all k-itemsets starting with X. This procedure introduces a new parameter l. Larger values of l lead to larger portions of the search space being skipped. However, lower values increase the odds of producing a tight enough upper bound to allow a skip. Informal experimentation shows that $l = 3$ typically provides the best trade-off, and is thus used in our experiments in Section 4.

Algorithm 5. So far, we have considered algorithms that provide exact solutions. We will see in Section 4 that such exact algorithms become impractical with increasing values of k. We therefore present a final algorithm (**ForwardSelection**) that produces approximate miki's. The major advantage of this algorithm is that it considers only a tiny fraction of all potential k-itemsets. The reported itemset is computed by progressively adding items to the initial empty set until k items are selected. At each step the new item to be added is chosen such that the increase in joint entropy is maximised (see pseudocode). This constitutes a greedy step. The algorithm has an asymptotic complexity of $O(k^2nN)$.

Algorithm **ForwardSelection(*k, n*)**

$X \leftarrow \varnothing$
for $i \in \{1,\ldots,k\}$
    $h_{\max} = 0$
    for $j \in \{1,\ldots,n\}$
        $h \leftarrow \text{JointEntropy}(X \cup \{j\})$
        if $j \notin X \wedge h > h_{\max}$
            $h_{\max} \leftarrow h$
            $m \leftarrow j$
    $X \leftarrow X \cup \{m\}$
return X

---

4. EXPERIMENTS

We start our experiments informally, by demonstrating the usefulness of miki's in the context of subgroup discovery, which is our main motivation for this work. Figure 1 (left) shows two numeric features (*lumo* en *logp*) associated with 188 molecules appearing in the *Mutagenesis* database [21]. Molecules appear in two classes, mutagenic (grey dots) and non-mutagenic (black dots). The axis-parallel lines represent the decision boundaries formed by the collective of 82 subgroups (rules) discovered by the mining package Safarii [13][20]. The package produces conjunctive rules, and as is clear from the figure, there is considerable redundancy in the individual conditions, as well as among the conjunctions. The 2-dimensional space is divided into far fewer areas than can be expected from the number of rules discovered.

By interpreting each rule as an item, we can use our approach to reduce redundancy in the set of rules. The figure on the right demonstrates a selection of 4 rules that form a miki. Clearly, this subset of rules captures most of the partitioning produced by the 82 rules. Redundant and overly specific decision boundaries (individual conditions) are avoided. The only location where important distinction between examples is discarded is in the lower left corner. The two important decision boundaries in this area can be added by increasing the number of selected subgroups to 6. In our further experiments we will see that the joint entropy reaches an optimum of 2.706 at $k = 6$ and remains constant after addition of more items (see last table).

In the remainder, we analyse how well the different algorithms presented scale with increasing values of k. Three datasets of varying sizes were used: the well-known *Mushroom* and *Chess* datasets, as well as *LumoLogp*, a dataset derived from the 82 rules discovered in the previously mentioned *Mutagenesis* (datasets can be obtained from the authors). We have run all five algorithms on these datasets with values of k between 2 and 7. Typical applications of miki's tend to fall well within this range. For each run, we state the number of table scans, the total time (*m*:*ss*) and the joint entropy of the result. Runs lasting more than 1000 minutes were terminated. In cases where none of the exact algorithms ran under 1000 minutes, a single run was executed with low priority, in order to obtain an exact value for the joint entropy.

Clearly, algorithm 1 performs impractically slow on all datasets, and can only serve as a baseline for the remaining results. The application of Proposition 3 (algorithm 2) and especially Proposition 4 (algorithm 3) provides a significant improvement on the first two datasets. A reduction of number of table scans of more than a factor 10^6 is common. In these cases, considering all candidates rather than scanning the data becomes the governing factor in the running time. Although algorithm 4 exploits this fact, its advantages turn out to be only marginal, because relatively few additional candidates can be skipped. For all these algorithms k = 7 seems to be the upper limit.

Unfortunately, the first four algorithms perform poorly on *LumoLogp*. The high level of redundancy means that most items can be potential elements of a miki, and few candidates can be discarded directly. Although the process that generated this dataset would typically be made more selective, some degree of redundancy is normal in applications of miki's, which makes these algorithms problematic.

The greedy fifth algorithm takes less than one second on each dataset for each chosen value of k, and is hence a fast alternative. The joint entropy of its (potentially suboptimal) solution is always within a few percent of the optimal value. In this case the redundancy in the data is advantageous, because making suboptimal greedy choices is less likely.

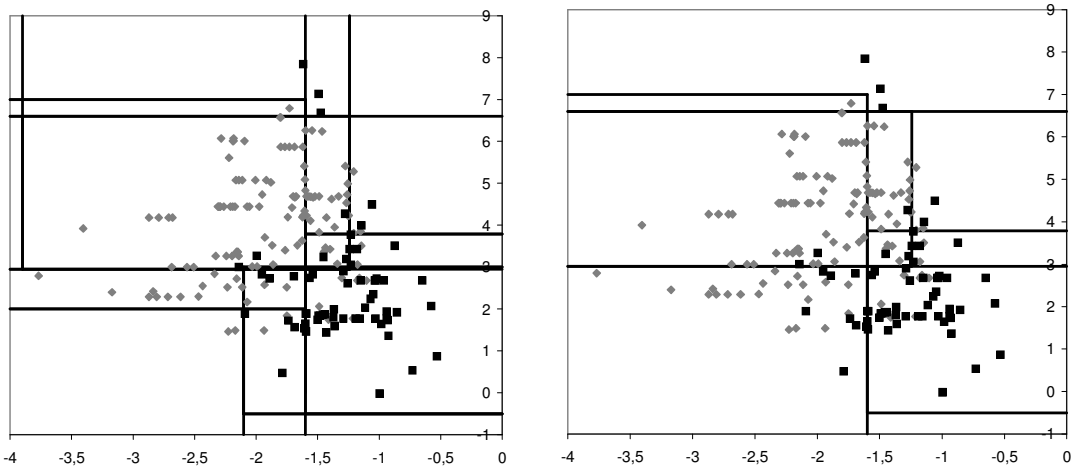

Figure 1 partitioning of a 2-dimensional space based on the full rule set of 82 rules, and the filtered rule set of 4 rules.

Mushroom (119 x 8124)

	$k = 2$			$k = 3$			$k = 4$		
algorithm	ts	time	max	ts	time	max	ts	time	max
1	7021	0:18	1.995	273,819	16:29	2.975	$7.94 \cdot 10^6$	735:23	3.934
2	12	0		265	0:03		4,917	1:34	
3	4	0:36		83	0:38		602	1:36	
4							602	1:37	
5	237	0	1.994	354	0	2.973	470	0:01	3.934

	$k = 5$			$k = 6$			$k = 7$		
algorithm	ts	time	max	ts	time	max	ts	time	max
1	$1.82 \cdot 10^8$	>1000	4.886	$3.47 \cdot 10^9$	>1000	5.635	$5.6 \cdot 10^{10}$	>1000	6.397
2	69,134	34:21		$1.23 \cdot 10^6$	692:42		$1.95 \cdot 10^7$	>1000	
3	9,747	23:25		211,934	445:58		$4.58 \cdot 10^6$	>1000	
4	9,747	16:17		209,329	244:11		$4.4 \cdot 10^6$	>1000	
5	585	0:03	4.886	354	0	5.619	470	0:01	6.313

Chess (75 x 3196)

	$k = 2$			$k = 3$			$k = 4$		
algorithm	ts	time	max	ts	time	max	ts	time	max
1	2275	0:02	1.982	67,525	2:02	2.963	$1.21 \cdot 10^6$	64:52	3.918
2	18	0		113	0		2,366	0:14	
3	14	0:07		46	0:07		334	0:13	
4							334	0:14	
5	149	0	1.982	222	0	2.96	294	0	3.918

	$k = 5$			$k = 6$			$k = 7$		
algorithm	ts	time	max	ts	time	max	ts	time	max
1	$1.72 \cdot 10^7$	>1000	4.852	$2.01 \cdot 10^8$	>1000	5.755	$1.98 \cdot 10^9$	>1000	6.593
2	36,178	4:13		406,396	59:42		$3.89 \cdot 10^6$	675:34	
3	3,372	1:41		33,007	18:37		411,668	203:46	
4	3,372	1:33		32,995	15:42		390,673	213:57	
5	365	0	4.852	435	0:01	5.755	504	0:01	6.593

LumoLogp (82 x 188)

	$k = 2$			$k = 3$			$k = 4$		
algorithm	ts	time	max	ts	time	max	ts	time	max
1	3,321	0	1.942	88,560	0:10	2.313	$1.74 \cdot 10^6$	5:50	2.584
2	2,787	0		88,560	0:10		$1.74 \cdot 10^6$	5:35	
3	249	0		62,778	0:08		$1.20 \cdot 10^6$	3:56	
4								4:03	
5	163	0	1.938	243	0	2.251	322	0	2.429

	$k = 5$			$k = 6$			$k = 7$		
algorithm	ts	time	max	ts	time	max	ts	time	max
1	$2.73 \cdot 10^7$	>1000	2.695	$3.50 \cdot 10^8$	>1000	2.706	$3.8 \cdot 10^9$	>1000	2.706
2	$2.73 \cdot 10^7$	128:47		$3.50 \cdot 10^8$	>1000		$3.8 \cdot 10^9$	>1000	
3	$2.73 \cdot 10^7$	129:14		$3.50 \cdot 10^8$	>1000		$3.8 \cdot 10^9$	>1000	
4	$2.73 \cdot 10^7$	121:28		$3.50 \cdot 10^8$	>1000		$3.8 \cdot 10^9$	>1000	
5	400	0	2.578	477	0	2.706	504	0:01	2.706

5. RELATED WORK

We have already mentioned the relation between miki's and frequent itemsets. Another obvious role for miki's is in feature selection. Many algorithms have been developed for this purpose (see [7] for an overview), particularly in a supervised setting: select only those features that are relevant for predicting the value of a class variable. Our framework on the other hand is unsupervised. The aim is simply to select features that allow the optimal distinction between examples, regardless of any specific classification or regression task.

A well-known example of a supervised feature selection algorithm is Relief [10]. It works by assigning a weight to features on the basis of their ability to distinguish between class values. The weights are updated according to an instance based learning approach, and only features with sufficient weight are returned as relevant features. An important limitation of this approach is that features are selected on relevancy only. No attempt is made to prevent redundancy within the selected features. This is in fact the inverse from our approach which returns a non-redundant feature set, but relevancy does not apply, due to its unsupervised nature.

Alternative approaches [15][19][22] do address redundancy, and are thus feature subset selection methods rather than feature selection methods. Most of these approaches work in a supervised setting. They come in two varieties: *filter* methods and *wrapper* methods [7]. Wrapper methods employ the performance of a specific learning algorithm (such as C4.5 or an instance based approach) to select features. A straight-forward example of this is given in [14] and [19], where the performance of a *simple decision table* is used to judge the quality of a particular feature subset. Filter methods on the other hand select feature subsets on the basis of quality measures that are relatively independent of the learning algorithm to be applied subsequently. These measures typically come from Information Theory [15][22].

Most authors recognise the exponential nature of the search space of feature subsets, and present heuristic search algorithms similar to our fifth algorithm. Typical examples of search strategies are Forward Selection, Backward Selection, and variations thereof [15][19][22], or more randomised methods. Unfortunately, few papers compare such greedy solutions to an exhaustive analysis, making a good judgement of the error with respect to the optimal solution difficult.

The framework presented is related to our previous work on discovering primary keys and functional dependencies [12]. In a sense, miki's can be seen as noisy counterparts of (candidate) keys, as they aim to optimise the distinction between examples. In fact, a miki of joint entropy equal to $log(N)$ forms a candidate key. Similarly, the feature subsets produced by a different method, called FOCUS [2], correspond to functional dependencies between the feature subset and the class variable.

Somewhat related to our work is the notion of Independent Component Analysis (ICA) [4][9]. It is a statistical method that expresses the original multidimensional, and typically numeric, data into a small number of variables that are more or less statistically independent. These variables are typically latent, in the sense that they do not appear as actual attributes in the original data. Clearly, our approach only returns attributes that appear in the data.

6. CONCLUSION

We have presented a new framework for mining binary data, based on information theoretical notions. Items are selected on the basis of their distinctive power, also relative to other selected items, such that redundant items will be ignored. As such, it provides an interesting alternative to the common frequent itemset framework. The framework has a number of interesting applications, notably the reduction of results produced by other pattern discovery techniques. Especially in rich domains such as structured or multi-relational data, where the expressiveness of pattern languages used cause high levels of redundancy, miki's allow the discovery of important patterns rather than simply interesting ones. This application was demonstrated by our first experiment, and will be further investigated in future work.

We have presented a number of algorithms of varying efficiency. Based on some basic information theoretical observations, it is often possible to prune large parts of the search space, and thus

find optimal solutions. Unfortunately, the exact algorithms break down in some data sets with high levels of redundancy, where few candidate itemsets can be discarded without going back to the data. Finally we have presented an approximate algorithm that is extremely fast, even with larger values of *k*, while still producing results comparable to the optimal solution.

Our application of choice for miki's is as a means of filtering results obtained by subgroup or rule discovery. In this context miki's capture the intuitive requirement that results be non-redundant. However, one can envisage alternative intuitions that would inspire different filtering methods. As an example, one could require selected patterns to be mutually exclusive, or optimal with respect to further classification (wrapper approach). In fact any quality measure for itemsets can be applied. In future work we intend to compare such measures.

7. REFERENCES

[1] Agrawal, R., Imielinski, T., Swami, A., *Mining Association Rules between Sets of Items in Large Databases*, In Proceedings ACM SIGMOD, 1993

[2] Almuallim, H., Dietterich, T.G., *Learning with Many Irrelevant Features*, In Proceedings of AAAI '91, 1991

[3] Bingham, E., Mannila, H. Seppänen, J.K., *Topics in 0-1 Data*, in Proceedings SIGKDD '02, 2002

[4] Comon, P., *Independent Component Analysis – a New Concept?* Signal Processing, 36:287-314, 1994

[5] Cormen, T.H., Leiserson, C.E., Rivest, R.L., *Introduction to Algorithms*, The MIT Press, 1990

[6] Cover, T.M., Thomas, J.A., *Elements of Information Theory*, John Wiley & Sons, 1991

[7] Guyon, I., Elisseeff, A., *An Introduction to Variable and Feature Selection*, Journal of Machine Learning Research 3, 1157-1182, 2003

[8] Han, J., Pei, J., Yin, Y., *Mining Frequent Patterns without Candidate Generation*, In Proceedings ACM SIGMOD, 2000

[9] Hyvärinen, A., Karhunen, J., Oja, E., *Independent Component Analysis*, John Wiley & Sons, 2001

[10] Kira, K., Rendell, L.A., *A Practical Approach to Feature Selection*, In Proceedings ML '92, 1992

[11] Klösgen, W., *Explora: A Multipattern and Multistrategy Discovery Assistant*, In Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, Cambridge, USA, 1996

[12] Knobbe, A.J., Adriaans, P.W., *Discovering Foreign Key Relations in Relational Databases*, In Proceedings of EMCSR '96, 1996

[13] Knobbe, A.J., *Multi-Relational Data Mining*, Ph.D. dissertation, 2004, http://www.kiminkii.com/thesis.pdf

[14] Kohavi, R., *The Power of Decision Tables*, In Proceedings of ECML '95, 1995

[15] Koller, D., Sahami, M., *Toward Optimal Feature Selection*, In Proceedings of ICML '96, 1996

[16] Kramer, S., Lavraĉ, N., Flach, P.A., *Propositionalization Approaches to Relational Data Mining*, in Relational Data Mining, Springer-Verlag, 2001

[17] Kreher, D.L., Stinson, D.R., *Combinatorial Algorithms*, CRC Press, 1999

[18] Lavraĉ, N., Flach, P.A., *An Extended Transformation Approach to Inductive Logic Programming*, ACM Transactions on Computational Logic, 2(4), 2001

[19] Pfahringer, B., *Compression-Based Feature Subset Selection*, In Proceedings of IJCAI '95, 1995

[20] *Safarii Multi-Relational Data Mining environment*, http://www.kiminkii.com/safarii.html, 2006

[21] Srinivasan, A., Muggleton, S.H., Sternberg, M.J.E., King, R.D., *Theories for mutagenicity: A study in first-order and feature-based induction*, Artificial Intelligence, 85(1,2), 1996

[22] Wang, H., Bell, D., Murtagh, F., *Axiomatic Approach to Feature Subset Selection Based on Relevance*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21(3), 1999

[23] Wrobel, S., *An Algorithm for Multi-Relational Discovery of Subgroups*, In Proceedings PKDD'97, 1997

[24] Zaki, M.J., Orihara, M., *Theoretical Foundations of Association Rules*, In Proceedings ACM SIGMED workshop on research issues in KDD, 1998