Tell Me What I Need to Know: Succinctly Summarizing Data with Itemsets

Michael Mampaey Nikolaj Tatti Jilles Vreeken

Department of Mathematics and Computer Science
Universiteit Antwerpen
{firstname.lastname}@ua.ac.be

ABSTRACT

Data analysis is an inherently iterative process. That is, what we know about the data greatly determines our expectations, and hence, what result we would find the most interesting. With this in mind, we introduce a well-founded approach for succinctly summarizing data with a collection of itemsets; using a probabilistic maximum entropy model, we iteratively find the most interesting itemset, and in turn update our model of the data accordingly. As we only include itemsets that are surprising with regard to the current model, the summary is guaranteed to be both descriptive and non-redundant. The algorithm that we present can either mine the top-k most interesting itemsets, or use the Bayesian Information Criterion to automatically identify the model containing only the itemsets most important for describing the data. Or, in other words, it will 'tell you what you need to know'. Experiments on synthetic and benchmark data show that the discovered summaries are succinct, and correctly identify the key patterns in the data. The models they form attain high likelihoods, and inspection shows that they summarize the data well with increasingly specific, yet non-redundant itemsets.

Categories and Subject Descriptors

H.2.8 [**Database management**]: Database applications-*Data mining*

General Terms

Theory, Algorithms, Experimentation

1. INTRODUCTION

Knowledge discovery from data is an inherently iterative process. That is, what we already know about the data greatly determines our expectations, and therefore, which results we would find interesting and/or surprising. Early on in the process of analyzing a database, for instance, we are happy to learn about the generalities underlying the data, while later on we will be more interested in the specifics that build upon these concepts. Essentially, this process comes down to summarization: we want to know what is interesting in the data, and we want this to be reported succinctly and without redundancy.

KDD'11, August 21–24, 2011, San Diego, California, USA.

As a simple example, consider supermarket basket analysis. Say, we just learned that *pasta* and *tomatoes* are very often sold together, and that we already know that many people buy *wine*. Then it is not very interesting to find out that the combination of these three items is also sold frequently. Even if we cannot predict this frequency exactly, we can say that this pattern is redundant. At the same time, at this stage of the analysis we are probably also not interested in highly detailed patterns, e.g., an itemset representing the many ingredients of an elaborate Italian dinner. While its frequency may be surprising, it is also very specific, and may well be better explained by some more general patterns. Still, this itemset might be regarded as highly interesting further on in the discovery process, after we have learned those more general patterns, and if this is the case, we would like it to be reported at that time. Consequently, this is the approach we adopt in this paper; we incrementally adjust our model as we discover new patterns, to obtain a non-redundant summary.

As natural as it may seem to update a knowledge model during the discovery process, few pattern mining techniques actually follow such a dynamic approach of discovering patterns that are surprising with regard to what we have learned so far. That is, while many techniques provide a series of patterns in order of interestingness, most score these patterns using a static model; during this process the model, and hence the itemset scores, are not updated with the knowledge gained from previously discovered patterns. For instance, Tan et al. study 21 of the most well-known interestingness measures, all of which are static, and most of which are based on the independence model [24]. The static approach gives rise to the typical problem of traditional pattern mining: overwhelmingly large and highly redundant collections of patterns.

Our objective is to find a succinct summary of a binary dataset, that is, to obtain a small, yet high-quality set of itemsets that describes key characteristics of the data at hand, in order to gain useful insights. This is motivated by the fact that many existing algorithms often return too large collections of patterns with considerable redundancy, as discussed above. The view that we take in this paper on succinctness and non-redundancy is therefore a fairly strict one.

To model the data, we use the powerful and versatile class of maximum entropy models. We construct a maximum entropy distribution that allows us to directly calculate the expected frequencies of itemsets. Then, at each iteration, we return the itemset that provides the most information, i.e., for which our frequency estimate was most off. We update our model with this new knowledge, and continue the process. The non-redundant model that contains the most important information is thus automatically identified. Therefore, we paraphrase our method as 'tell me what I need to know'.

While solving the maximum entropy model is infeasible in general, we show that in our setting it can be solved efficiently, depending on the amount of overlap between the selected patterns.

Similarly, we give an efficient method for estimating frequencies from the model. Further, we provide an efficient convex heuristic for pruning the search space for the most informative itemsets. This approach allows us to mine our collection of itemsets on the fly, instead of picking them from a larger candidate set which would have to be mined and stored beforehand. Our approach is parameter-free: no maximal error threshold needs to be provided, nor a minimum support or a significance level. The best model can automatically be determined through the Bayesian Information Criterion (BIC); alternatively, we can also mine the top-k most interesting itemsets. Finally, the user can easily infuse background knowledge into the model (in the form of itemset frequencies), to avoid redundancy with regard to what the user already knows.

2. RELATED WORK

Selecting or ranking interesting patterns is a well-studied field in data mining. Existing techniques can roughly be split in two groups.

The first group consists of techniques that measure how *surprising* the support of an itemset is compared against some null hypothesis: the more the observed frequency deviates from the expected value, the more interesting it is. The simplest null hypothesis is the independence model [1, 2]. More flexible models have been suggested, for example, Bayesian Networks [14]. The major caveat of these approaches is that the null hypothesis is static and hence we keep rediscovering the same information. As a result, this will lead to pattern collections with high levels of redundancy. The alternative approach is to select itemsets using a dynamic hypothesis. That is, when a new itemset is discovered, the model is updated such that we take the already discovered information into account.

The use of maximum entropy models in pattern mining has been proposed by several authors [15, 26, 28, 30]. Discovering an itemset collection with a good BIC score was suggested by Tatti and Heikinheimo [28]. Alternatively, Tatti [27] samples collections and bases the significance of an itemset on its occurrence in the discovered collections. However, in order to guarantee that the score can be computed, the authors restrict themselves to downward closed and decomposable collections. The method of Tatti [26] uses local models, that is, to compute the support of an itemset X, we are only allowed to use sub-itemsets of X, and it outputs a p-value. A threshold is needed to determine whether X is important. Related, Webb [31] defines itemsets as *self-sufficient*, if their support differs significantly from what can be inferred from their sub- and supersets; therefore such a model is also local. Wang and Parthasarathy [30] incrementally build a maximum entropy model by adding itemsets that deviate more than a given error threshold. The approach ranks and adds itemsets in level-wise batches. This may still, however, lead to redundancy within a batch of itemsets. The method introduced by Kontonasios and De Bie [15] uses row and column margins to construct a maximum entropy model of the data, from which noisy tiles [9] are then discovered, using an MDL-based information score. An important difference, however, is that the data is treated as a single sample from the space of datasets, whereas we consider distributions of transactions.

An alternative approach, called swap randomization, has been suggested by Gionis et al. [10] and Hanhijärvi et al. [13]. The former approach ranks itemsets using a static hypothesis, the latter uses a dynamic hypothesis. Here as well, the authors treat the whole data as a sample from the space of datasets, having the same certain set of statistics, including row and column margins. The authors use an MCMC approach to sample datasets and compute empirical p-values. In both approaches, the framework needs a threshold for deciding which itemsets are significant.

The MINI algorithm by Gallo et al. [7] similarly uses row and column margins to rank itemsets. It first orders all potentially interesting itemsets by computing their p-value according to these margins. Then, as subsequent itemsets are added, the p-values are recomputed, and the itemsets are re-ordered according to their new p-values. This method, however, does not allow querying.

The method of Yan et al. [32] summarizes a collection of itemsets by clustering them, and then representing each cluster as a *profile*. The approach is different from ours, in that it summarizes a given set of patterns, rather than the data itself.

KRIMP, by Siebes et al. [23], employs the MDL principle to select those itemsets that together compress the data best. As such, patterns that essentially describe the same part of the data are rejected. The models it finds are not probabilistic, and therefore cannot easily be used to calculate probabilities. Further, while non-redundant from a compression point of view, many of the patterns it selects are variations of the same theme. Other differences to our method are that KRIMP considers its candidates in a static order, and that it is not trivial to make it consider background knowledge.

It should be noted that in contrast to the algorithm that we propose here, most of the above methods require the user to set one or several parameters, such as a maximum error threshold or a significance level. Many also cannot easily be used to estimate the frequency of an itemset. Further, all of them are two-phase algorithms, i.e., they require that the user provides a collection of candidate (frequent) itemsets to the algorithm, which must be completely mined and stored first, before running the actual algorithm.

3. PRELIMINARIES AND NOTATION

This section provides some preliminaries and the notation that we will use throughout the paper.

By a *transaction* we mean a binary vector of size N generated by some unknown distribution. The ith element in a random transaction corresponds to an *attribute* or an *item* a_i, a Bernoulli random variable. We denote the set of all items by $A = \{a_1, \ldots, a_N\}$. We denote the set of all possible transactions by $\mathcal{T} = \{0, 1\}^N$.

The input of our method is a *binary dataset* D, which is simply a sample of $|D|$ (not necessarily distinct) transactions. Given the data D we define an empirical distribution

$$q_D(a_1 = v_1, \ldots, a_N = v_N) = |\{t \in D \mid t = v\}|/|D| \ .$$

An *itemset* X is a subset of A. For notational convenience, given a distribution p, an itemset $X = \{x_1, \ldots, x_L\}$, and a binary vector v of length L, we often use $p(X = v)$ to denote $p(x_1 = v_1, \ldots, x_L = v_L)$. If v consists entirely of 1's, then we use the notation $p(X = 1)$. Given the data D, the *frequency* of an itemset X is defined $fr(X) = q_D(X = 1)$. An *indicator function* $S_X : \mathcal{T} \to \{0, 1\}$ of an itemset X maps a transaction t to a binary value such that $S_X(t) = 1$ if and only if t contains X.

The *entropy* of a distribution p over $\mathcal{T}$ is defined as

$$H(p) = -\sum_{t \in \mathcal{T}} p(A = t) \log p(A = t) \ ,$$

where the base of the logarithm is 2, and by convention $0 \log 0 = 0$.

4. IDENTIFYING THE BEST SUMMARY

Our goal is to discover the set of itemsets $\mathcal{C}$ that provides the most important information about the data, while containing as little redundancy as possible. Here, we regard information as whether we are able to reliably predict the data using these itemsets and their frequencies. By non-redundancy, we mean that any subset of $\mathcal{C}$ provides a significantly different description of the data. This

is equivalent to requiring that the frequency of an itemset $X \in \mathcal{C}$ should be surprising with respect to $\mathcal{C} \setminus X$. In other words, we do not want $\mathcal{C}$ as a collection to be unnecessarily complex, or capture spurious information. We want it to contain only those itemsets that we really need.

Informally, assume that we have a score $s(\mathcal{C})$ which measures the quality of an itemset collection $\mathcal{C}$. Then our aim is to find that $\mathcal{C}$ with the best score $s(\mathcal{C})$. Analogously, if we only want to know k itemsets, we look for the set $\mathcal{C}$ of size at most k, with the best $s(\mathcal{C})$.

Next, we will detail how we define our models, how we define this score, provide theoretical evidence why it is a good choice, and discuss how to compute it efficiently.

EXAMPLE 1. *As a running example, assume we have a transaction dataset D with eight items, a to h. Furthermore, consider the set of itemsets $\mathcal{C} = \{abc, cd, def\}$ with frequencies 0.5, 0.4 and 0.8, respectively. Assume for the moment that based on $\mathcal{C}$, our method predicts that the frequency of the itemset agh is 0.19. Now, if we observe in the data that $fr(agh) = 0.18$, then we can safely say that agh is redundant because it does not contribute a lot. On the other hand, if $fr(agh) = 0.7$, then the frequency of agh is surprising, and therefore $\mathcal{C} \cup \{agh\}$ would give an improved description of D.*

4.1 Maximum Entropy Model

In our approach we make use of maximum entropy models. This is a class of probabilistic models that are identified by the Maximum Entropy principle [4] as those models that make optimal use of the provided information. That is, they rely only on this information and are fully unbiased otherwise. This property makes these models very suited for identifying good patterns: by using maximum entropy models to measure the quality of a set of patterns, we know that our measurement only relies on the provided frequencies of the patterns, and that it will not be thrown off due to some spurious structure in the data. These models have a number of theoretically appealing properties, which we will discuss after a formal introduction.

Assume that we are given a set of itemsets $\mathcal{C} = \{X_1, \ldots, X_k\}$. Each itemset X_i has a frequency $fr(X_i)$ in the data. We are interested in distributions that satisfy these frequencies, that is, let us consider the following set of distributions

$$\mathcal{P} = \{p \mid p(X_i = 1) = fr(X_i)\,, i = 1, \ldots, k\} \ .$$

Among these distributions we are interested in only one, namely the unique distribution that maximizes the entropy,

$$p^*_{\mathcal{C}} = \arg\max_p \{H(p) \mid p \in \mathcal{P}\} \ .$$

For notational convenience we will omit $\mathcal{C}$ from p^* whenever it is clear from the context. Although we restrict ourselves to itemset frequencies here, many other patterns or count statistics that can be expressed as linear combinations of transactions could be used, e.g., transaction lengths [29], association rule confidence, etc. Computing the maximum entropy model, however, is far from trivial, and is treated in Section 4.5.

4.2 Model Scoring

First we discuss how we will measure the quality of a model. A natural first choice would be to directly measure the goodness of fit, using the log-likelihood of the maximum entropy model, that is, $\log p^*(D) = \sum_{t \in D} \log p^*(A = t)$. However, this choice suffers from overfitting: larger collections of itemsets will always provide more information, hence allow for better estimates, and therefore have a better log-likelihood. Consequently, we need to prevent our method from overfitting. Therefore, we use the well-founded Bayesian Information Criterion (BIC), which favors models that fit the data well with few parameters. It has a strong theoretical support in Bayesian model selection [22] as well as through the Minimum Description Length principle [12]. The BIC score of a collection $\mathcal{C}$ is defined as

$$s(\mathcal{C}) = -\log p^*_{\mathcal{C}}(D) + 1/2\ |\mathcal{C}| \log |D| \ .$$

The smaller this score, the better the model. The first term is simply the negative log-likelihood of the model, while the second term is a penalty on the number of parameters—the number of itemsets in our case. Consequently, the best model is identified as the model that provides a good balance between high likelihood and low complexity. Moreover, we automatically avoid redundancy, since models with redundant itemsets are penalized for being too complex, without sufficiently improving the likelihood.

4.3 Properties of the Model

In this subsection we discuss some properties of the quality measure and the maximum entropy model.

We begin by stating a famous theorem that the maximum entropy model has an exponential form.

THEOREM 2 (THEOREM 3.1 IN [4]). *Given a collection of itemsets $\mathcal{C} = \{X_i\}_{i=1}^k$ with frequencies $fr(X_i)$, let us define $\mathcal{P} = \{p \mid p(X_i = 1) = fr(X_i)\}$. If there is a distribution in $\mathcal{P}$ that has only non-zero entries, then the maximum entropy distribution p^* can be written as*

$$p^*(A = t) = u_0 \prod_{X \in \mathcal{C}} u_X^{S_X(t)} \ ,$$

where $u_X \in \mathbb{R}$, and u_0 is a normalization factor.

This form will help us to discover the model and compute the likelihood term in the scoring function.

COROLLARY 3 (OF THEOREM 2). *The log-likelihood of the maximum entropy distribution p^* for a set of itemsets $\mathcal{C}$ is equal to*

$$\log p^*(D) = |D|(\log u_0 + \sum_{X \in \mathcal{C}} fr(X) \log u_X) = -|D| H(p^*) \ .$$

Thus, to calculate the BIC score $s(\mathcal{C})$, it suffices to compute the parameters u_X and u_0 of the distribution p^*.

4.4 Reducing Redundancy

Here we show that our score favors itemset collections with low redundancy, and make a theoretical link with some existing redundancy reduction techniques for pattern mining.

A baseline technique for ranking itemsets is to compare the observed frequency against the expected value of some null hypothesis. The next theorem shows that if the observed frequency of an itemset X agrees with the expected value $p^*(X = 1)$, then X is redundant.

THEOREM 4. *Let $\mathcal{C}$ be a collection of itemsets and let p^* be the corresponding maximum entropy model. Let $X \notin \mathcal{C}$ be an itemset such that $fr(X) = p^*(X = 1)$. Then $s(\mathcal{C} \cup \{X\}) > s(\mathcal{C})$.*

PROOF. We will prove the theorem by showing that the likelihood terms for both collections are equal. Define the collection $\mathcal{C}_1 = \mathcal{C} \cup \{X\}$ and let $\mathcal{P}_1$ be the corresponding set of distributions. Let p^*_1 be the distribution maximizing the entropy in $\mathcal{P}_1$. Note that since $\mathcal{C} \subset \mathcal{C}_1$, we have $\mathcal{P}_1 \subseteq \mathcal{P}$ and hence $H(p^*_1) \leq H(p^*)$. On the other hand, the assumption in the theorem implies that $p^* \in \mathcal{P}_1$ and so $H(p^*) \leq H(p^*_1)$. Thus, $H(p^*) = H(p^*_1)$ and since the distribution maximizing the entropy is unique, we have $p^* = p^*_1$. This shows that the likelihood terms in $s(\mathcal{C})$ and $s(\mathcal{C}_1)$ are equal. The BIC penalty term is larger in $s(\mathcal{C}_1)$ which concludes the proof. □

Algorithm 1: ITERATIVESCALING($\mathcal{C}$)

input : itemset collection $\mathcal{C} = \{X_1, \ldots, X_k\}$, frequencies $fr(X_1), \ldots, fr(X_k)$
output : parameters u_X and u_0 of the maximum entropy distribution $p^*_{\mathcal{C}}$ satisfying $p^*_{\mathcal{C}}(X_i) = fr(X_i)$ for all i

1 initialize p;
2 **while** p has not converged **do**
3 **for each** X in $\mathcal{C}$ **do**
4 compute $p(X = 1)$;
5 $u_X \leftarrow u_X \frac{fr(X)}{p(X=1)} \frac{1-p(X=1)}{1-fr(X)}$;
6 $u_0 \leftarrow u_0 \frac{1-fr(X)}{1-p(X=1)}$;
7 **return** p;

Theorem 4 states that adding an itemset X to $\mathcal{C}$ improves the score only if its observed frequency deviates from the expected value. The amount of deviation required is determined by the penalty term. This gives us a convenient advantage over methods that are based solely on deviation, since they require a user-specified threshold.

Two corollaries stated below follow directly from Theorem 4. The first relates our approach to *closed* itemsets [21]. An itemset is closed if all of its supersets have a strictly lower support. An itemset is a *generator* if all of its subsets have a strictly higher support. The second corollary provides a similar relation with *non-derivable* itemsets [3]. An itemset is called derivable if its support can be inferred exactly from the supports of all of its proper subsets.

COROLLARY 5 (OF THEOREM 4). *Let $\mathcal{C}$ be a collection of itemsets. Assume that $X, Y \in \mathcal{C}$ such that $X \subset Y$ and $fr(X) = fr(Y) \neq 0$. Assume that $Z \notin \mathcal{C}$ such that $X \subset Z \subset Y$. Then $s(\mathcal{C} \cup \{Z\}) > s(\mathcal{C})$.*

COROLLARY 6 (OF THEOREM 4). *Let $\mathcal{C}$ be a collection of itemsets. Assume that $X \notin \mathcal{C}$ is a derivable itemset and all subitemsets of X are included in $\mathcal{C}$. Then $s(\mathcal{C} \cup \{X\}) > s(\mathcal{C})$.*

Corollaries 5 and 6 connect our approach with popular techniques that losslessly remove redundancy—so-called *condensed representations*. The advantage of our method is that it does not have to be exact. For example, in Corollary 5, $fr(X)$ does not have to equal $fr(Y)$ exactly in order to reject Z from $\mathcal{C}$. This allows us to prune redundancy more aggressively.

4.5 Efficiently Computing the Model

Computing the maximum entropy model comes down to finding the u_0 and u_X parameters from Theorem 2. To achieve this, we use the well-known Iterative Scaling procedure [5], which is given as Algorithm 1. Simply put, it iteratively updates the parameters of the distribution, until it converges to the maximum entropy distribution p^* which satisfies a given set of constraints—itemset frequencies in our case. The distribution is initialized with the uniform distribution, which is done by setting the u_X parameters to 1, and $u_0 = 2^{-N}$ to normalize. Then, for each itemset $X \in \mathcal{C}$, we adjust the corresponding parameter u_X to enforce $p(X = 1) = fr(X)$ (line 5,6). This process is repeated in a round robin fashion until p converges, and it can be shown [5] that p always converges to the maximum entropy distribution p^*. Typically the number of iterations required for convergence is low (usually < 10 in our experiments).

EXAMPLE 7. *In our running example, with $\mathcal{C} = \{abc, cd, def\}$, the maximum entropy model has three parameters u_1, u_2, u_3, and a normalization factor u_0. Initially we set $u_1 = u_2 = u_3 = 1$ and $u_0 = 2^{-N} = 2^{-8}$. Then we iteratively loop over the itemsets and scale the parameters. For instance, for the first itemset abc with frequency 0.5, we first compute its current estimate to be $2^{-3} = 0.125$. Thus, we update the first parameter $u_1 = 1 \cdot (0.5/2^{-3}) \cdot ((1 - 2^{-3})/0.5) = 7$. The normalization factor becomes $u_0 = 2^{-8} \cdot 0.5/(1 - 2^{-3}) \approx 2.2 \cdot 10^{-3}$. Next, we do the same for cd, and so on. After a few iterations, the model parameters converge to $u_1 = 28.5$, $u_2 = 0.12$, $u_3 = 85.4$, and $u_0 = 3 \cdot 10^{-4}$.*

The main bottleneck of this procedure is the inference of an itemset's probability on line 4 of the algorithm,

$$p(X = 1) = \sum_{t \in \mathcal{T}; S_X(t)=1} p(A = t) \ .$$

Since this sum ranges over all possible transactions containing X, it is infeasible to do this in a brute force manner for any non-trivial number of items N. In fact, it has been shown that querying the maximum entropy model is **PP**-hard [25].

Therefore, in order to be able to query the model efficiently, we introduce a partitioning scheme, which makes use of the observation that many transactions have the same probability. Remark that an itemset collection $\mathcal{C}$ partitions $\mathcal{T}$ into blocks of transactions that contain the same set of itemsets. That is, two transactions t_1 and t_2 belong to the same block T if and only if $S_X(t_1) = S_X(t_2)$ for all X in $\mathcal{C}$. Hence, from Theorem 2 we know $p(A = t_1) = p(A = t_2)$. This allows us to define $S_X(T) = S_X(t)$ for any $t \in T$ and $X \in \mathcal{C}$. We denote the partition of $\mathcal{T}$ induced by $\mathcal{C}$ as $\mathcal{T}_{\mathcal{C}}$. Now we can compute the probability of an itemset as

$$p(X = 1) = \sum_{T \in \mathcal{T}_{\mathcal{C}}; S_X(T)=1} p(A \in T) \ .$$

The sum has been reduced to a sum over blocks of transactions, and the inference problem has been moved from the transaction space $\mathcal{T}$ to the block space $\mathcal{T}_{\mathcal{C}}$. In our setting we will see that $|\mathcal{T}_{\mathcal{C}}| \ll |\mathcal{T}|$, which makes inference a lot more feasible. In the worst case, this partition may contain $2^{|\mathcal{C}|}$ blocks, however, through the interplay of the itemsets, it can be as low as $|\mathcal{C}| + 1$. As explained further on, we can exploit, or even choose to limit, the structure of $\mathcal{C}$, such that practical computation is guaranteed.

All we must do now is obtain the block probabilities $p(A \in T)$. Since all transactions t in a block T have the same probability $p(A = t) = u_0 \prod_{X \in \mathcal{C}} u_X^{S_X(t)}$, it suffices to compute the number of transactions in T to get $p(A \in T)$. So, let us define $e(T)$ to be the number of transactions in T, then

$$p(A \in T) = \sum_{t \in T} p(A = t) = e(T)\, u_0 \prod_{X \in \mathcal{C}} u_X^{S_X(T)} \ .$$

Algorithm 2 describes COMPUTEBLOCKSIZES. In order to compute the block sizes $e(T)$, we introduce a partial order on $\mathcal{T}_{\mathcal{C}}$. Let

$$sets(T; \mathcal{C}) = \{X \in \mathcal{C} \mid S_X(T) = 1\}$$

be the itemsets in $\mathcal{C}$ that occur in the transactions of T. Note that every block corresponds to a unique subset of $\mathcal{C}$, but conversely not every subset of $\mathcal{C}$ corresponds to a (nonempty) transaction block. We can now define the partial order on $\mathcal{T}_{\mathcal{C}}$ as follows,

$$T_1 \subseteq T_2 \quad \text{if and only if} \quad sets(T_1; \mathcal{C}) \subseteq sets(T_2; \mathcal{C}) \ .$$

In order to compute the size $e(T)$ of a block, we first compute its *cumulative* size,

$$c(T) = \sum_{T' \supseteq T} e\big(T'\big) \ ,$$

Algorithm 2: COMPUTEBLOCKSIZES($\mathcal{C}$)

input : itemset collection $\mathcal{C} = \{X_1, \ldots, X_k\}$
output : block sizes $e(T)$ for each T in $\mathcal{T}_\mathcal{C}$

for T in $\mathcal{T}_\mathcal{C}$ **do**
  $I \leftarrow \bigcup\{X \mid X \in sets(T;\mathcal{C})\}$;
  $c(T) \leftarrow 2^{N-|I|}$;
sort the blocks in $\mathcal{T}_\mathcal{C}$;
for T_i in $\mathcal{T}_\mathcal{C}$ **do**
  $e(T_i) \leftarrow c(T_i)$;
  for T_j in $\mathcal{T}_\mathcal{C}$, with $j < i$ **do**
    if $T_i \subset T_j$ **then**
      $e(T_i) \leftarrow e(T_i) - e(T_j)$;
return $\mathcal{T}_\mathcal{C}$;

which is the number of transactions that contain *at least* all the itemsets in $sets(T;\mathcal{C})$. Let $I = \bigcup\{X \mid X \in sets(T;\mathcal{C})\}$. That is, I are the items that occur in all transactions of T. Then it holds that $c(T) = 2^{N-|I|}$, where N is the total number of items. Finally, to extract the block sizes $e(T)$ from the cumulative sizes $c(T)$, we use the inclusion-exclusion principle. To that end, we topologically sort the blocks such that if $T_2 \subset T_1$, then T_1 occurs before T_2 (which can easily be ensured when constructing $\mathcal{T}_\mathcal{C}$). Then we simply iterate over the blocks in reverse, and subtract the sizes of their super-blocks,

$$e(T) = c(T) - \sum_{T' \supsetneq T} e\big(T'\big) \ .$$

EXAMPLE 8. *Assume again that we have a dataset with eight items (a to h), and an itemset collection containing three itemsets $\mathcal{C} = \{abc, cd, def\}$ with frequencies 0.5, 0.4 and 0.8.*

Table 1 shows the sizes of the transaction blocks. Note that while there are 256 transactions in $\mathcal{T}$, there are only 7 blocks in $\mathcal{T}_\mathcal{C}$, whose sizes and probabilities are to be computed. (The eighth combination, abc and def but not cd, is clearly impossible.)

Let us compute the sizes of the first three blocks. For the first block, $I = abcdef$ and therefore $c(T) = 4$, for the second block $I = abcd$, and for the third block $I = abc$. Since the first block is the maximum with respect to the order $\subseteq$, its cumulative size is simply its size, so $e(T) = 4$. For the second block, we subtract the first block, and obtain $e(T) = 16-4 = 12$. From the third block we subtract the first two blocks, and we have $e(T) = 32-12-4 = 16$. Now, to compute, say, $p(abc = 1)$, we simply need the sizes of the blocks containing abc, and the current model parameters,

$$p(abc = 1) = 4(u_0u_1u_2u_3) + 12(u_0u_1u_2) + 16(u_0u_1).$$

Table 1: Transaction blocks for the running example above, with $X_1 = abc$, $X_2 = cd$, and $X_3 = def$.

X_1	X_2	X_3	$c(T)$	$e(T)$	$p(A = t)$
1	1	1	4	4	$u_0u_1u_2u_3$
1	1	0	16	12	$u_0u_1u_2$
1	0	0	32	16	u_0u_1
0	1	1	16	12	$u_0u_2u_3$
0	1	0	64	36	u_0u_2
0	0	1	32	16	u_0u_3
0	0	0	256	160	u_0

Lastly, the algorithm can be significantly optimized as follows. Assume that we can divide $\mathcal{C}$ into two disjoint groups $\mathcal{C}_1$ and $\mathcal{C}_2$, such that if $X_1 \in \mathcal{C}_1$ and $X_2 \in \mathcal{C}_2$, then $X_1 \cap X_2 = \emptyset$. Let $B = \bigcup \mathcal{C}_1$ be the set of items occurring in $\mathcal{C}_1$. Theorem 2 implies that $p^*(A) = p^*(B)p^*(A \setminus B)$. In other words, the maximum entropy distribution can be factorized into two *independent* distributions, namely $p^*(B)$ and $p^*(A \setminus B)$, more importantly, the factor $p^*(B)$ depends only on $\mathcal{C}_1$. Consequently, if we wish to compute the probability $p^*(X = 1)$ such that $X \in B$, we can ignore all variables outside B and all itemsets outside $\mathcal{C}_1$. The number of computations to perform by COMPUTEBLOCKSIZES can now be greatly reduced, since in the case of independence $|\mathcal{T}_\mathcal{C}| = |\mathcal{T}_{\mathcal{C}_1}| \times |\mathcal{T}_{\mathcal{C}_2}|$, and we can simply compute the block sizes for $\mathcal{T}_{\mathcal{C}_1}$ and $\mathcal{T}_{\mathcal{C}_2}$ separately. Naturally, this decomposition can also be applied when there are more than two of such disjoint groups of itemsets.

Further, in order to *guarantee* that we can apply the above separation, we could reduce the solution space slightly by imposing a limit on the number of items or itemsets per group, such that the number of blocks remains small. Alternatively, we could first partition the items of the dataset into smaller, approximately independent groups [16], and then apply the algorithm for each group separately.

4.6 Including Frequencies of Individual Items

Often it is useful to inspect the frequencies of the individual items (i.e., the column margins) in a dataset, since they supply basic, yet intuitive and easily calculable information about the data. For instance, they say which combinations of items are more (or less) likely to occur together frequently. However, in our method we cannot add the set of all singleton itemsets $\mathcal{I}$ to a collection $\mathcal{C}$, since the number of transaction blocks would become $|\mathcal{T}_{\mathcal{C}\cup\mathcal{I}}| = |\mathcal{T}| = 2^N$, by which we would be back at square one. We sketch (due to space restrictions) how this can be solved easily. Let $\mathcal{C}' = \mathcal{C} \cup \mathcal{I}$. We continue working with $\mathcal{T}_\mathcal{C}$ rather than $\mathcal{T}_{\mathcal{C}'}$. As before, the maximum entropy model has an exponential form: $p^*_{\mathcal{C}'}(A = t) = u_0 \prod_{X\in\mathcal{C}} u_X^{S_X(t)} \prod_{i\in\mathcal{I}} v_i^{S_i(t)}$. The second product defines an independence distribution $v = v_0 \prod_i v_i^{S_i(t)}$. Then $p^*_{\mathcal{C}'}(A \in T) = v(A \in T)\frac{u_0}{v_0} \prod_{X\in\mathcal{C}} u_X^{S_X(T)}$. Thus, we simply need to compute $v(A \in T)$, which is computed very similar to $e(A \in T)$ with COMPUTEBLOCKSIZES—note that $e(A = t)$ is basically the uniform distribution multiplied with 2^N. Hence, we can include the item frequencies at an only marginal additional cost.

4.7 Querying the Model

We have seen how we can efficiently query the probability of an itemset $X \in \mathcal{C}$ when given the maximum entropy distribution p^*. In order to compute the probability of an arbitrary itemset Y that is not a member of $\mathcal{C}$, we do the following. We first set $\mathcal{G} = \mathcal{C} \cup \{Y\}$ and compute the block probabilities $e(T)$ or $v(A \in T)$ for T in $\mathcal{T}_\mathcal{G}$ by calling COMPUTEBLOCKSIZES. Then, we can simply use the parameters of p^* to compute $p^*(Y = 1)$,

$$p^*(Y = 1) = \sum_{\substack{T\in\mathcal{T}_\mathcal{G} \\ S_Y(T)=1}} v(A \in T)\frac{u_0}{v_0} \prod_{X\in\mathcal{C}} u_X^{S_X(T)} \ .$$

Thus, to obtain the probability of an itemset, it suffices to compute the block probabilities in $\mathcal{T}_\mathcal{G}$, for which we know that $|\mathcal{T}_\mathcal{G}| \leq 2|\mathcal{T}_\mathcal{C}|$.

4.8 Computational Complexity

Let us analyze the complexity of ITERATIVESCALING. To this end, we define $ps(\mathcal{C}) = |\mathcal{T}_\mathcal{C}|$ as the number of blocks in a partition. Note that $ps(\mathcal{C}) \leq \min\Big(2^{|\mathcal{C}|}, 2^N\Big)$. The computational complexity of COMPUTEBLOCKSIZES is $O(ps(\mathcal{C})^2)$ for a given

Algorithm 3: MTV(D)

input : binary dataset D, background knowledge itemsets $\mathcal{B}$, integer k if mining top-k
output : itemset collection $\mathcal{C}$

$\mathcal{I} \leftarrow$ items in D;
$\mathcal{C} \leftarrow \mathcal{B}$;
while $s(\mathcal{C})$ decreases **and** $|\mathcal{C}| < k$ **do**
  $X \leftarrow$ FINDBESTITEMSET($\emptyset, \mathcal{I}, \emptyset$);
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{X\}$;
  $p^*_{\mathcal{C}} \leftarrow$ ITERATIVESCALING($\mathcal{C}$);
  compute $s(\mathcal{C})$;
return $\mathcal{C}$;

collection $\mathcal{C}$. Assume now that we can partition $\mathcal{C}$ into L disjoint parts $\mathcal{C} = \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_L$, such that if $X \in \mathcal{C}_i$ and $Y \in \mathcal{C}_j$ then $X \cap Y = \emptyset$. As mentioned in Section 4.5, we can now simply compute L independent distributions at a lower total cost. Denoting $B_i = \bigcup_{X \in \mathcal{C}_i} X$, it holds that $ps(\mathcal{C}_i) \leq \min\left(2^{|\mathcal{C}_i|}, 2^{|B_i|}\right)$. If $\mathcal{C}_i$ cannot be partitioned further, this usually means that either $|\mathcal{C}_i|$ is small, or the itemsets in $\mathcal{C}_i$ overlap a lot and $ps(\mathcal{C}_i) \ll 2^{|\mathcal{C}_i|}$. The total execution time of ITERATIVESCALING is $O(K \sum_{i=1}^{L} ps(\mathcal{C}_i)^2)$, where K is the number of iterations, which is usually low. The complexity of estimating the frequency of an itemset requires running COMPUTEBLOCKSIZES once and hence equals $O(\sum_{i=1}^{L} ps(\mathcal{C}_i)^2)$.

5. PROBLEM STATEMENT

In this section we formally state the problem we intend to solve, based on the theory introduced above.

THE PROBLEM. *Given a collection of itemsets $\mathcal{B}$ that represents our background knowledge of a dataset D, a collection of potentially interesting itemsets $\mathcal{F}$, and an integer k, find the subset $\mathcal{C} \subseteq \mathcal{F}$ of size at most k, such that $s(\mathcal{B} \cup \mathcal{C})$ is minimal.*

Note that $\mathcal{F}$ can simply consist of all itemsets, or e.g., be restricted to a collection of frequent itemsets. If we do not wish to constrain the size of $\mathcal{C}$, and essentially disregard k, we can simply set $k = \infty$. Also note that the problem statement does not require $\mathcal{F}$ to be explicitly available beforehand, i.e., it does not have to be mined or materialized in advance (we postpone the details to Section 6.2).

6. MINING SUCCINCT SUMMARIES

In Section 4 we described how to compute the maximum entropy model and its BIC score *given* a set of itemsets. Finding the optimal collection as stated in Section 5, however, is clearly infeasible. The size of the search space is $\sum_{j=0}^{k} \binom{|\mathcal{F}|}{j} \leq 2^{|\mathcal{F}|}$. If we do not restrict the candidate itemsets, then the number of all non-singleton itemsets is $|\mathcal{F}| = 2^N - N - 1$. Moreover, the score function is not monotonic, which prevents us from straightforwardly exploring the search space.

Therefore, we resort to using a heuristic, greedy approach. Starting with a set of itemsets representing our background knowledge—for instance the singletons—we incrementally construct our summary by iteratively adding the itemset that reduces the BIC score the most. The algorithm stops either when k interesting itemsets are found, or when the score no longer decreases. The pseudo-code for our MTV algorithm, which mines Maximally informaTiVe summaries, is given as Algorithm 3.

6.1 A Heuristic for Scoring Itemsets

Finding the best itemset to add to the current collection is practically infeasible, since it involves solving the maximum entropy model for each and every candidate. This remains infeasible even if we restrict the search space (for example, using only frequent itemsets). Therefore, instead of selecting the candidate that optimizes the BIC score directly, we select the candidate that maximizes a heuristic which expresses the divergence between its frequency and its estimate. To derive and motivate this heuristic we first need the following theorem.

THEOREM 9. *Given an itemset collection $\mathcal{C}$, it holds that*

$$\begin{aligned} \arg\min_X s(\mathcal{C} \cup \{X\}) &= \arg\max_X KL\big(p^*_{\mathcal{C} \cup \{X\}} \,\|\, p^*_{\mathcal{C}}\big) \\ &= \arg\min_X KL\big(q_D \,\|\, p^*_{\mathcal{C} \cup \{X\}}\big) \end{aligned}$$

where KL is the Kullback-Leibler divergence between two distributions, defined as $KL(p \,\|\, q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$, and q_D is the empirical distribution of the data.

PROOF. Let us write $\mathcal{G} = \mathcal{C} \cup \{X\}$. Corollary 3 states that $-\log p^*_{\mathcal{G}}(D) = |D| H(p^*_{\mathcal{G}})$. In addition, we can show with a straightforward calculation that $KL(p^*_{\mathcal{G}} \,\|\, p^*_{\mathcal{C}}) = H(p^*_{\mathcal{C}}) - H(p^*_{\mathcal{G}})$. Therefore, minimizing $s(\mathcal{G})$ is equivalent to maximizing the divergence $KL(p^*_{\mathcal{G}} \,\|\, p^*_{\mathcal{C}})$. The second equality follows similarly. □

Thus, we search for the itemset X such that the new distribution diverges maximally from the previous one, or equivalently, brings us as close to the empirical distribution as possible. The heuristic that we use is an approximation of the KL divergence, where we group the terms containing X in one term, and the terms not containing X into another term. We define $h : [0,1] \times [0,1] \to \mathbb{R}^+$ as

$$h(x, y) = x \log \frac{x}{y} + (1 - x) \log \frac{1 - x}{1 - y} \,.$$

We then pick the itemset maximizing $h(fr(X), p^*(X = 1))$, which we will denote simply as $h(X)$ when fr and p^* are clear from the context. To compute this heuristic, we only need the frequency of X, and its estimate according to the current p^* distribution. This gives us a measure of the divergence between $fr(X)$ and $p^*(X = 1)$, i.e., its surprisingness given the current model.

The following theorem gives and indication of how KL and h relate to one another.

THEOREM 10. *For an itemset collection $\mathcal{C}$ and itemset X, it holds that*

$$0 \leq h(X) \leq KL\big(p^*_{\mathcal{C} \cup \{X\}} \,\|\, p^*_{\mathcal{C}}\big) \,.$$

Moreover, $h(X) = 0$ if and only if $KL\big(p^*_{\mathcal{C} \cup \{X\}} \,\|\, p^*_{\mathcal{C}}\big) = 0$, i.e., when $fr(X) = p^*(X = 1)$.

PROOF. The second inequality follows from the log-sum inequality. The equality to zero is trivially verified. □

6.2 Looking for the Best Itemset

In order to find the itemset maximizing h, we take a depth-first branch-and-bound approach. We exploit the fact that h is convex, and employ the bound introduced by Nijssen et al. [20] to prune large parts of the search space as follows. Say that for a candidate itemset X in the search space, its maximal possible extension in the branch below it is $X \cup Y$ (denoted XY), then for any itemset W such that $X \subseteq W \subseteq XY$, it holds that

$$h(W) \leq \max\left\{h\left(fr(X), p^*(XY)\right), h\left(fr(XY), p^*(X)\right)\right\} \,.$$

If this bound is lower than the best value of h seen so far, we know that no (local) extension W of X can ever become the best itemset with respect to h, and therefore we can safely prune the branch of the search space below X. The algorithm is given in Algorithm 4.

Algorithm 4: FINDBESTITEMSET(X, Y, Z)

input : itemset X, remaining items Y, currently best set Z
output : itemset between X and XY maximizing h, or Z

1 compute $fr(X)$ and $p^*(X)$;
2 **if** $h(X) = h(fr(X), p^*(X)) > h(Z)$ **then**
3 $Z \leftarrow X$;
4 compute $fr(XY)$ and $p^*(XY)$;
5 $b \leftarrow \max\{h(fr(X), p^*(XY)), h(fr(XY), p^*(X))\}$;
6 **if** $b > h(Z)$ **then**
7 **for** $y \in Y$ **do**
8 $Y \leftarrow Y \setminus \{y\}$;
9 $Z \leftarrow$ FINDBESTITEMSET($X \cup \{y\}$, Y, Z);
10 **return** Z;

Table 2: Synthetic and real datasets used in the experiments.

	$\|A\|$	$\|D\|$	$minsup$	$\|\mathcal{F}\|$
Independent	50	100 000	15%	25 110
Markov	20	100 000	5%	8 748
Mosaic	50	100 000	10%	12 256
Abstracts	3 933	859	1%	101 673
Chess (kr-k)	58	28 056	0.1%	19 620
DNA Amplification	391	4 590	0.5%	672 345
Mammals	121	2 183	20%	2 169 624
Mushroom	119	8 124	5%	3 755 512
Paleo	139	124	2%	635 496

An advantage of this approach is that we do not need to collect the frequencies of all candidate itemsets beforehand. Instead, we just compute them on the fly as we need them (line 1). For instance, if we wish to pick itemsets from a collection $\mathcal{F}$ of frequent itemsets for some minimum support threshold, we can integrate the support counting in the depth-first traversal of the algorithm, rather than first mining and storing $\mathcal{F}$ in its entirety. Since mining real datasets with non-trivial minimal support thresholds can easily yield billions of frequent itemsets, this is indubitably a great benefit.

7. EXPERIMENTS

In this section we experimentally evaluate our method and empirically validate the quality of the returned summaries. We implemented a prototype of our algorithm in C++, and provide the source code for research purposes.[1] All experiments were executed on a six-core Intel Xeon machine with 12GB of memory, running Linux.

We evaluate our method on three synthetic datasets, as well as on six real datasets. Their basic characteristics are given in Table 2. The *Independent* data has independent items with random frequencies between 0.2 and 0.8. In the *Markov* dataset each item is a noisy copy of the previous one, with a random copy probability between 0.2 and 0.8. The *Mosaic* dataset is generated by randomly planting five itemsets of size 5 with random frequencies between 0.2 and 0.5, in a database with 1% noise. The *Abstracts* dataset contains the abstracts of all accepted papers at the ICDM conference up to 2007, where words have been stemmed and stop words removed. The *Chess (kr-k)* dataset was obtained from the UCI ML Repository [6], and converted into binary form. The *DNA* data contains information on DNA copy number amplifications. Such copies activate oncogenes and are hallmarks of nearly all advanced tumors [19]. The *Mammals* presence data consists of presence records of European mammals within geographical areas of 50×50 km^2 [17, 18]. The *Mushroom* dataset was obtained from the FIMI dataset repository [11]. Finally, *Paleo* is a dataset of fossil records.

Our method is inherently parameter-free. That is, given enough time, it can select the best set of itemsets from the complete space of possible itemsets. However, although our algorithm is quite efficient, in practice it may not always be feasible to consider *all* itemsets for dense or large datasets. In general, choosing a larger candidate space, yields better models. In our experiments we therefore consider collections of frequent itemsets $\mathcal{F}$ mined at support thresholds as low as feasible. The actual thresholds and corresponding size of $\mathcal{F}$ are depicted in Table 2. Note that the minsup threshold is used to limit the size of $\mathcal{F}$, and is strictly speaking not a parameter of the algorithm itself. For the synthetic datasets, we let the algorithm decide on the best summary size, that is, we set $k = \infty$. As for the real datasets, we do impose a maximum of k itemsets that our algorithm may select. We set k such that the runtime remains within one hour. In all experiments, we initialize $\mathcal{C}$ with the singleton itemsets, i.e., we start from the independence model.

7.1 Model BIC Scores

In Table 3 we give the scores of top-k summaries, the time required to compute them, and for comparison we include the score of the independence model. We see that for most datasets the BIC score (and thus relatedly the negative log-likelihood) decreases a lot, which implies that the summaries we find are of high quality. For very structured datasets, such as *DNA*, this improvement is very large, while it only takes a handful of itemsets to achieve this.

7.2 Summary Evaluation

Here we inspect the discovered data summaries in closer detail.

For the *Independent* dataset we see that the first itemset that the algorithm tries to add immediately increases the BIC score. Therefore, the summary contains only singleton itemsets, which correctly corresponds to the independence model.

With the *Markov* data we notice that the itemsets in the summary are all quite small, and consist of consecutive items. This is in line with expectations, since the items form a Markov chain. The algorithm finds that together with the individual items, a summary of 21 itemsets suffices.

The summary of the *Mosaic* data contains the itemsets that were used to construct the dataset. Figure 1 depicts the evolution of the BIC score. We see that as the five embedded itemsets are discovered, the BIC score is drastically reduced. Afterwards, some additional itemsets are discovered, which contain items from two or more of the generating itemsets. However, they are not noise, and are needed to

Table 3: The BIC scores of the discovered models, compared to the independence model. (Lower scores are better.)

	k	time	$s(\mathcal{C})$	$s(indep)$
Independent	0	10 s	4 494 656	4 494 656
Markov	21	4 834 s	1 826 576	2 000 143
Mosaic	19	686 s	812 792	2 168 276
Abstracts	24	2 774 s	254 974	256 937
Chess (kr-k)	15	3 104 s	774 899	787 078
DNA Amplification	132	3 303 s	91 125	185 499
Mammals	13	1 826 s	107 268	120 132
Mushroom	15	1 817 s	343 367	441 903
Paleo	13	2 307 s	7 787	8 822

[1] http://www.adrem.ua.ac.be/implementations

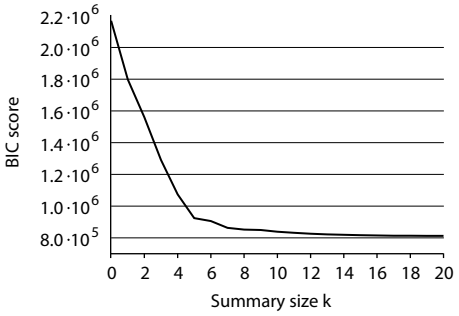

Figure 1: BIC scores for the *Mosaic* dataset for increasing k. The minimum BIC score is attained at $k = 19$.

explain the overlapping behavior of some of the generating itemsets. When k reaches 20, the BIC score increases, and the algorithm stops. (Due to the scale of the plot this is not clearly visible in Figure 1.) Therefore, the algorithm decides that the dataset can be best described with 19 itemsets, of which 5 (the generating itemsets) clearly provide the most information.

The transactions in the *Mammals* data represent geographical areas of 50×50 km^2 in Europe. The itemsets discovered here represent sets of mammals that co-exist in these regions. Investigating the areas where these sets of mammals are present, reveals that many of them form contiguous geographically sound territories, e.g., Scandinavia, the Iberian peninsula, or Eastern Europe. (Pictures are not shown here due to lack of space.)

In the case of the *DNA* data, our algorithm reached more than 100 iterations within one hour. As this dataset is banded, it contains a lot of structure [8]. Our method correctly discovers these bands, i.e., blocks of consecutive items corresponding to related genes, lying on the same chromosomes. The first few dozen sets are large, and describe the general structure of the data. Then, as we continue, we start to encounter smaller itemsets, which describe more detailed nuances in the correlations between the genes. Figure 2 depicts a detail of the *DNA* dataset (note that the figure is transposed), together with a few of the itemsets from the discovered summary.

7.3 Comparison with Other Methods

In Table 4, we give the top-10 itemsets in the *Abstracts* dataset, as discovered by our method (top left), KRIMP [23] (top right), Kontonasios and De Bie's method [15] (bottom left), and Tiling [9] (bottom right). We see that our algorithm discovers recognizable

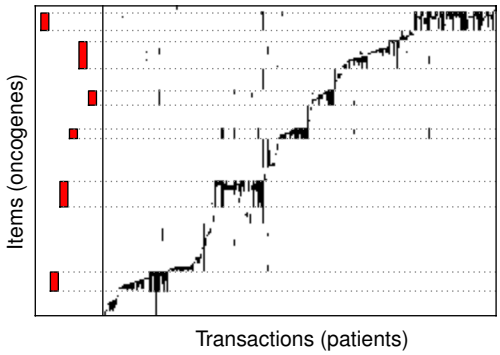

Figure 2: Detail of the *DNA* dataset (right), along with some of the discovered itemsets (left). (Figure is rotated to the left.)

Table 4: Top-10 itemsets of the *Abstracts* dataset for our method (top left), KRIMP [23] (top right), Kontonasios and De Bie [15] (bottom left), and Tiling [9] (bottom right).

machin support svm vector	algorithm experiment result set
associ mine rule	demonstr space
algorithm frequent mine pattern	larg databas
analysi discrimin lda linear	consid problem
algorithm cluster dimension high	knowledg discoveri
nearest neighbor	demonstr experiment
bay naiv	rule mine associ databas
frequent itemset mine	algorithm base approach cluster
analysi compon princip	state art
dimension high real subspac synthet	global local
vector machin support	algorithm mine
discov frequent effici pattern mine algorithm	algorithm base
associ rule database mine algorithm	result set
train learn classifi perform set	approach problem
frequent itemset	method propos
dimensional high cluster mine	result experiment
synthetic real	algorithm perform
seri time	base model
decis tree classifi	set method
experiment propos problem approach result	algorithm gener

data mining topics such as *support vector machines*, *frequent itemset mining* and *principle component analysis*. Further, there is little overlap between the itemsets, and there is no variations-on-the-same-theme type of redundancy present.

For KRIMP we depict the itemsets from the code table which have the highest usage. From a compression point of view, the items in these sets co-occur often, and thus result in small codes for the itemsets. Arguably, this does not necessarily make them the most interesting, however, and we observe that some rather general terms such as *state [of the] art* or *consider problem* are ranked highly. The results of Kontonasios and De Bie's algorithm, based on the Information Ratio of tiles, are different from ours, but seem to be more or less similar in quality for this particular dataset. Finally, for Tiling we provide the top-10 itemsets of size at least two. Without this size restriction, only singletons are returned, which, although having the largest area, are not very informative. Still, the largest discovered tiles are of size two, and contain quite some redundancy, for instance, the top-10 contains only 13 (out of 20) distinct items.

8. DISCUSSION

The approach introduced in this paper fulfills several intuitive expectations one might have about summarization, such as succinctness, providing a characteristic description of the data, and having little redundancy. The experiments show that quantitatively we can achieve good BIC scores with only a handful of itemsets, and that these results are highly qualitative and meaningful; moreover, we can discover them in a relatively short amount of time.

In this paper we consider data mining as an iterative process. By starting off with what we already know, we can identify those patterns that are the most surprising. Simply finding the itemset that is most surprising, is a problem that Hanhijhärvi et al. summarize as 'tell me something I don't know'. When we repeat this process, in the end, we will have identified a group of itemsets that 'tell me all there is to know' about the data. Clearly, this group strongly overfits the data. This is where BIC provides a solution, as it automatically identifies the most informative group. Hence, we paraphrase our approach as 'tell me what I need to know'.

The view that we take here on succinctness and non-redundancy is fairly strict. Arguably, there are settings conceivable where limited redundancy (at the cost of brevity) can give some robustness to a technique, or provide alternative insights by restating facts differently. However, this is not the intention of this paper, and

we furthermore argue that to this end our method can perfectly be complemented by techniques such as redescription mining [33].

There are a number of possible improvements for our method. We are interested in refining the penalty in our score. Our current score only penalizes the *number* of elements in $\mathcal{C}$, not their complexity (e.g., their size). Such a refinement would make it more strict, and could provide better results. Another problem setting in which our method is applicable, is that of finding the best specialization of a given itemset X. That is, to identify the superset Y of X that provides the best score $s(\mathcal{C} \cup \{Y\})$. This setup allows experts to interactively discover interesting itemsets. As part of future work, we are currently investigating this in practice for finding patterns in proteomics and mass-spectrometry data.

9. CONCLUSION

In this paper we introduced a well-founded method for iteratively mining a non-redundant collection of interesting itemsets from transaction data. We employ the Maximum Entropy principle to build a probabilistic model of the data, use this model to iteratively identify the most surprising itemsets, and then update our model accordingly. As such, unlike static interestingness models, our approach does not return patterns that are redundant with regard to what we have learned, and keeps the result set succinct yet informative.

Experiments show that we discover succinct summaries, which correctly identify important patterns in the data. The resulting models attain high log-likelihoods, and are easy to interpret.

Acknowledgements

Michael Mampaey is supported by the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen), Nikolaj Tatti and Jilles Vreeken are supported by Post-Doctoral Fellowships of the Research Foundation—Flanders (FWO).

10. REFERENCES

[1] C. C. Aggarwal and P. S. Yu. A new framework for itemset generation. In *Proc. PODS'98*, pages 18–24, 1998.

[2] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In *Proc. ACM SIGMOD'97*, pages 265–276, 1997.

[3] T. Calders and B. Goethals. Non-derivable itemset mining. *Data Min. Knowl. Disc.*, 14(1):171–206, 2007.

[4] I. Csiszár. I-divergence geometry of probability distributions and minimization problems. *The Annals of Probability*, 3(1):146–158, Feb. 1975.

[5] J. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.

[6] A. Frank and A. Asuncion. UCI machine learning repository. http://archive.ics.uci.edu/ml, 2010.

[7] A. Gallo, N. Cristianini, and T. De Bie. MINI: Mining informative non-redundant itemsets. In *Proc. PKDD'07*, pages 438–445, 2007.

[8] G. C. Garriga, E. Junttila, and H. Mannila. Banded structure in binary matrices. In *Proc. ACM SIGKDD'08*, pages 292–300, 2008.

[9] F. Geerts, B. Goethals, and T. Mielikäinen. Tiling databases. In *Proc. DS'04*, pages 278–289, 2004.

[10] A. Gionis, H. Mannila, T. Mielikäinen, and P. Tsaparas. Assessing data mining results via swap randomization. *Trans. Knowl. Disc. Data*, 1(3), 2007.

[11] B. Goethals and M. Zaki. Frequent itemset mining dataset repository. http://fimi.ua.ac.be/.

[12] P. D. Grünwald. *The Minimum Description Length Principle*. MIT Press, 2007.

[13] 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 *Proc. ACM SIGKDD'09*, pages 379–388, 2009.

[14] S. Jaroszewicz and D. A. Simovici. Interestingness of frequent itemsets using bayesian networks as background knowledge. In *Proc. ACM SIGKDD'04*, pages 178–186, 2004.

[15] K.-N. Kontonasios and T. De Bie. An information-theoretic approach to finding noisy tiles in binary databases. In *Proc. SDM'10*, pages 153–164, 2010.

[16] M. Mampaey and J. Vreeken. Summarising data by clustering items. In *Proc. ECML PKDD'10*, pages 321–336, 2010.

[17] A. Mitchell-Jones, G. Amori, W. Bogdanowicz, B. Krystufek, P. H. Reijnders, F. Spitzenberger, M. Stubbe, J. Thissen, V. Vohralik, and J. Zima. *The Atlas of European Mammals*. Academic Press, 1999.

[18] T. Mitchell-Jones. Societas europaea mammalogica. http://www.european-mammals.org.

[19] 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.

[20] S. Nijssen, T. Guns, and L. De Raedt. Correlated itemset mining in ROC space: a constraint programming approach. In *Proc. ACM SIGKDD'09*, pages 647–656, 2009.

[21] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proc. ICDT'99*, pages 398–416, 1999.

[22] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

[23] A. Siebes, J. Vreeken, and M. van Leeuwen. Item sets that compress. In *Proc. SDM'06*, pages 393–404, 2006.

[24] P. Tan, V. Kumar, and J. Srivastava. Selecting the right interestingness measure for association patterns. In *Proc. ACM SIGKDD'02*, pages 32–41. ACM, 2002.

[25] N. Tatti. Computational complexity of queries based on itemsets. *Inf. Proc. Letters*, pages 183–187, 2006.

[26] N. Tatti. Maximum entropy based significance of itemsets. *Knowl. Inf. Sys.*, 17(1):57–77, 2008.

[27] N. Tatti. Probably the best itemsets. In *Proc. ACM SIGKDD'10*, pages 293–302. ACM, 2010.

[28] N. Tatti and H. Heikinheimo. Decomposable families of itemsets. In *Proc. ECMLPKDD'08*, pages 472–487, 2008.

[29] N. Tatti and M. Mampaey. Using background knowledge to rank itemsets. *Data Min. Knowl. Disc.*, 21(2):293–309, 2010.

[30] C. Wang and S. Parthasarathy. Summarizing itemset patterns using probabilistic models. In *Proc. ACM SIGKDD'06*, pages 730–735, 2006.

[31] G. I. Webb. Self-sufficient itemsets: An approach to screening potentially interesting associations between items. *Trans. Knowl. Disc. Data*, 4(1), 2010.

[32] X. Yan, H. Cheng, J. Han, and D. Xin. Summarizing itemset patterns: a profile-based approach. In *Proc. ACM SIGKDD'05*, pages 314–323, 2005.

[33] M. J. Zaki and N. Ramakrishnan. Reasoning about sets using redescription mining. In *Proc. ACM SIGKDD'05*, pages 364–373, New York, NY, USA, 2005. ACM Press.