Discovering Succinct Pattern Sets Expressing Co-Occurrence and Mutual Exclusivity

Jonas Fischer
Max Planck Institute for Informatics and
Saarland University, Saarbrücken, Germany
fischer@mpi-inf.mpg.de

Jilles Vreeken
CISPA Helmholtz Center for Information Security,
Saarbrücken, Germany
jv@cispa.saarland

ABSTRACT

Pattern mining is one of the core topics of data mining. We consider the problem of mining a succinct set of patterns that together explain the data in terms of mutual exclusivity and co-occurence. That is, we extend the traditional pattern languages beyond conjunctions, enabling us to capture more complex relationships, such as replacable sub-components or antagonists in biological pathways.

We formally define this problem in terms of the Minimum Description Length principle, by which we identify the best set of patterns as the one that most succinctly describes the data. To avoid spurious results—in sparse data mutual exclusivity is likely just due to chance—we propose an efficient statistical test for K-ary mutual exclusivity. As the search space for the optimal model is enormous and unstructured, we propose Mexican, a heuristic algorithm to efficiently discover high quality sets of patterns of co-occurences and mutual exclusivity. Through extensive experiments we show that Mexican recovers the ground truth on synthetic data, and meaningful results on real-world data. Both in stark contrast to the state of the art, that result in millions of spurious patterns.

CCS CONCEPTS

• **Information systems → Data mining**.

KEYWORDS

Pattern Set Mining, Mutual Exclusivity, Pattern Language, Interestingness, MDL

ACM Reference Format:
Jonas Fischer and Jilles Vreeken. 2020. Discovering Succinct Pattern Sets Expressing Co-Occurrence and Mutual Exclusivity. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3394486.3403124

1 INTRODUCTION

Whenever you encounter a polar bear in the wild, it may be comforting to know you do not additionally have to worry about being attacked from behind by a penguin. Polar bears live in the arctic, penguins live in the antarctic, and hence their presence is mutually exclusive. While this common knowledge may be helpful in surviving the arctic, discovering patterns of mutual exclusivity can in general reveal valuable insight that goes well beyond what simple associations or co-occurrences are able to express.

We are particularly interested in discovering a small, non-redundant and easily interpretable set of patterns that together summarize the data and clearly express the significant co-occurrences and mutual exclusivity within. In supermarket basket analysis, patterns of mutual exclusivity allow to express typical buying preferences of customers, such as products of *either* the one *or* the other brand. By combining information of mutual exclusivity with co-occurrences, we can discover the ingredients of a fancy dinner with meat and its vegetarian replacement.

While transaction data is the classic application for pattern mining, the key motivation for this work comes from biology, in particular from single cell sequencing analysis. We consider binary data where for each cell (transactions) we are given which genes are active or which epigenetic features are present (items), and want to gain insight in how these interact. The traditional task is to discover groups of significantly co-activated genes, which are of interest because such genes may be part of genetic pathways or encode part of protein complexes. Patterns of co-occurrence only tell part of the story, however. Genes with *mutually exclusive* activation allow us additionally to discover e.g. *antagonistic* relationships within pathways, such as gene co-activations that are lethal, and *exchangable* sub-components in protein complexes, slightly changing the function of the protein complex—analogous to exchanging the bit of a screwdriver.

Things become even more interesting when we consider a pattern language that additionally allows *combinations* of mutually exclusivity and co-occurrence, as this enables us to discover and succinctly describe a much larger class of possible interactions. For example, we can then express that the co-activation of two genes A and B is mutually exclusive with the co-activation of genes C, D and E, or that we often see either gene A or B activated, but whichever one it is, always together with one out of C, D, and E. Neither of these interactions would be possible to capture with statements on co-occurrence or mutual exclusivity alone, we truly need a pattern language that includes both.

In this paper we define exactly such a pattern language, with the goal to discover the set of patterns over this language that together summarize the data best. We formalize this problem in terms of the Minimum Description Length principle, which permits a score such that it is robust against noise in the occurrences of these patterns, and, can avoid spurious results through an efficient statistical test for K-ary mutual exclusivity. As the combinatorial problem of mining the best set of patterns does not lend itself

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403124

to efficient exact search, we propose MEXICAN, a highly efficient bottom-up heuristic to discover good pattern sets from data.

We evaluate MEXICAN on both synthetic and real-world data. On synthetic data we confirm that, unlike the state of that art, MEXICAN is robust to noise and reconstructs the ground truth, and on a wide range of real world datasets, we find that it discovers small sets of patterns that we confirm to provide meaningful information. For example, from real single cell sequencing data we discover previously unknown patterns of mutual exclusivity that reflect driving factors of local processes, and which can be confirmed with results from the literature.

2 RELATED WORK

Pattern mining is one of the core topics of data mining. The field started with the seminal work by Agrawal and Srikant [1] on mining frequent itemsets. An enormous amount of research effort was focused on mining such patterns efficiently [12, 37] and summarizing the set of all frequent patterns [4, 21, 26]. Frequency turned out to be a bad measure of interestingness, leading to overly many spurious and redundant results. The main idea to alleviate this is to use statistical tests for individual patterns [7, 11, 25, 27, 28, 35] or to use well-founded statistical methods for sets of patterns as a whole [8, 9, 19, 34]. While both approaches have been shown to yield high quality patterns, so far either are restricted to discovering patterns and rules of conjunctions. In addition, our goal is to extract meaningful patterns of mutual exclusivity and co-occurrence.

To the best of our knowledge, the task of mining (non-redundant, significant) patterns of mutual exclusiveness has not yet been explored. There do however exist proposals to generalize pattern mining towards richer boolean expressions, including disjunctions [24, 33, 36], but these are again limited to frequency as a measure of interestingness. Closer to our goal are approaches to mine association rules with negative dependencies [2, 11], as whenever we discover both $A \rightarrow \neg B$ and $B \rightarrow \neg A$, we can conclude mutual exclusiveness between items A and B, i.e. infer $A \otimes B$. This gives us the first baseline approach we will consider in the experiments, i.e. we mine significant association rules using KINGFISHER [11], and post-process its results to identify patterns of mutual exclusivity.

Another related approach is that of mining low entropy sets [13], which are itemsets for which the contingency table exhibits low entropy. Low entropy sets hence generalize frequency, and can detect any type of dependency, including mutual exclusivity. We consider this as the second baseline, where we simply mine itemsets with an entropy lower than τ, and post-process the result to identify those that exhibit mutual exclusivity.

3 PRELIMINARIES

In this section we discuss preliminaries and introduce notation.

3.1 Notation

We consider binary transaction data. Let $\mathcal{I}$ be a set of items, e.g. products for sale at a store. A transaction $t \in \mathcal{P}(\mathcal{I})$ then corresponds to the set of products a customer bought. A database D over $\mathcal{I}$ is a bag of transactions, e.g. the sales transactions in a month. In general, X denotes an itemset $X \subseteq \mathcal{I}$. We say that a transaction t contains an itemset $X \subseteq \mathcal{I}$ iff $X \subseteq t$.

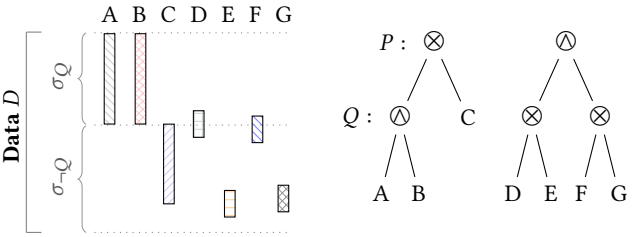


Figure 1: *Toy database with example pattern forest.* Left: Database D of items $A, ..., G$. Item presence is indicated by bars. Transactions where pattern-subtree Q holds are indicated by σ_Q, the remainder of data where the tree does not apply by $\sigma_{\neg Q}$. Right: Pattern trees of database D.

As we are interested in patterns of co-occurrence, mutual exclusivity, as well as combinations thereof, we need a slightly richer notation than usual in pattern mining. First, we need the projection $\pi_X(D)$ of a database D onto an itemset X, which yields the intersection of each transaction $t \in D$ with X, i.e. $\pi_X(D) = \{t \cap X \mid t \in D\}$. Second, we need the selection $\sigma_c(D)$ of a logical condition c over database D, which yields all transactions $t \in D$ that satisfy c, i.e. $\sigma_c(D) = \{t \in D \mid c(t) \equiv \top\}$. We call the number of transactions where this condition *holds*, its support $supp_D(c) = |\sigma_c(D)|$.

We denote the logical k-ary AND by $\textcircled{\wedge}_{c_1,...,c_k}$. For a given transaction t it resolves to $\top$ iff all the given conditions hold, i.e. $\left(\textcircled{\wedge}_{c_1,...,c_k}(t) \equiv \top\right) \leftrightarrow \left(\forall_{i=0}^{k} c_i(t) \equiv \top\right)$. Analogue, we denote the logical k-ary XOR by $\otimes_{c_1,...,c_k}$, which resolves to $\top$ iff exactly one of the provided conditions holds, i.e. $\left(\otimes_{c_1,...,c_k}(t) \equiv \top\right) \leftrightarrow \left(\exists_j c_j(t) \equiv \top \wedge \forall_{i \neq j} c_i(t) \equiv \bot\right)$. We connect conditions back to items $I \in \mathcal{I}$ by introducing the base case $\textcircled{\wedge}_I(t) \equiv \top \leftrightarrow I \in t$. Vice versa, $it(c)$ gives us the itemset X of all items involved in a condition c. We then have the projection of D onto condition c as $\pi_{it(c)}(D)$.

To ease notation, we will write $\odot(c_1, \ldots, c_k)$ wherever t is clear from context. In addition, we will directly use single items $I \in \mathcal{I}$ as conditions, instead of writing $\textcircled{\wedge}(I)$. As an example, we will write $\textcircled{\wedge}(A, B, C)$ and $\otimes(A, B, C)$ to denote the co-occurence pattern resp. the pattern of mutual exclusivity over ABC.

We can express more complex patterns by using hierarchies of conditions, e.g. we can express by $\otimes(\textcircled{\wedge}(A, B), \textcircled{\wedge}(C, D))$ that AND-pattern AB holds mutually exclusively with AND-pattern CD. In other words, a condition c forms a *pattern tree*, with conditions $c \in \{\otimes, \textcircled{\wedge}\}$ as inner nodes, and items $I \in \mathcal{I}$ as leafs. From now on, we will use formulas, patterns, and pattern trees interchangably.

3.2 Minimum Description Length

The Minimum Description Length (MDL) principle [29] is a computable and statistically well-founded approximation of Kolmogorov complexity [18]. For given data D, MDL identifies the best model M^* in a given model class $\mathcal{M}$ as that model that yields the best lossless compression. In one-part, or, *refined* MDL we consider the length in bits of describing data D using the entire model class, $L(D \mid \mathcal{M})$, which gives strong optimality guarantees [10] but is only feasible for certain model classes. In practice we hence often use two-part MDL, which is defined as $L(M) + L(D \mid M)$. Here $L(M)$ is the length of the description of the model, and $L(D \mid M)$

the length of the description of the data using M. We will use two-part codes where we have to, and one-part codes where we can. Note that in MDL we are only concerned with code *lengths*, not actual codes. Since we are interested in measuring lengths in bits, all logarithms are base 2, and we use $0 \log 0 = 0$.

4 THEORY

To solve our problem using MDL, we need to formally define a model class $\mathcal{M}$. As we use two-part codes, we will further need to define a code length function that gives the number of bits needed to describe a model, and a code length function that yields the number of bits needed to describe the data at hand given a model. Before we formally introduce our approach, we provide the intuitions of the problem and how MDL naturally lends itself to solve it.

4.1 The Problem, informally

Given a database, our goal is to find a set of patterns that together succinctly describe the data. Here, we are interested in patterns that capture co-occurence as well as mutually exclusive relationships in the data. These can be simple relationships, such as the co-occurence of items A, B, C, captured by $\textcircled{\wedge}(A, B, C)$, or the mutual exclusive occurence of them, captured by $\otimes(A, B, C)$. We are also interested in more complex, nested relationships, e.g. two item sets $X = \{X_1, ..., X_i\}$ and $Y = \{Y_1, ..., Y_j\}$, with items in X co-occuring, and items in Y co-occuring, but X and Y occuring mutually exclusive, which results in $\otimes(\textcircled{\wedge}(X_1, ..., X_i), \textcircled{\wedge}(Y_1, ..., Y_j))$. With $X = \{A, B\}$, $Y = \{C\}$ we show this pattern tree in Fig. 1.

We hence define a model $M \in \mathcal{M}$ as a set of pattern trees $\mathcal{P}$, which we refer to as a pattern forest. We require that every M always contains all singleton tree $\textcircled{\wedge}(I)$ for all $I \in \mathcal{I}$. This gives us the baseline encoding that ensures we can always model any data D over $\mathcal{I}$. Wherever a non-singleton pattern tree in M holds, however, we will transmit the corresponding data accordingly. As an example, considering Fig. 1 again, we can succinctly transmit where A, B, C hold by sending all transactions where pattern tree P holds in one go, rather than sending this for each item individually. This allows us to detect patterns even of low support, as if the corresponding items co-occur sufficiently strongly, encoding them together will reduce the code length.

Overall, we aim to find that model $M^* \in \mathcal{M}$ such that the overall cost for the model and data is minimal.

4.2 MDL for Pattern Forests

We will now formalize an MDL score based on the intuition of pattern forests above. First, we describe how to compute the model costs and then define the cost of transmitting data given a model .

Encoding a Model. To transmit a model, we first send how many pattern trees are there in our model M. We then send each pattern tree along with the items used in each tree. Our model costs are thus defined as

$$L(M) = L_{\mathbb{N}}(|M|) + \sum_{P \in M} \Bigg(\underbrace{\log \binom{|\mathcal{I}|}{|it(P)|} + L_{pc}(|\mathcal{I}|, 2)}_{\text{NML code}} + L^D(P) \Bigg) + \sum_{I \in \mathcal{I}} L_{st}(I) \,,$$

where the last term is the cost for the singleton stumps. We transmit the number of trees in the forest using the MDL-optimal code $L_{\mathbb{N}}(n)$ for integers $n \geq 1$, which is defined as $L_{\mathbb{N}}(n) = \log^* n + \log c_0$, where $\log^*(n) = \log n + \log\log n + ...$, and c_0 is a constant chosen such that $L_{\mathbb{N}}$ satisfies the Krafft-inequality [30]. To send an individual pattern tree $P \in M$, we first transmit which items $it(P) \subseteq \mathcal{I}$ are used in the tree, which we do using a refined MDL code, in particular the Normalized Maximum Likelihood (NML) code for multinomials. This code consists of the cost of the data using an optimal prefix code over the model class—the log multinomial—and the so-called parametric complexity L_{pc} that optimally encodes the complexity of this model class. For details we refer to Kontkanen et al. [16]. Once we know the relevant items, we proceed to transmit the actual tree using $L^D(P)$, which we define next.

We encode trees recursively, starting from the root. For every node we have to use 1 bit to encode whether it is an internal node or a leaf. If P is a leaf, we are done, and have $L^D(P) = 1$. If P is an internal node, we have to additionally encode the type of the operator ($\textcircled{\wedge}, \otimes$), the number of children $|ch(P)|$, and the items $it(Q)$ each child $Q \in ch(P)$ contains, after which we can recurse.

For an internal $\textcircled{\wedge}$ node P with children $ch(P) = \{Q_1, \ldots, Q_k\}$, and $D' = \sigma_P(D)$ that part of the data where P holds,

$$L^D(P) = 2 + L_{\mathbb{N}}(|ch(P)|) + L_{pc}(|D|, 2) + \sum_{Q_i \in ch(P)} L^{D'}(Q_i) + L_{pc}(|it(P)|, |ch(P)|) + \log \binom{|it(P)|}{|it(Q_1)|, \ldots, |it(Q_k)|} \,,$$

where the terms on the second line together encode the relevant items per child. Analogue, whenever P is an $\otimes$ node, we have

$$L^D(P) = 2 + L_{\mathbb{N}}(|ch(P)|) + \sum_{i=1}^{k} \left(L_{pc}(|D'_{\geq k}|, 2) \right) + \sum_{Q \in ch(P)} L^{D'_{\geq k}}(q) + L_{pc}(|it(P)|, |ch(P)|) + \log \binom{|it(P)|}{|it(Q_1)|, \ldots, |it(Q_k)|} \,,$$

where $D'_{\geq k} = D \setminus (\cup_{j=1}^{k-1} \sigma_{Q_j})$ that data excluding transactions covered by children before Q_k. Finally, for singleton trees P_I we get

$$L_{st}(P_I) = 1 + L_{pc}(|D|, 2) \,,$$

where we use 1 bit to indicate the root is a leaf node, and the parametric complexity L_{pc} for the log binomial over all rows.

Encoding the Data. To encode data D using a model M, we make use of the information that the pattern trees $P \in M$ provide about the dependencies in the data. In particular, we use a pattern P to encode that part of the data where P holds, while we encode the remaining data using the singleton trees P_I for each singleton $I \in \mathcal{I}$. We will start with encoding the data for which a singleton tree P_I holds, i.e., where I is present. To do so we use optimal data-to-model codes [18], which are essentially an index over a canonically ordered enumeration,

$$L^D\left(\pi_{P_I}(D) \mid P_I\right) = \log \binom{|D|}{|\sigma_{P_I}|} \,.$$

For a non-singleton pattern tree, things are slightly more involved. First, we have to encode where the root node, i.e. the logical formula for this tree, holds, after which we can recurse. We have two cases, that of an $\textcircled{\wedge}$ node, and that of an $\otimes$ node. We start with the former,

where we consider data D and a tree $P_{\textcircled{\wedge}}$ with root node $\textcircled{\wedge}$ with children $ch(P) = \{Q_1, \ldots, Q_k\}$. We first encode for which of the $|D|$ transactions P holds, after which we can recurse for each child Q only for that part of the data $D' = \sigma_{P_{\textcircled{\wedge}}}(D)$ where $P_{\textcircled{\wedge}}$ holds,

$$L^D\big(\pi_{P_{\textcircled{\wedge}}}(D) \mid P_{\textcircled{\wedge}}\big) = \log\binom{|D|}{|\sigma_{P_{\textcircled{\wedge}}}|} + \sum_{k=1}^{i} L^{D'}\big(\pi_{Q_k}(D') \mid Q_k\big) \ ,$$

where $\pi_P(D)$ is the projection of data D on pattern P. Analogue, for a pattern tree $P_{\otimes}$ with an $\otimes$ as root node, we iteratively encode where each child holds, while actively using information about already transmitted children. We have

$$L^D\big(\pi_{P_{\otimes}}(D) \mid P_{\otimes}\big) = \sum_{k=1}^{i} \left(\binom{D'_{\geq k}}{|\sigma_{Q_k}|} + L^{D'_{\geq k}}\big(\pi_{Q_k}(\sigma_{P_{\otimes}}) \mid Q_k\big)\right) .$$

where $D'_{\geq k} = D \setminus (\cup_{j=1}^{k-1} \sigma_{Q_j})$ that data excluding data covered by children before Q_k. Importantly, this encoding is independent of the order in which we iterate over the children (see Apx. A.4).

As an example, consider Fig. 1, where we would like to encode data for pattern P. Following to the equation above, for an $\otimes$ pattern we first encode for each children where they hold, and then recurse, which yields $\log\binom{|D|}{|\sigma_Q|}$ bits for identifying transactions of Q and $\log\binom{|\sigma_{\neg Q}|}{|\sigma_C|}$ bits for identifying transactions of the leaf C. The recursion on the $\textcircled{\wedge}$ child Q yields 0 bits, as we already identified the corresponding transactions with the code for the root node. The same is the case when we recurse on the leaf nodes.

Putting all together, we now send for each pattern tree where it holds and transmit the remaining data with singleton trees,

$$L(D \mid M) = \sum_{P \in M} \Big(L^D\big(\pi_P(D) \mid P\big)\Big) + \sum_{I \in \mathcal{I}} \Big(L^D\big(\pi_I(D) \mid I\big)\Big) ,$$

where the last term corresponding to the singleton trees only transmits transactions that are not covered by a pattern, i.e. we consider a modified transaction multiset $\sigma'_I(t) = \{t \in D \mid I \in t \wedge (\forall P \in M.\ I \notin \pi_P(t))\}$. With the above, we have a lossless encoding for a dataset D given a model M.

4.3 The Problem, formally

With the scores above, we can now formally state the problem.

Problem 1 (Minimal Pattern Forest Problem). *Given a database D over items $\mathcal{I}$, find the smallest set of pattern trees M that minimizes the total description length*

$$L(D, M) = L(M) + L(D \mid M) \ .$$

Although our defined model encoding allows for arbitrary hierarchies over $\otimes$ and $\textcircled{\wedge}$, we can apriori reject certain combinations because we know they will not be insightful. For example, if a node and its children are all $\textcircled{\wedge}$, we can obtain a much simpler model without loss by merging these nodes into a single $\textcircled{\wedge}$ node. We can hence safely exclude directly nested $\textcircled{\wedge}$ nodes from our search.

We also exclude pattern trees with directly connected $\otimes$ nodes—not for reasons of inefficiency, but rather because we are not interested in what these represent. Consider, for example a nested $\otimes$ pattern such as $\otimes(A, \otimes(B, C))$. Rather than expressing that *one* of either A, B, or C holds, which is what we are explicitly interested in, this pattern expresses that an odd number of its items is true. Although arguably an interesting statement to discover in data, it is not mutual exclusivity and hence not what we are after.

Overall, we therefore enforce alternating layers of operators, that is a children of $\otimes$ can only be a $\textcircled{\wedge}$ or a leaf, and a children of $\textcircled{\wedge}$ can only be $\otimes$ or a leaf. To keep all discovered patterns interpretable, we restrict ourselves in practice to pattern trees of depth at most 2.

Discovering the exact solution to the *Minimal Pattern Forest Problem* requires the enumeration of all possible sets of pattern trees, for the simple reason that our score does not exhibit any trivially exploitable structure such as convexity, monotonicity, or sub-modularity. However, the model space is of size

$$|\mathcal{M}| = \sum_{k=1}^{|\mathcal{I}|/2} \sum_{i=2k}^{|\mathcal{I}|} \binom{|\mathcal{I}|}{i} \sum_{\substack{l_1+..+l_k=i \\ l_1,..,l_k \geq 2}} \binom{i}{l_1, .., l_k}$$
$$\cdot \prod_{j=1}^{k} 2 \sum_{m=0}^{l_j/2} \sum_{\substack{h_1,..,h_m \geq 2 \\ 2m \leq h_1+..+h_m \leq l_j}} \binom{h_1 + .. + h_m}{h_1, .., h_m} (l_j - (h_1 + .. + h_m))! \ ,$$

a detailed derivation can be found in Appendix A.1. It is thus practically infeasible to enumerate this search space.

Hence, we resort to heuristics.

4.4 The Chance of Being Exclusive

Before we present our algorithm we first discuss the issue of spurious discoveries. That is, especially in sparse data, we are very likely to find that two or more items are perfectly mutually exclusive but just so by chance. To rule out that we include such spurious patterns in a model, we introduce a statistical test that supplements our MDL score, that is able to rule out patterns in a model that are likely to arise by chance. Formally, we want a statistical test that yields the likelihood of seeing an MDL gain similar or better than observed for a given $\otimes$ pattern over itemset X, assuming independence between the items $I \in X$. It turns out that the MDL gain for a two item pattern $\otimes(A, B)$ is monotone in the joint count.

Theorem 4.1 (Monotonicity of gain). *For items A, B with marginals n_A, n_B and joint n_{AB}, the MDL gain of adding $\otimes(A, B)$ to the model M is smaller than for any $n'_{AB} < n_{AB}$.*

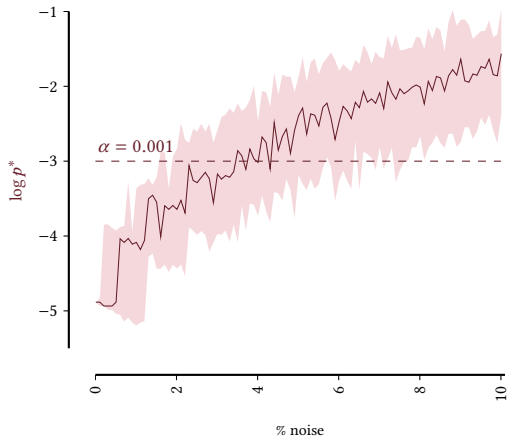
Proof. Proof postponed to Appendix A.2. □

By this theorem we hence know that any datasets for A, B with similar or better gain are exactly those datasets with smaller or equal joint count.

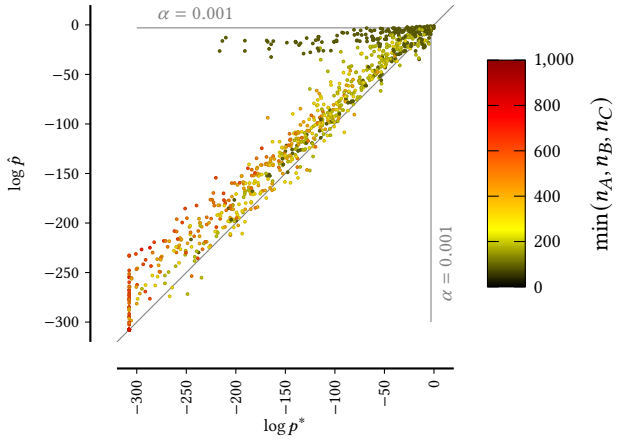
For the simple case of two variables, we can obtain an exact p-value through Fisher's exact test. Fisher's exact test leverages the fact that an observed joint count with fixed marginal counts follows a hypergeometric distribution, hence we can compute the exact p-value by

$$p_2^* = \sum_{i=0}^{n_{AB}} \frac{\binom{n_A}{i}\binom{n-n_A}{n_B-i}}{\binom{n}{n_B}} \ ,$$

where n_X is the number of rows that contain X. For given significance thresholds α, this yields an exact test to decide for the simple case of 2-ary $\otimes$ pattern if it is significant or not—and which allows evaluating the influence of noise on mutually exclusivity over A, B.



(a) Exact p-value p^* for experiments of 2-ary $\otimes$ with varying levels of noise in the data. Significance threshold α is given as dashed line, 25% and 75% quantiles are indicated by the red band.



(b) Approximate p-values $\hat{p}$ against exact p-values p^* for experiments of 3-ary $\otimes$ with varying margins. Minimum margin for an experiment is indicated by color.

Figure 2: Sensitivity of Fisher's exact test regarding noise (left) and evaluation of the p-value approximation (right).

To do so, we generate data of $n = 1000$ transactions, where we plant a perfect $\otimes$ pattern with margins $n_A = n_B = 100, n_{AB} = 0$, and vary the level of noise. In particular, we add noise by flipping $\{0.01, 0.11, ..., 10\}$ % entries uniformly at random. For each noise level we generate 100 folds, measure the p-value, for which we report the median and 25% to 75% quantile range in Figure 2a. We see that already at very modest amounts of noise (2%) the observed gains could just as well be by chance, and clearly illustrates the need for a statistical test.

However, while with Fisher's exact test we have a test for the case of 2 variables, there is no hypergeometric distribution we can use for three variables—we have to resort to plain combinatorics, and enumerate all possibilities of observing data with the given marginals, given by

$$p_3^* = \sum_{i=0}^{n_{AB}} \sum_{j=0}^{n_{AC}} \sum_{k=0}^{n_{BC}} \sum_{l=0}^{n_{ABC}} \binom{n}{n_{\mathrm{A}}} \binom{n_A}{n_{\mathrm{AB}}} \binom{n_{\mathrm{AB}}}{n_{\mathrm{ABC}}} \binom{n - n_{\mathrm{A}}}{n_{\mathrm{B}} - n_{\mathrm{AB}}} \binom{n_{\mathrm{B}} - n_{\mathrm{AB}}}{n_{\mathrm{BC}} - n_{\mathrm{ABC}}}$$
$$\binom{n - n_{\mathrm{A}} - n_{\mathrm{B}} + n_{\mathrm{AB}}}{n_{\mathrm{C}} - n_{\mathrm{BC}} - n_{\mathrm{AC}} + n_{\mathrm{ABC}}} \binom{n_{\mathrm{A}} - n_{\mathrm{AB}}}{n_{\mathrm{AC}} - n_{\mathrm{ABC}}} \Big/ \left(\binom{n}{n_{\mathrm{A}}} \binom{n}{n_{\mathrm{B}}} \binom{n}{n_{\mathrm{C}}} \right) .$$

While somewhat doable for patterns of three items and small joint counts, this quickly becomes infeasible otherwise: if we generalize this formula to K-ary XOR, the number of sums grows to $2^K - K - 1$, while the number of terms in each summation grows to $2^K - 1$. That is, the computational complexity of just computing the p-value for K-ary XOR grows exponentially in K. As we are explicitly interested in arbitrary sized sets of mutually exclusive items, we hence need an alternative solution. To this end, we present a good approximation[1] for the p-value of K-ary XOR, for which the computation time is independent of K.

Suppose we know that for a 3-ary XOR ABC that $\otimes(A, B)$ is significant for a given significance threshold α under p_2^*. Then, we can use an adaptation of Fisher's exact test to approximate p_3^*, by treating AB as a new item that we call $\otimes_{AB}$. Furthermore, we denote by n_{AB} the number of rows that contain both A and B hence are not mutually exclusive. We then get an approximation by

$$\hat{p}_3 = \sum_{i=0}^{n_{ABC}} \frac{\binom{n_{\otimes_{AB}} + n_{AB}}{i} \binom{n - n_{\otimes_{AB}} - n_{AB}}{n_C - i}}{\binom{n}{n_C}} .$$

More generally, for two sets of variables X, Y for which we know $\otimes(X)$ and $\otimes(Y)$ is significant, the p-value of $\otimes(X \cup Y)$ is approximated by

$$\hat{p} = \sum_{i=0}^{n_{\widetilde{X}\widetilde{Y}}} \frac{\binom{n_{\widetilde{X}}}{i} \binom{n - n_{\widetilde{X}}}{n_{\widetilde{Y}} - i}}{\binom{n}{n_{\widetilde{Y}}}} ,$$

where $n_{\widetilde{X}} = |\{t \in D \mid (t \cap X) \neq \emptyset\}|$ is the number of rows where at least one of the items of X is present. Note that we thus essentially condensed X and Y each in a new item $\widetilde{X}$ and $\widetilde{Y}$. Since we are now working on the case of two variables again, Theorem 4.1 applies and we can leverage the fact that the gain is monotone in the joint count. Furthermore, we can use the following recursive definition of terms q_i – the i-th term in the summation – that drastically reduces numerical instabilities for large n,

$$\hat{p}^0 = \frac{\binom{n - n_{\widetilde{X}}}{n_{\widetilde{Y}}}}{\binom{n}{n_{\widetilde{Y}}}} ,$$

$$\hat{p}^i = \hat{p}^{i-1} \cdot \frac{(n_{\widetilde{X}} - i)(n_{\widetilde{Y}} - i)}{(i+1)(n - n_{\widetilde{X}} - n_{\widetilde{Y}} + i + 1)} .$$

We provide a proof of this recurrence in Appendix A.3.

To explore how well we approximate the true p-value, we generate data with $n = 500, m = 3$, vary the marginal densities for each of the three variables in $\{5, 10, ..., 100\}$, the joints between each of the two variables in $\{0, 5, 10, 15\}$ and the full joint in steps of 2 up to the minimum pairwise joint. This leaves us with a total of 144000 experiments covering the space of very sparse to dense dataset margins and combinations of those. In our tests, $\hat{p}_3$ shows to be a

[1] We refer to it Fischer's inexact test.

good, slightly more conservative approximation of p_3^* that deviates from the exact value only if an item I has low support n_I in the order of ten rows (see Fig. 2b), while computation is on average 200 times faster. Knowing that with larger K-ary XOR the running time increases exponentially for the exact test, while our approximation remains constant regarding K, we get a good, computable approximation that allows us to prune for insignificant mutually exclusive patterns. While due to multiple hypothesis testing issues that plague every significant pattern mining approach, we cannot claim significance of the $\otimes$ patterns, we can leverage this test as a powerful filtering technique to prevent many spurious patterns to be even considered in the model.

5 MEXICAN

To discover high quality pattern forests in practice, we propose the Mexican algorithm,[2] which leverages heuristics to efficiently explore the search space. To discover patterns representing mutual exclusivity and co-occurrences that are easily interpretable, we restrict the depth of pattern trees to 2 and alternating operators between layers. The Mexican algorithm can however be trivially adapted to discover patterns of arbitrary depth d.

5.1 Merging Trees

The main idea of Mexican is that, instead of enumerating all possible models, we iteratively refine the current model by combining pattern trees in the current model. We do so as follows.

Mexican starts with a model M that only consists of singleton trees. We can combine two singleton trees A and B by introducing a new root, yielding $\otimes(A, B)$ and $\oslash(A, B)$ as candidate pattern trees. A pattern tree $\otimes(A, B)$ and a singleton C we can combine in the following two ways. We either merge C into the XOR, and have $\otimes(A, B, C)$, or we create a new *conjunctive* node with both patterns as children, i.e. we have $\oslash(\otimes(A, B), C)$. Analogue, a pattern tree $\oslash(A, B)$ and singleton C we can again either merge, and have $\oslash(A, B, C)$, or combine under a new mutual exclusivity root node, and have $\otimes(\oslash(A, B), C)$.

If we have two pattern trees with root nodes of the same type, e.g. $\oslash(A, B)$ and $\oslash(C, D)$, resp. $\otimes(A, B)$ and $\otimes(C, D)$, we can either merge them and obtain $\oslash(A, B, C, D)$ resp. $\otimes(A, B, C, D)$, or combine them by introducing a new root node of the alternate kind, and have $\otimes(\oslash(A, B), \oslash(C, D))$ resp. $\oslash(\otimes(A, B), \otimes(C, D))$. Overall, we require that the depth of the new tree is ≤ 2, and that operators alternate along a path from root to leaf.

Summarizing the above, we create candidate patterns from pairs of pattern trees P and R as follows:

- Make R and P children of a new root $r \in \{\otimes, \oslash\}$
- Make R a new child of P
- Merge R with existing child Q of P
  - if $root(R) = root(Q)$, add children of R to children of Q
  - if Q is singleton, make Q child of R, and R child of P
- Merge R and P that have same root operator

We thus obtain an algorithm `mergeTrees(`M`)` that given a current model M yields a set of candidate patterns to refine M. Although heuristic, this scheme does explore large parts of the relevant search space: intuitively, a conjunction specifies that items usually co-occur, and hence that all subsets of these items usually co-occur, and similarly so for mutual exclusive patterns. This is captured by the idea of our bottom-up search.

[2] for summarizing using *M*utual *EX*clus*I*ve and *C*onjunctive p*A*tter*N*s

Algorithm 1: Mexican

input : Dataset D, Significance threshold α
output: Heuristic approximation to the optimal model M^*

1 $M \leftarrow \mathcal{I}$; // Initialize model with singletons
2 **do**
3 $\quad C \leftarrow$ `mergeTrees(`M`)` ; // Generate candidates
4 $\quad M' \leftarrow M$;
5 $\quad \Delta' \leftarrow 0$;
6 $\quad$ **for** $P \in C$ **do**
7 $\quad\quad \Delta \leftarrow L(D, M \oplus P) - L(D, M)$; // Compute gain
8 $\quad\quad p \leftarrow 0$;
9 $\quad\quad$ **if** $root(P) = \otimes$ **then** // If candidate is XOR
10 $\quad\quad\quad p \leftarrow \hat{p}$ of P ; // Compute p-Value
11 $\quad\quad$ **if** $\Delta < \Delta'$ and $p < \alpha$ **then** // Update current best
12 $\quad\quad\quad M' \leftarrow M \oplus P$;
13 $\quad\quad\quad \Delta' \leftarrow \Delta$;
14 $\quad M \leftarrow M'$; // Update best model
15 **while** $L(D, M)$ decreases;
16 **return** M

5.2 Algorithm

Putting together the candidate generation strategy and the MDL code length definitions, we arrive at Mexican, for which we give the pseudocode in Algorithm 1. Starting with the set of all singleton trees as initial model (line 1), we iteratively refine our model by generating a set C of candidate pattern trees using `mergeTrees` (line 3), from which we find the pattern P that gives the best gain $\Delta = L(D, M \oplus P) - L(D, M)$ in terms of our previously defined MDL score (line 7). In case the candidate is an $\otimes$ pattern, we also compute the p-Value estimate $\hat{p}$ (line 9, 10) and filter it out if it is above the signifance threshold α (line 11). We add the pattern P with the highest gain to the model, $M \oplus P$, while removing all non-singleton patterns that were used to construct P from M (line 12). If there is no candidate that would decrease the current codelength, we terminate and return the current best model (line 15, 16).

5.3 Complexity

Normally, we analyze the complexity of an alogrithm in terms of the size of the input. Here, the theoretical worst case time complexity in terms of the input size is exponential in the number of items. However, if we only consider small models – which is what MDL ensures – this statement is neither insightful nor a tight bound. Thus, we will analyze the complexity in terms of the size of the discovered model rather than the input. The following theorems capture complexity properties of Mexican in terms of the number of found patterns k, a proof of which can be found at Apx. A.5.

Theorem 5.1 (#Candidates of Mexican). *Given that we mine k pattern trees, with at most l leafs each, for a given dataset D, Mexican evaluates at most $O((k \cdot l + m)^2 l^2)$ candidates per iteration.*

It now remains to include the number of iterations, which results in the following time complexity of Mexican.

Theorem 5.2 (Runtime of Mexican). *Given that we mine k pattern trees with at most l leafs each for a given dataset D, the runtime of Mexican is in $O(k \cdot l \cdot (k \cdot l + m)^2 l^2)$.*

Proof. From Th. 5.1 and the observation that in each iteration one of the tree grows by at least 1 in the number of leafs because of the pairwise merge, we know that there are at most $k \cdot l$ merges possible. Otherwise we would get a tree that is larger than any tree in the final model. □

6 EXPERIMENTS

We empirically evaluate Mexican on both synthetic data with known ground truth as well as on real world data. For that, we implemented Mexican in C++. There is no direct competitor that is able to mine mutually exclusive patterns, we instead compare to the two closest cousins. The first method that we compare to is an implementation for mining low entropy patterns, LEminer [13]. To compare to LEminer, we postprocessed the results to extract patterns where at least some subset of features has few overlapping transactions with the rest of the features, which can be seen as mutual exclusive patterns over itemsets. The second method is an adaptation of Kingfisher that derives mutual exclusive patterns from statistically significant association rules [11]. We reimplemented Kingfisher in C++, such that it scales to larger datasets and processed the results to merge rules $A \rightarrow \neg B$ and $B \rightarrow \neg A$ to $\otimes(A, B)$. Note that Kingfisher is only able to find rules with single items in the consequent, hence we can only mine for mutual exclusive feature pairs. We want to emphasize at this point that both methods LEminer and Kingfisher were originally not designed to discover patterns of mutual exclusivity and the comparison is thus slightly unfair. However, these methods come closest to what we do. All experiments were carried out single-threaded on Intel Xeon E5-2643 v3 machines with 256GB RAM running Linux. Mexican finishes within seconds to minutes on all synthetic and real data, with the exception of the single cell and Instacart data, for which it needs hours, due to the large width respectively height of the data sets. The code and all data is available online[3].

6.1 Synthetic data

As a sanity check, we first consider data without any structure. We generate data sets of size 1000×100 with d% 1s that are set uniformly at random, and report the average number of found patterns across 10 folds for each method. We show the results in Fig. 3a, and see that while Mexican recovers the ground truth in all cases but one, its competitors Kingfisher and LEminer, discover up to millions of spurious patterns[4].

[3] http://eda.mmci.uni-saarland.de/mexican/

[4] Note that similar behaviour of state-of-the-art methods based on frequency or statistical testing on pure noise have also been reported regarding conjunctive patterns [8]

			Kf	Le	Mexican		
Dataset	n	m	$\#\otimes$	$\#\odot^k$	$\#\barwedge$	$\#\otimes$	$\#\odot^k$
Accidents	340K	468	38K	216M^*	1	34	6
DLQ	10K	4.2K	830K	34M^*	10	130	4
DNA	1.3K	392	20K	1B^*	114	0	0
Mammals	2.2K	121	5K	198K	4	15	12
SC	65	20.2K	63M	0*	131	500	41
Instacart	2.6M	1236	-	-	1	95	3

Table 1: Reported are number of rows n and columns m in data set and number of found AND and XOR patterns $\#\barwedge$, $\#\otimes$, and nested patterns $\#\odot^k$ for Kingfisher (Kf), LEminer (Le), and Mexican. By * we indicate the forced termination of LEminer after either 1 Billion patterns were mined, or >500GB of disk space was consumed. Resulting numbers are patterns returned on termination, hence a lower bound.

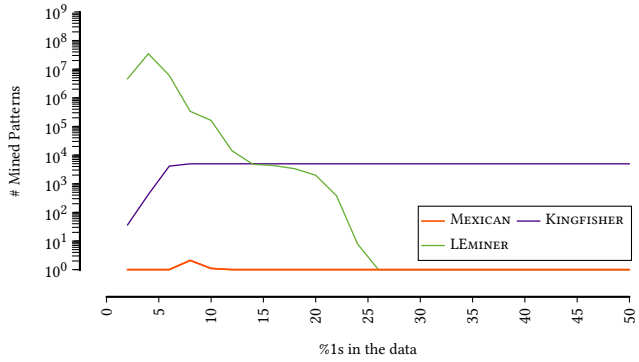
Simple patterns. To evaluate whether these methods can recover ground truth beyond pure noise, we generate data where we plant 10 pure $\otimes$, respectively pure $\barwedge$ patterns over at most 5 items in data with $n = 10000$ rows. We add noise to this data by flipping 0.1% of the entries – an average of 10 entries per column – uniformly at random, ensuring that $\otimes$ patterns are not spurious (see also Fig. 2a).

We generate 10 datasets for each of the two setups, and record the number of patterns each method discovers. We find that on average Kingfisher reports hundreds, and LEminer more than a million patterns for each of the two experiments. In contrast, Mexican recovers the 10 patterns in the $\barwedge$ data exactly, and on average 7.6 patterns for the $\otimes$ data. As on data without noise it does recover all patterns, this is likely due to the (strong) effect of noise on the significance of $\otimes$ patterns of higher arity.

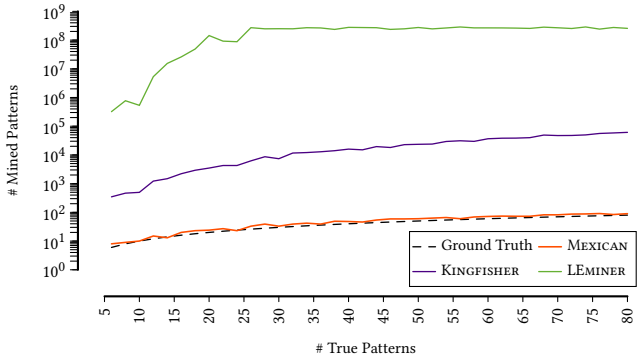
Nested Patterns. Next, we investigate how well Mexican performs on data that contains more complex patterns, in particular including nested logical formulas, we plant $k \in \{5, 7, ..., 80\}$ pattern trees into $n = 10000$ rows. We draw uniformly at random $c \in \{2, 3, 4\}$ children for the root. For each of those children we draw up to 2 nodes as their children. We then draw an operator $o \in \{\barwedge, \otimes\}$ for each inner node. For each pattern tree we draw rows from $\mathcal{N}(500x, 10x)$, where x is the maximum number of children of any $\otimes$ subtree, or 1 if none exists, to avoid overly sparse features. We then partition the rows for each $\otimes$ subtree according to a multinomial over the children. Finally, we apply noise as in the previous experiment. We run each of the methods, and report the number of discovered patterns in Figure 3b. We see that, Kingfisher and LEminer discover thousands up to billions of patterns, whereas Mexican recovers models that are—both in numbers, as well as upon close inspection—very close to the ground truth.

6.2 Real-World Data

To evaluate Mexican on real data, we look at six different binarized datasets over different domains. We consider data on Belgian traffic *Accidents*, lemmatized words from abstracts of Deep Learning and Quantum Computing papers on ArXiv (*DLQ*) [6], *DNA* amplification data [22], European *Mammals* [20], and Single Cell



(a) Patterns found in pure noise data of different sparsity.



(b) Patterns found in synthetic data with planted patterns. Numbers of LEMiner are a lowerbound as each run was terminated after 1 hour.

Figure 3: *Synthetic data* Left: #Patterns found in pure noise data. Right: #Patterns found in data with planted patterns.

RNA-sequencing (*SC*) data [3]. Data dimensions and results are reported in Tab. 1. These results show that Mexican is able to retrieve succinct, hence interpretable, sets of patterns, where the other methods discover orders of magnitude more patterns than there are rows and columns in the data. Furthermore, these results show that our method scales up to many thousands of features and millions of rows. Both Kingfisher and LEMiner are not able to process *Instacart*. Examining the results of Mexican by operator type, we observe that Mexican for some data sets discovers many $\otimes$ patterns, especially sparse data such as *Instacart*, whereas other data that has highly correlated margins, such as *DNA*, yields more $\oplus$ patterns. When appropriate, Mexican discovers also more complex, nested patterns that describe the data well, which we examine further below. The experiments further show that Mexican is fast despite the theoretically challenging problem, taking only seconds for *DNA* and *Mammals*, up to hours on data of very high dimensionality or sample size, such as *Instacart* and *SC*. So far, we focused on the pattern set size, which is a good indicator of accessibility of the sets to a human expert, however does not provide any qualitative statement about the results. In the following we examine the quality of pattern sets for *DLQ*, and *SC*.

DLQ data. For the *DLQ* dataset we find informative patterns that capture the discrepancies between papers of the two communities. By observing that discovered patterns reflect classical word co-occurences, such as $\oplus$(monte, carlo) or $\oplus$(nearest, neighbour), and the XOR pattern $\otimes$($\oplus$(deep, learning), quantum) that summarizes how the data was generated, it is clear that Mexican is able to infer patterns that capture main properties of the data. Close inspection of the patterns further reveals that we are able to retrieve more subtle distinctions between the Deep Learning and Quantum Computing fields, such as $\otimes$(adversarial, free), or $\otimes$(stochastic, superposition). Furthermore, Mexican discovers larger patterns that might indicate subdomains within Deep Learning or Quantum Computing, such as $\otimes$(operation, radiation, autoencoder), or $\otimes$(layer, atom, cryptography, hamiltonians, reality, remote).

Single Cell Data. Finally, we look at the case of the single cell sequencing data set. Single cell sequencing is a recent breakthrough in genetics that can be compared to that of deep learning in machine learning: it enables to measure genetic and epigenetic features for a separate, single cells instead of cell batches that were measured previously. Hence, instead of analysing only global averages over all cell states, we can now obtain the state of each cell individually, and thus capture the patterns underlying cellular dynamics. The major challenge of such data is the large number of features as we are interested in analyzing all (several tens of thousands of) genes. Furthermore, there is only a limited understanding of the whole gene regulatory system. This is both a limitation in terms of how we can validate but also an opportunity to suggest new relationships derived from our discoveries.

For the *SC data*, many patterns discovered by Mexican reflect distinct local mechanisms within the cellular life[5]. One example is the pattern $\oplus$(IMPDH2, UMPS). Both of these genes are essential for the synthesis of building blocks of the DNA and thus are key for cell proliferation. In particular, UMPS is responsible for the final two steps of pyrimidine synthesis, a buildung block for certain nucleotids (the N in DNA) [17]. IMPDH2 catalyzes a crucial step in the synthesis of the nucleotide Guanine [5]. Due to its direct effect on the supply of Guanine and thus the rate of cell proliferation, IMPDH2 is also used as drug target for e.g. cancer treatment [23].

Another example is $\otimes$(ZNF692, BRF1) for which we can hypothesize an anthagonistic role of the two genes influencing gene regulation. In particular, ZNF692 encodes a protein that acts as a transcriptional repressor [15], while BRF1 is a component of RNAPolIII, responsible for the transcription of genes [14].

Finally, with the pattern $\otimes$(RP11.446E9.1, SETD1B, MIOX), we can suggest novel relationships between genes. MIOX is related to the Polyol metabolism, and is tightly regulated by DNA methylation in its promoter [32]. SETD1B, on the other hand, is part of a complex modifying Histone proteins [31]. These modifications are known epigenetic markers for gene regulation. Thus, the Lysine modification introduced by SETDB1 might play a repressive role in the regulation of MIOX, an interesting subject of further study.

[5]Please note that due to the history of discovery, many genes have been assigned more than one name. The aliases can be looked up at e.g. https://www.genecards.org/.

These findings show that MEXICAN discovers a succinct set of patterns that characterise cellular mechanisms, and thus can be leveraged to guide future research by suggesting potential relationships as subject for further investigations.

7 DISCUSSION AND CONCLUSION

We considered the problem of discovering patterns of co-occurence and mutual exclusivity. In particular, we proposed a pattern language over logical conjunctions and mutual exclusivity, and defined the goal of discovering succinct and non-redundant sets of patterns over this language that together generalize the data.

We defined the problem in terms of the Minimum Description Length principle, and as the resulting score does not lend itself for efficient exact search, we proposed an effective heuristic approach called MEXICAN to efficiently approximate the optimal solution. To filter out spurious results, we proposed a computationally efficient statistical test approximation for K-ary mutual exclusivity.

With the among the state of the art unique ability of discovering both co-occurrences as well as mutual exclusive relationships, we show that MEXICAN gives an extended view on the distribution of the data by conducting experiments with synthetic and real world data. On synthetic data, we showed that MEXICAN is able to recover the ground truth without picking up on noise, where the state-of-the-art methods discovered millions of patterns even when there are none. Through experiments on real data we confirmed that MEXICAN consistently returns succinct pattern sets interpretable by human experts, scaling up to millions of rows and thousands of features. Close inspection revealed that patterns MEXICAN discovered are indeed meaningful and correspond to domain knowledge.

One major application of a pattern language equipped with mutual exclusivity is biological data on the gene regulatory system, where such patterns could indicate replacable sub-components or antagonistic players in a pathway. To showcase the efficacy of MEXICAN in this domain, we considered a case study on single cell RNA sequencing data. The discovered patterns reflect local cellular mechanisms that we validated with the literature, but also suggest new relationships of genes about only little is known so far, which could be subject to further experiments.

Although MEXICAN meets the goals we set for this work and yields highly encouraging results, our ultimate goal is to summarize data in terms of patterns of arbitrary logical statements—that is, including negations, disjunctions, rules, etc. Each of these extensions alone are far from trivial, and the development of a single approach that incorporates all will make for engaging future work.

REFERENCES

[1] R. Agrawal, T. Imielinksi, and A. Swami. 1993. Mining association rules between sets of items in large databases. In *SIGMOD*. ACM, 207–216.

[2] M.-L. Antonie and O. R. Zaïane. 2004. Mining Positive and Negative Association Rules: An Approach for Confined Rules. In *PKDD*. Springer, 27–38.

[3] F. B. Ardakani, K. Kattler, K. Nordström, N. Gasparoni, G. Gasparoni, S. Fuchs, A. Sinha, M. Barann, P. Ebert, J. Fischer, B. Hutter, G. Zipprich, C. D. Imbusch, B. Felder, J. Eils, , B. Brors, T. Lengauer, T. Manke, P. Rosenstiel, J. Walter, and M. H. Schulz. 2018. Integrative analysis of single-cell expression data reveals distinct regulatory states in bidirectional promoters. *Epigen. & chrom.* 11, 1 (2018), 66.

[4] T. Calders and B. Goethals. 2002. Mining all Non-Derivable Frequent Itemsets. In *PKDD*. 74–85.

[5] S. F. Carr, E. Papp, J. C. Wu, and Y. Natsumeda. 1993. Characterization of human type I and type II IMP dehydrogenases. *J. Biol. Chem.* 268, 36 (1993), 27286–27290.

[6] S. Dalleiger and J. Vreeken. 2020. Explainable Data Decompositions. AAAI.

[7] T. De Bie. 2011. Maximum entropy models and subjective interestingness: an application to tiles in binary databases. *Data Min. Knowl. Disc.* 23, 3 (2011), 407–446.

[8] J. Fischer and J. Vreeken. 2019. Sets of Robust Rules, and How to Find Them. Springer.

[9] J. Fowkes and C. Sutton. 2016. A Subsequence Interleaving Model for Sequential Pattern Mining. In *KDD*.

[10] P. Grünwald. 2007. *The Minimum Description Length Principle*. MIT Press.

[11] W. Hämäläinen. 2012. Kingfisher: an efficient algorithm for searching for both positive and negative dependency rules with statistical significance measures. *Knowl. Inf. Sys.* 32, 2 (2012), 383–414.

[12] J. Han, J. Pei, and Y. Yin. 2000. Mining frequent patterns without candidate generation. In *SIGMOD*. ACM, 1–12.

[13] H. Heikinheimo, J. K. Seppänen, E. Hinkkanen, H. Mannila, and T. Mielikäinen. 2007. Finding low-entropy sets and trees from binary data. In *KDD*. 350–359.

[14] Y. J. Hsieh, T. K. Kundu, Z. Wang, R. Kovelman, and R. G. Roeder. 1999. The TFIIIC90 subunit of TFIIIC interacts with multiple components of the RNA polymerase III machinery and contains a histone-specific acetyltransferase activity. *Mol. Cell. Biol.* 19, 11 (1999), 7697–7704.

[15] E. Inoue and J. Yamauchi. 2006. AMP-activated protein kinase regulates PEPCK gene expression by direct phosphorylation of a novel zinc finger transcription factor. *Biochem. Biophys. Res. Commun.* 351, 4 (2006), 793–799.

[16] P. Kontkanen and P. Myllymäki. 2007. A linear-time algorithm for computing the multinomial stochastic complexity. *Inf. Process. Lett.* 103, 6 (2007), 227–233.

[17] J. Krungkrai, N. Wutipraditkul, P. Prapunwattana, S. R. Krungkrai, and S. Rochanakij. 2001. A nonradioactive high-performance liquid chromatographic microassay for uridine 5'-monophosphate synthase, orotate phosphoribosyltransferase, and orotidine 5'-monophosphate decarboxylase. *Anal. Biochem.* 299, 2 (2001), 162–168.

[18] M. Li and P. Vitányi. 1993. *An Introduction to Kolmogorov Complexity and its Applications*. Springer.

[19] M. Mampaey, J. Vreeken, and N. Tatti. 2012. Summarizing Data Succinctly with the Most Informative Itemsets. *ACM TKDD* 6 (2012), 1–44. Issue 4.

[20] T. Mitchell-Jones. 1999. Societas Europaea Mammalogica. http://www.european-mammals.org. (1999). http://www.european-mammals.org

[21] F. Moerchen, M. Thies, and A. Ultsch. 2011. Efficient mining of all margin-closed itemsets with applications in temporal knowledge discovery and classification by compression. *Knowl. Inf. Sys.* 29, 1 (2011), 55–80.

[22] S. Myllykangas, J. Himberg, T. Böhling, B. Nagy, J. Hollmén, and S. Knuutila. 2006. DNA copy number amplification profiling of human neoplasms. *Oncogene* 25, 55 (2006), 7324–7332.

[23] R. Naffouje, P. Grover, H. Yu, A. Sendilnathan, K. Wolfe, N. Majd, E. P. Smith, K. Takeuchi, T. Senda, S. Kofuji, and A. T. Sasaki. 2019. Anti-Tumor Potential of IMP Dehydrogenase Inhibitors: A Century-Long Story. *Cancers (Basel)* 11, 9 (2019).

[24] A. A. Nanavati, K. P. Chitrapura, S. Joshi, and R. Krishnapuram. 2001. Mining Generalised Disjunctive Association Rules. In *CIKM*. ACM, 482–489.

[25] L. Papaxanthos, F. Llinares-López, D. A. Bodenham, and K. M. Borgwardt. 2016. Finding significant combinations of features in the presence of categorical covariates. In *NIPS*. 2271–2279.

[26] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. 1999. Discovering Frequent Closed Itemsets for Association Rules. In *ICDT*. ACM, 398–416.

[27] L. Pellegrina, M. Riondato, and F. Vandin. 2019. SPuManTE: Significant Pattern Mining with Unconditional Testing. In *KDD*. ACM, 1528–1538.

[28] L. Pellegrina and F. Vandin. 2018. Efficient Mining of the Most Significant Patterns with Permutation Testing. In *KDD*. 2070–2079.

[29] J. Rissanen. 1978. Modeling by shortest data description. *Automatica* 14, 1 (1978), 465–471.

[30] J. Rissanen. 1983. A Universal Prior for Integers and Estimation by Minimum Description Length. *Annals Stat.* 11, 2 (1983), 416–431.

[31] D. C. Schultz, K. Ayyanathan, D. Negorev, G. G. Maul, and F. J. Rauscher. 2002. SETDB1: a novel KAP-1-associated histone H3, lysine 9-specific methyltransferase that contributes to HP1-mediated silencing of euchromatic genes by KRAB zinc-finger proteins. *Genes Dev.* 16, 8 (2002), 919–932.

[32] I. Sharma, R. K. Dutta, N. K. Singh, and Y. S. Kanwar. 2017. High Glucose-Induced Hypomethylation Promotes Binding of Sp-1 to Myo-Inositol Oxygenase: Implication in the Pathobiology of Diabetic Tubulopathy. *Am. J. Pathol.* 187, 4 (2017), 724–739.

[33] Y. Shima, S. Mitsuishi, K. Hirata, and M. Harao. 2004. Extracting Minimal and Closed Monotone DNF Formulas. In *DS*. Springer, 298–305.

[34] J. Vreeken, M. van Leeuwen, and A. Siebes. 2011. KRIMP: Mining Itemsets that Compress. *Data Min. Knowl. Disc.* 23, 1 (2011), 169–214.

[35] G. I. Webb. 2010. Self-sufficient itemsets: An approach to screening potentially interesting associations between items. *ACM TKDD* 4, 1 (2010), 1–20.

[36] M. Zaki, N. Ramakrishnan, and L. Zhao. 2010. Mining Frequent Boolean Expressions: Application to Gene Expression and Regulatory Modeling. *IJKDB* 1 (09 2010), 68–96.

[37] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. 1997. New algorithms for fast discovery of association rules. In *KDD*.

A THEORETICAL RESULTS

A.1 Size of the model space

The size of the model space is given by

$$|\mathcal{M}| = \sum_{k=1}^{|\mathcal{I}|/2} \sum_{i=2k}^{|\mathcal{I}|} \binom{|\mathcal{I}|}{i} \sum_{\substack{l_1+..+l_k=i \\ l_1,..,l_k \geq 2}} \binom{i}{l_1,..,l_k} \cdot \prod_{j=1}^{k} 2 \sum_{m=0}^{l_j/2} \sum_{\substack{h_1,..,h_m \geq 2 \\ 2m \leq h_1+..+h_m \leq l_j}} \binom{h_1+..+h_m}{h_1,..,h_m} (l_j - (h_1+..+h_m))! \,.$$

In order of terms in the first line, the first sum indicates the number of trees in a model, the second sum the number of items covered by these trees, the first binomial the number of subset of this size, the third sum together with the first multinomial indicates the possible partitionings of this item subset over the trees. In the second line we describe the number of trees possible, given by the first product, and then multiply by 2, which is the number of ways we can fix the operators. We then have to go over all different tree shapes. For that, we first sum over all the possible number of inner nodes in the tree. The second sum in the second line together with the first multinomial gives the partition of items over the leafs of these inner nodes. The last term gives the combinations of distributing the remaining items over the leafs that are children of the root.

A.2 Monotonicity result

Theorem A.1 (Monotonicity of gain). *For features A, B with marginals n_A, n_B and joint n_{AB}, the MDL gain of adding $\otimes(A, B)$ to the model M is smaller than for any $n'_{AB} < n_{AB}$.*

Proof. We will show that the MDL costs for a joint count $n_{AB}+1$ is the cost for joint count n_{AB} plus some $\log \epsilon > 0$. We will start off with the MDL costs, given by the costs of transmitting transactions covered by the XOR pattern and transmitting the transactions where A, B overlap using the singleton code. The model costs can be ignored as they are the same for different joint counts of the same pattern. Hence, we get

$$\begin{aligned}
&\log\binom{n}{n_\text{A}-1} + \log\binom{n-n_\text{A}+1}{n_\text{B}-1} + 2\cdot\log\binom{n}{n_\text{AB}+1} \\
&\overset{(1)}{=} \log\left(\frac{n_\text{A}}{n-n_\text{A}+1}\binom{n}{n_\text{A}}\right) + 2\cdot\log\left(\frac{n-n_\text{AB}}{n_\text{AB}+1}\binom{n}{n_\text{AB}}\right) \\
&\quad + \log\left(\frac{n-n_\text{A}+1}{n_\text{B}-1}\binom{n-n_\text{A}}{n_\text{B}-2}\right) \\
&\overset{(2)}{=} \log\left(\frac{n_\text{A}}{n-n_\text{A}+1}\binom{n}{n_\text{A}}\right) + 2\cdot\log\left(\frac{n-n_\text{AB}}{n_\text{AB}+1}\binom{n}{n_\text{AB}}\right) \\
&\quad + \log\left(\frac{n-n_\text{A}+1}{n_\text{B}-1}\frac{n_\text{B}}{n-n_\text{A}-n_\text{B}+1}\frac{n_\text{B}-1}{n-n_\text{A}-n_\text{B}+2}\binom{n-n_\text{A}}{n_\text{B}}\right) \\
&\overset{(3)}{=} \log\binom{n}{n_\text{A}} + \log\binom{n-n_\text{A}}{n_\text{B}} + 2\cdot\log\binom{n}{n_\text{AB}} \\
&\quad + \log\Bigg(\underbrace{\frac{n_\text{A}n_\text{B}(n-n_\text{AB})^2}{(n-n_\text{A}-n_\text{B}+1)(n-n_\text{A}-n_\text{B}+2)(n_\text{AB}+1)^2}}_{=\epsilon}\Bigg) \,.
\end{aligned}$$

For equality (1), we use the well known equations $\binom{n}{k-1} = \frac{k}{n-k+1}\binom{n}{k}$ and $\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$ to change the n, respectively k of the binomial coefficient by 1. Equality (2) is a recursive application of the first binomial coefficient equation. Equality (3) is essentially reordering and cancelling out terms and pulling the 2 into the logarithm. Starting with the costs of transmitting the data using the pattern with joint count $n_{AB} - 1$ we thus arrived at the costs of transmission costs for the pattern with joint n_{AB} plus some $\log \epsilon$. It remains to show that $\epsilon \geq 1$.

We will now bound each of the terms in the denominator of ϵ by one of the numerator terms from above. Assume that *a)* $n_A, n_B \geq 1$, *b)* $n_A > n_{AB}$, and *c)* $n_B > n_{AB}$. Then we get

$$\begin{aligned}
&n_A \overset{b)}{\geq} n_{AB}+1, \quad n_B \overset{c)}{\geq} n_{AB}+1 \,, \\
&n - n_{AB} \overset{b)}{\geq} n - n_A + 1 \overset{a)}{\geq} n - n_A - n_B + 2 > n - n_A - n_B + 1 \,.
\end{aligned}$$

It follows that $\epsilon \geq 1$, which complestes the proof. □

A.3 Fisher's recurrence

We now proof by induction that the following recurrence relation holds for the terms $\hat{p}_i$ of the summation in $\hat{p}$

$$\hat{p}_0 = \frac{\binom{n-n_{\widetilde{X}}}{n_{\widetilde{Y}}}}{\binom{n}{n_{\widetilde{Y}}}} \,,$$

$$\hat{p}_i = \hat{p}_{i-1} \cdot \frac{(n_{\widetilde{X}}-i)(n_{\widetilde{Y}}-i)}{(i+1)(n-n_{\widetilde{X}}-n_{\widetilde{Y}}+i+1)} \,.$$

Proof. *Induction base.* Assume $n_{\widetilde{X}\widetilde{Y}} = 0$. Obviously holds as $\hat{p}_0$ is the the original equation.
Induction step. Assume that the equation holds for i. Then we get

$$\begin{aligned}
&\binom{n_{\widetilde{X}}}{i+1}\binom{n-n_{\widetilde{X}}}{n_{\widetilde{Y}}-i-1} \Big/ \binom{n}{n_{\widetilde{Y}}} \\
&= \frac{n_{\widetilde{X}}-i}{i+1}\binom{n_{\widetilde{X}}}{i}\binom{n-n_{\widetilde{X}}}{n_{\widetilde{Y}}-i-1} \Big/ \binom{n}{n_{\widetilde{Y}}} \\
&= \frac{n_{\widetilde{X}}-i}{i+1}\frac{n_{\widetilde{Y}}-i}{n-n_{\widetilde{X}}-n_{\widetilde{Y}}+i+1}\binom{n_{\widetilde{X}}}{i}\binom{n-n_{\widetilde{X}}}{n_{\widetilde{Y}}-i} \Big/ \binom{n}{n_{\widetilde{Y}}}
\end{aligned}$$

by using the well known identity for binomials $\binom{n}{k} = \frac{n-k+1}{k}\binom{n}{k-1}$. □

A.4 Order independence of XOR

For the length of the encoding for an XOR node of a pattern tree, or more precisely for the encoding of the transaction that this tree encodes, it does not matter in which order we send the children. We will prove it for the case of 3 children, the case for an arbitrary number of l children follows the same reasoning.

Theorem A.2. *Given a node $P = \otimes(i, j, k)$ with corresponding margins n_i, n_j, n_k of the children, it does not matter in which order we send where the children hold using $L_{D'}(\pi_P(D') \mid P)$.*

Proof. We essentially need to show that we can flip the children order without changing the cost, for that assume a new order $P =$

$\otimes(k,i,j)$, then we show that

$$\log\binom{n}{n_i} + \log\binom{n-n_i}{n_j} + \log\binom{n-n_i-n_j}{n_k}$$
$$\stackrel{!}{=} \log\binom{n}{n_k} + \log\binom{n-n_k}{n_i} + \log\binom{n-n_i-n_k}{n_j}.$$

We use the definition of the binomial with factorials, use the standard rules for logarithmic arithmetic to pull the binomials apart, and then add new terms that add up to 0 to derive the equation above.

$$\log\frac{n!}{(n-n_i)!n_i!} + \log\frac{(n-n_i)!}{(n-n_i-n_j)!n_j!} + \log\frac{(n-n_i-n_j)!}{(n-n_i-n_j-n_k)!n_k!}$$
$$= \log(n!) - \cancel{\log((n-n_i)!)} - \log(n_i!) + \cancel{\log((n-n_i)!)}$$
$$- \cancel{\log((n-n_i-n_j)!)} - \log(n_j!) + \cancel{\log((n-n_i-n_j)!)}$$
$$- \log((n-n_i-n_j-n_k)!) - \log(n_k!)$$
$$\underbrace{+\log((n-n_k)!) - \log((n-n_k)!)}_{=0}$$
$$\underbrace{+\log((n-n_i-n_k)!) - \log((n-n_i-n_k)!)}_{=0}$$
$$= \log\frac{n!}{(n-n_k)!n_k!} + \log\frac{(n-n_k)!}{(n-n_i-n_k)!n_i!} + \log\frac{(n-n_i-n_k)!}{(n-n_i-n_j-n_k)!n_j!}$$

The other permutations as well as the case for more than 3 children follow the exact same resoning. □

A.5 Mexican candidate evaluations

Here, we provide the proof for how many candidates Mexican generates and evaluates in one iteration, given that the final result are k pattern trees with maximum l leafs each.

Theorem A.3 (#Candidates of Mexican). *Given that we mine k pattern trees, with at most l leafs each, for a given dataset D, Mexican evaluates at most $O((\lfloor\frac{kl}{2}\rfloor + m)^2 l^2)$ candidates per iteration.*

Proof. In a given iteration, we can have at most m singletons and $k' = \lfloor\frac{kl}{2}\rfloor$ pattern trees that together form a model. The term k' is the upper bound that we get when we divide the trees in the final model into trees of size 2, which is the smallest possible tree as otherwise we would have a singleton. We can merge each tree with any other tree resulting in $(k'+m)(k'+m-1)/2$ tree pairs to look at. Without making further assumptions on the shape or content of the tree, we have to assume that we can merge one tree in any leaf or inner node of the other tree. There are at most l leafs and at most $\lfloor l/2\rfloor + 1$ inner nodes, using the constraint that a tree has at most depth 2. For each merge, including merging under a new root or as a new child, we can use two different operators. This gives at most $(k'+m)(k'+m-1)\cdot 2\cdot(2\cdot(l+(\lfloor l/2\rfloor+1)+(\lfloor l/2\rfloor+1))+1)$ candidates. □

B EXPERIMENTS

B.1 Program calls

Mexican. To rule out exploring unreasonable patterns, we constrain Mexican to candidates with a minimum overlap of 50% for $\triangle$, resp. a maximum overlap of 10% for $\otimes$. Furthermore we set the minimum support to 0.1% of the number of rows in the data and a conservative significance threshold $\alpha = 0.001$. To keep the results interpretable for the *SC* data, we set a minimum support threshold for a pattern to 3.

Kingfisher. We set a minimum confidence of 50% for the rule mining and a minimum support of 0.1% of the number of rows in the data and a conservative significance threshold $\alpha = 0.001$, similar to Mexican.

LEminer. To somewhat limit the number of patterns that LEminer finds, we set the maximum number of bits to 1.5 and the dependency threshold to 0.7 and only mine attribute sets (`-s`).

B.2 Data preprocessing

B.2.1 DLQ data. The arXiv data base was queried with keywords *Deep Learning*, respectively *Quantum Computing* to retrieve paper abstracts. Words in all abstracts were lemmatized, and words that are not nouns, verbs or adjectives got removed. Five thousand abstracts of each of the two categories were put together in a data set. The words of all abstracts were treated as bag of words and encoded as separate binary columns, words with a support of less than 10 were removed from the bag.

B.2.2 Instacart. The Instacart dataset originally has many items of the same type but different vendors (e.g. tens of different toilet papers), while transactions are usually very short. This makes the data very sparse, driving the support for even small patterns down to single digits, which in context of a database with more than 2 million rows is nothing. Hence we decided to process the data by hand to summarize similar items into buckets, and focussing on the food data. We provide the processed data set online under TODO.

B.2.3 Single Cell Data. The original single cell data was processed to have a transcript level normalization, by using the established TPM (Transcripts Per Million) values. Furthermore, we carried out a per cell normalization, setting the 75% quantile of gene expression levels to 0, the upper quantile to 1, as common in the literature. Columns containing only zeroes were removed and the resulting data was converted into a transaction file format.