Label-Descriptive Patterns and Their Application to Characterizing Classification Errors

Michael A. Hedderich [*1] **Jonas Fischer** [*2] **Dietrich Klakow** [1] **Jilles Vreeken** [3]

Abstract

State-of-the-art deep learning methods achieve human-like performance on many tasks, but make errors nevertheless. Characterizing these errors in easily interpretable terms gives insight into whether a classifier is prone to making systematic errors, but also gives a way to act and improve the classifier. We propose to discover those feature-value combinations (ie. patterns) that strongly correlate with correct resp. erroneous predictions to obtain a global and interpretable description for arbitrary classifiers. We show this is an instance of the more general label description problem, which we formulate in terms of the Minimum Description Length principle. To discover a good pattern set, we develop the efficient PREMISE algorithm. Through an extensive set of experiments we show it performs very well in practice on both synthetic and real-world data. Unlike existing solutions, it ably recovers ground truth patterns, even on highly imbalanced data over many features. Through two case studies on Visual Question Answering and Named Entity Recognition, we confirm that PREMISE gives clear and actionable insight into the systematic errors made by modern NLP classifiers.

1. Introduction

State-of-the-art deep learning methods are known for their ability to achieve human-like performance on challenging tasks. As much as 'to err is human,' these classifiers make errors too. Some of these errors are due to noise that is inherent to the process we want to model, and therewith relatively benign. Systematic errors, on the other hand, e.g. those due to bias or misspecification, are much more serious as these lead to models that are inherently unreliable. If we know under what conditions a model performs poorly, we can actively intervene, e.g. by augmenting the training data, and so improve overall reliability and performance. Before we can do so, we first need to know whether a classifier makes systematic errors, and if so, how to characterize them in easily understandable terms.

Instances	Correct Prediction?
How many ducks are in the picture?	✗
What are the ducks eating?	✗
How many roosters are in the puddle?	✗
Do you see ducks in the puddle?	✓
Are there many ducks playing?	✓

Figure 1. Toy example with input instances and the label specifying if the classifier predicted correctly. The pattern ⩓(*how, many*) correlates with misclassification. The word *ducks* is also a frequent pattern but independent of the label and therefore not of relevance.

Given a dataset with labels that specify which instances were classified correctly or incorrectly, we are interested in finding combinations of features that describe where the classifier's predictions are incorrect. For a Natural Language Processing (NLP) task, the input features are words. If, for example, the combination of words "*how, many*" strongly correlates with misclassified instances, this can indicate that our classifier struggles with the concept of counting. A toy example is visualized in Figure 1.

Local explanation methods like LIME (Ribeiro et al., 2016) describe the decision boundary of each instance. In contrast, we are interested in an efficient way to obtain a global and non-redundant description of our classifier's issues on the given input data. To this end, we turn to data mining. Here, a combination of features is a pattern, and we look for the set of patterns that best characterizes on which instances the classifier tends to perform well or poorly. This can be phrased as the more general problem of label description. For data with binary features, we are interested in the associations between the feature data and the labels. We formulate

[*]Equal contribution [1]Saarland University, Saarland Informatics Campus, Saarbrücken, Germany. [2]Max Planck Institute for Informatics, Saarbrücken, Germany. [3]CISPA Helmholtz Center for Information Security, Saarbrücken, Germany. Correspondence to: Michael A. Hedderich <mhedderich@lsv.uni-saarland.de>, Jonas Fischer <fischer@mpi-inf.mpg.de>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

this problem in terms of the Minimum Description Length (MDL) principle, which identifies the best set of patterns as the one that best compresses the data without loss.

To capture phenomena of text input, e.g. synonyms, we consider a rich pattern language that allows us to express conjunctions, mutual exclusivity, and nested combinations thereof. As the search space is twice exponential and does not exhibit any easy-to-exploit structure, we propose the efficient and hyper-parameter-free PREMISE algorithm to heuristically discover the *premises* under which we see the given predictions.

We evaluate PREMISE both on synthetic and real-world data. We show that, unlike the state of the art in data mining, PREMISE is robust to noise, scales to large numbers of features, and deals well with class imbalance, as well as varying association strength of patterns to labels. Through two case studies, we show that PREMISE discovers patterns that provide clear insight into the systematic errors of NLP classifiers. For Visual Question Answering (VQA), we elucidate the issues of two classifiers (Tan & Bansal, 2019; Zhu et al., 2016), including aspects like counting, spatial orientation and higher reasoning. For a neural Named Entity Recognition (NER) model (Ma & Hovy, 2016), we show that PREMISE discovers patterns that are both interpretable and that can be acted upon.

2. Related Work

Label Description in Data Mining

Describing labels in terms of features is obviously related to classification. Here, however, we are not so much interested in prediction, but rather description and therewith value interpretability of the results over accuracy. We share this notion with emerging pattern mining (Dong & Li, 1999) which aims to discover those conditions under which a target attribute has an exceptional distribution. The key difference is that we are not interested in discovering *all* patterns that are associated, which would be overly redundant and hard to interpret as a whole, but rather want a small and non-redundant set of patterns capturing relevant associations.

Subgroup discovery (García-Vico et al., 2018; Wrobel, 1997) returns the top-k patterns that correlate most strongly. This keeps the result sets of manageable size but does not solve the problem of redundancy (van Leeuwen & Knobbe, 2012). Statistical pattern mining aims to discover patterns that correlate *significantly* to a class label (Llinares-López et al., 2015; Papaxanthos et al., 2016; Pellegrina & Vandin, 2018; Webb, 2007). In practice, these methods discover many hundreds of thousands of 'significant' patterns even for small data. For surveys we refer to (Atzmueller, 2015; García-Vico et al., 2018; Novak et al., 2009).

Rule mining aims to discover rules of the form $X \rightarrow Y$ (Agrawal et al., 1993; Hämäläinen, 2012), lending themselves to describe labels, too. Like above, most existing methods evaluate patterns individually, thereby discovering millions of rules even if the data is pure noise. GRAB (Fischer & Vreeken, 2019) instead mines small sets of rules that together summarize the data well, an approach that is strongly related to ours. One drawback for our setting, however, is that a rule found by GRAB does not make a statement about the data where the rule does *not* apply. Here, we are however interested in exactly this differential description: where does the consequent appear predominantly in one label and not the other. CLASSY (Proença & van Leeuwen, 2020) instead discovers rule lists that specifically characterize a given label. We show that both these approaches do not scale well and are sensitive to label imbalance.

In pattern mining, the goal is instead to describe binary data in terms of relevant feature co-occurrences. Specific works based on boolean matrix factorization (Miettinen, 2012) or the minimum description length principle (Budhathoki & Vreeken, 2015) addressed the problem of finding patterns that are common and distinct between multiple databases. While this can be applied in our setting, where labels induce a partition of the data into multiple databases, these algorithms do not scale to the applications that we consider. The boolean matrix factorization approach C-SALT (Hess & Morik, 2017) was specifically designed for data with given class labels. Our results show, however, that it has low recall, missing most interesting patterns.

Explainable ML and Misclassification

Specifically to explain classifiers, several approaches aim to capture dependencies of features or attributes that a classifier uses to make a prediction, e.g. in terms of patterns or rules (Barakat & Diederich, 2005; Henelius et al., 2014), by model distillation (Frosst & Hinton, 2017; Lakkaraju et al., 2017), or to discover patterns of neurons within neural networks that drive a decision (Fischer et al., 2021). These, however, focus on the dependencies the classifier exploits for successful prediction as opposed to understanding where – or why – something goes wrong. Here, Duivesteijn & Thaele (2014) use the CORTANA tool (Meeng & Knobbe, 2011) to explain where a classifier performs particularly poorly in terms of feature subspaces. SliceFinder (Chung et al., 2020) follows a similar idea. However, both models were only evaluated on data with less than 50 features. Our experiments show that these methods do not scale well to the feature spaces common in NLP data.

Among the first methods of rule mining in NLP are template-based information extraction approaches (Califf & Mooney, 1999; Riloff & Wiebe, 2003). These distantly related methods require pre-specified templates for which then rules

that fulfill the templates can be extracted from given text. More recently, for specific applications in NLP, manual approaches based on challenging test sets (Gardner et al., 2020; Ribeiro et al., 2020) or testing a hypothetical cause for misclassification (Lee et al., 2019; Rondeau & Hazen, 2018; Wu et al., 2019) have been suggested. Such manual approaches, however, require existing knowledge about the difficulties of the models. Local explanation methods like LIME (Ribeiro et al., 2016) provide insights into what changes in the input influence a classifier's decision on a specific instance. ANCHORS (Ribeiro et al., 2018) obtains such explanations in an interpretable form similar to our patterns. As they need to explore the local decision boundary, they require, however, multiple classifier evaluations per instance. For a survey focused on local methods for explainable NLP, we refer to Danilevsky et al. (2020).

Here, we propose to mine sets of patterns that provide concise, interpretable, and global descriptions of the given label, which we formulate in MDL terms. We further propose an efficient heuristic to discover such pattern sets in practice, which we test against state-of-the-art across all aforementioned fields on synthethic data with known ground truth, as well as real world case studies. We show that PREMISE is the only approach to be scalable and robust to noise and label imbalance while retrieving succinct pattern sets, all of which is crucial to tackle real world applications.

3. Preliminaries

In this section, we introduce the notation we use throughout the paper and give a brief primer to MDL.

3.1. Notation

We consider binary data, such as a sequence of input words of an NLP task where each word of the vocabulary is a binary feature (bag-of-words, word is present or not present). In data mining terms, each instance of our dataset is a transaction and each word present in the instance is an item of the transaction. For each instance, we also have a label that expresses whether the instance is misclassified by our classifier. Our whole dataset can then be described as binary transaction data D over a set of items $\mathcal{I}$, where each transaction $t \in D$ is assigned a binary label $\ell(t) \in \{l_-, l_+\}$. For ease of notation, we define the partition of the database according to this binary label $D^- = \{t \in D \mid \ell(t) = l_-\}$ and $D^+ = \{t \in D \mid \ell(t) = l_+\}$. In general, $X \subseteq \mathcal{I}$ denotes an itemset, the set of transactions that contain X is defined as $T_X = \{t \in D \mid X \subseteq t\}$. The projection of D on an itemset X is $\pi_X(D) = \{t \cap X \mid t \in D\}$.

We are looking for human-interpretable associations of items that best explain a given database partition. We describe these associations in terms of 'patterns', which we define by logical conditions over sets of items. For a logical condition c, we define a selection operator as $\sigma_c(D) = \{t \in D \mid c(t) \equiv \top\}$. For an item $I \in \mathcal{I}$, it holds that $[c_I(t) \equiv \top \leftrightarrow I \in t]$. The k-ary AND operator $\bigwedge(c_1, \ldots, c_k)$ describes patterns of co-occurrence and holds iff all its conditions hold. Similarly, the k-ary XOR operator $\bigotimes$ describes patterns of mutual exclusivity and holds if exactly one of its condition holds. We denote $it(c)$ for the items in the condition and define the projection on a condition as $\pi_c(D) = \pi_{it(c)}(D)$. Conditions can be nested; specifically we are interested in patterns of AND operator over XOR operations, i.e. $\bigwedge(\bigotimes_{c_1,\ldots,c_k}, \ldots, \bigotimes_{c'_1,\ldots,c'_k})(t)$. An XOR operation is called clause, $\gamma(c)$ lists all clauses in conjunctive condition c. To simplify notation, we drop t when clear from context, write I for conditions on a single item $c(I)$, and use condition and pattern interchangeably.

3.2. Minimum Description Length

The Minimum Description Length (MDL) principle (Rissanen, 1978) is a practical approximation of Kolmogorov complexity (Li & Vitányi, 1993) that is both statistically well-founded and computable. It identifies the best model M^* for data D out of a class of models $\mathcal{M}$ as the one that obtains the maximal lossless compression. For refined, or one-part, MDL, the length of the encoding in bits is obtained using the entire model class $L(D|\mathcal{M})$. While this variant of MDL provides strong optimiality guarantess (Grünwald, 2007), it is only attainable for certain model classes. In practice, crude two-part MDL is often used, which computes the length of the model encoding $L(M)$ and the length of the description of the data given the model $L(D|M)$ separately. The total length of the encoding is then given as $L(M) + L(D|M)$. We use one-part MDL where possible and two-part MDL otherwise. Here, we are only interested in the codelengths and not the actual codes. Codelengths are measured in bits, hence all log operations are base 2 and we define $0 \log 0 = 0$.

4. Theory

To discover those patterns best describing the given labels, we first provide an intuition behind the concepts of the Minimum Description Length (MDL) principle in terms of our problem, and then formally introduce the class of models $\mathcal{M}$ and corresponding codelength functions.

4.1. The Problem, informally

Given a dataset of binary transaction data and corresponding binary labels, we aim to find a set of patterns that together identify the partitioning of the data according to the labels. As an application, consider the sequence of input words of an NLP task as transactions, along with labels that express whether an instance is misclassified by a given model. We

are interested in patterns of words that describe these labels to better understand the classifier's errors. In essence, we want to find word combinations such as $\textcircled{\wedge}$(*how, many*), or mutual exclusive patterns, e.g. $\otimes$(*color, colour*), that capture synonyms or different writing styles, all occuring predominantly when a misclassifcation happens. The pattern language we use is a combination of the two, namely conjunctions of mutual exclusive clauses, e.g. $\textcircled{\wedge}$(*what,* $\otimes$(*color, colour*)). We provide an example in Figure 2.

We thus define a model $M \in \mathcal{M}$ as the set of patterns $\mathcal{P}$ that help to describe given labels. We look at this problem of identifying a good model M through the lens of information theory. In a nutshell, we could consider this problem as transmission of the given data where we assume that the receiver knows the labels of the data. We would first send the model containing the patterns, which we then use to send the data to the receiver. For patterns describing the labels – the partitions (D^+, D^-) – well, we can use efficient codes to transmit the database. If we use too many or overly redundant patterns, or ones that do not describe relevant structures, we spend unnecessary bits in the transmission of the model. We are, hence, after the model $M^* \in \mathcal{M}$ that minimizes the cost of transmitting data and model, which is captured by the MDL principle.

To ensure that we can always encode any data, M contains all singleton words $I \in \mathcal{I}$, describing the entire data D without taking labels into account. This model of all singletons also acts as a baseline implementing the assumption that there are no associations that describe the label.

Let us consider the example in Figure 2, where we would first send $\textcircled{\wedge}(A, \otimes(B, C))$ occurrences in D^+, and then its occurrences in D^-. Thus, we identify where A, B, and C hold at once, and we leverage the fact that $\textcircled{\wedge}(A, \otimes(B, C))$ occurs predominantly in D^+, resulting in more efficient transmission. Intuitively, a bias of a pattern to occur in one label more than in the other corresponds to a large deviation between the conditional probability – the pattern occurrence conditioned on the label – and the unconditional probability – the pattern occurence in the whole database. In this case, the codes are hence more efficient when sending a pattern separately for D^+ and D^-.[1] Coming back to the example, F however occurs similarly often in both data partitions – there is almost no deviation between conditional and unconditional probability – hence it is unlikely that it identifies a structural error. Here, the baseline encoding transmitting F as singleton in all of D will be most efficient. This approach allows us to identify patterns that occur predominantly for one of the labels as the patterns that yield better compression when conditioned on the labels, and thus characterise labels in easily understandable terms.

[1]For a link between probabilities and codelengths, consider for example the Shannon Entropy $H(X) = \sum_i P(x_i) \log(P(x_i))$, which gives the minimum number of bits needed to transmit a random variable X.

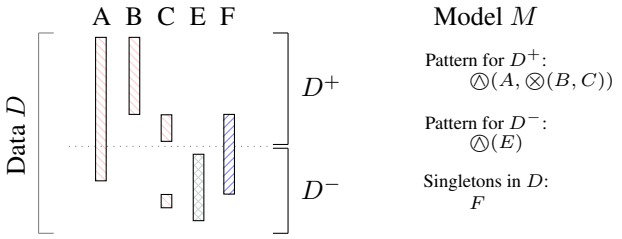


Figure 2. Example database and model. Left: a toy database D over a set of items, separated by labels into D^+ and D^-. Right: the corresponding model M containing patterns describing data partitions D^- and D^+ induced by labels l_- and l_+.

In the following sections, we will formalize this approach using an MDL score to identify that pattern set that best describes the data given the labels. We will first detail how to compute the encoding cost for the data given the model and then the cost for the model itself.

4.2. Cost of Data Given Model

Let us start by explaining how to encode a database D with singleton items I in the absence of any labels, which will later serve as the baseline encoding corresponding to independence between items and labels. To encode in which transaction an item I holds, optimal data-to-model codes are used, which are indices over canonically ordered enumerations (Li & Vitányi, 1993). Hence, the data costs are

$$L\left(\pi_I\left(D\right) \mid I\right) = \log \binom{|D|}{|\sigma_I\left(D\right)|} .$$

Taking into account the partitioning of D along the label, yielding D^+ and D^-, we encode I separately:

$$L\left(\pi_I\left(D\right) \mid I\right) = \log \binom{|D^-|}{|\sigma_I\left(D^-\right)|} + \log \binom{|D^+|}{|\sigma_I\left(D^+\right)|} .$$

As such, we explicitly reward patterns (here, singletons) that have a different distribution between the unconditional probability, i.e. its frequency in D, and the conditional probability of I conditioned on the label – i.e. its frequency in D^- respectively D^+. It models the property that we are interested in: patterns that characterize a certain label. It is straightforward to extend to patterns of co-occurring items $P = \textcircled{\wedge}(X_1, \ldots, X_k)$ by selecting on transactions where the pattern holds

$$L\left(\pi_P\left(D\right) \mid P\right) = \log \binom{|D^-|}{|\sigma_P\left(D^-\right)|} + \log \binom{|D^+|}{|\sigma_P\left(D^+\right)|} .$$

There might be transactions where individual items of $\mathcal{P}$ are present, but not all of P holds. To ensure a lossless encoding, the singleton code $L(\pi_I(D) \mid I)$ is modified to cover all item occurrences left unexplained after transmitting $\mathcal{P}$, i.e.

$$L_s(\pi_I(D) \mid P) = \log \binom{|D|}{|\sigma_I(D) \setminus \left(\bigcup_{P \in \mathcal{P}, I \in P} \sigma_P(D)\right)|} .$$

For patterns expressing conjunctions over mutual exclusive items, e.g. $\owedge(\otimes(A,B), \otimes(C,D))$, we first send for both D^- and D^+ for which transactions the pattern holds, after which we specify which of the items is active where. We specify items one by one, as we know that when the pattern holds and A is present, B cannot be present too. With each transmitted item of the clause, there are thus fewer transactions where the remaining items could occur, hence the codelength is reduced. More formally, the codelength for a pattern P of conjunctions of clauses is given as

$$L(\pi_P(D) \mid P) = \sum_{l \in \{-,+\}} \log \binom{|D^l|}{|\sigma_P(D^l)|} + \sum_{\substack{cl \in \\ \gamma(P)}} \sum_{\substack{I \in \\ cl}} \log \binom{|\sigma_P(D^l)| - \sum_{I' \in cl, I' \leq I} |\sigma_{I'}(\sigma_P(D^l))|}{|\sigma_I(\sigma_P(D^l))|} ,$$

assuming a canonical order on $\mathcal{I}$. With clauses of only length 1 we arrive at a simple conjunctive pattern, and the function resolves to the codelength function for conjunctive patterns discussed above. Note here that the codelength is the same regardless of the order assumed on the $\mathcal{I}$. This statement trivially holds for clauses of length 2, we provide an argument for the case of l items in the Appendix.

This concludes the definition of codelength functions for transmitting the data. The overall cost of transmitting the data D given a model M is hence

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

4.3. Cost of the Model

Let us now discuss how to transmit the model M for pattern set $\mathcal{P}$. First, we transmit the number of patterns $|\mathcal{P}|$ using the MDL-optimal code for integers $L_{\mathbb{N}}(|\mathcal{P}|)$. It is defined as $L_{\mathbb{N}}(n) = \log^* n + \log c_0$ with $\log^* n = \log n + \log\log n + \ldots$ and c_0 being a constant so that $L_{\mathbb{N}}(n)$ satisfies the Kraft-inequality (Rissanen, 1983). Then, for each pattern P, we transmit the number of clauses via $L_{\mathbb{N}}(|\gamma(P)|)$. For each such clause, we transmit the items it contains using a log binomial, requiring $\log \binom{|\mathcal{I}|}{|cl|}$ bits plus a parametric complexity term $L_{pc}(|\mathcal{I}|)$. The log binomial along with the parametric complexity form the normalized maximum likelihood code for multinomials, which is a refined MDL code. The parametric complexity for multinomials is computable in linear time (Kontkanen & Myllymäki, 2007). Lastly, we transmit the parametric complexities of all binomials used in the data encoding.

Combining the above, the overall model cost is

$$L(M) = L_{\mathbb{N}}(|\mathcal{P}|) + \sum_{P \in \mathcal{P}} (L_{\mathbb{N}}(|\gamma(P)|) + L_{pc}(|D^+|) + L_{pc}(|D^-|)) + \sum_{cl \in P} \left(\log \binom{|\mathcal{I}|}{|cl|} + L_{pc}(|\mathcal{I}|)\right) + \sum_{I \in \mathcal{I}} L_{pc}(|D|) .$$

4.4. The Problem, formally

Based on the above, we can now formally state the problem.

MINIMAL LABEL DESCRIPTION PROBLEM *Given data D over $\mathcal{I}$ and partitions D^- and D^+, find model $M \in \mathcal{M}$ that minimizes the codelength $L(M) + L(D \mid M)$.*

Solving this problem through enumeration of all models is computationally infeasible as the model space is extremely large (see App. Sec. A.3), and does not lend itself for efficient search. Hence, we resort to an efficient heuristic for discovering good models.

5. PREMISE

To find good pattern sets in practice, we present PREMISE which efficiently explores the search space in a bottom-up heuristic fashion.

5.1. Creating and Merging Patterns

PREMISE starts with a model M that contains only singletons. It then iteratively improves the model by adding, extending, and merging patterns until it can not achieve more gain in the MDL score. To ease the explanation, we first introduce the setting with conjunctive patterns only.

- *single items*: $I \in \mathcal{I}$ that improves the MDL score when transmitted separately for D^- and D^+,
- *pairs of items*: a new conjunctive pattern $\owedge(I_1, I_2) \in \mathcal{I} \times \mathcal{I}$,
- *patterns and items*: a new conjunctive pattern $\owedge(P, I)$ by merging an existing pattern $P \in M$ with an $I \in \mathcal{I}$,
- *pairs of patterns*: a new conjunctive pattern $\owedge(P_1, P_2)$ obtained by merging two existing patterns $P_1, P_2 \in M$.

We can speed up the search by pruning infrequent and therewith uninteresting patterns. Pairs of items for which the transaction sets barely overlap are unlikely to compress well

as conjunctive patterns. Hence, we introduce a minimum overlap threshold of 0.3 in all experiments. This straightforwardly leads to algorithm `createCandidates` that, based on a current model M, outputs a set of possible candidate patterns that we will consider as additions to the model. We give pseudocode in the App. Sec. A.1.

5.2. Filtering Noise

Additionally to the MDL score, Fischer & Vreeken (2020) proposed to use Fisher's exact test as a filter for spurious patterns. Here, we use it to test our candidate patterns. Fisher's exact test allows to assess statistically whether two items co-occur independently based on contingency tables. We assume the hypothesis of homogeneity; in our case that there is no difference in the pattern's probability between D^- and D^+. Fisher showed that the values of the contingency table follow a hypergeometric distribution (Fisher, 1922). We can then compute the p-value for the one-sided test directly via

$$p = \sum_{i=0}^{\min(a,d)} \frac{\binom{a+b}{a-i}\binom{c+d}{c+i}}{\binom{n}{a+c}}.$$

with $c = |\sigma_P(D)|$, $a = |D| - c$, $d = |\sigma_P(D^+)|$, $b = |D^+| - d$ and $n = |D|$ for a pattern P labeled with l_+. For patterns labeled with l_-, the other tail of the distribution is tested (with a and b as well as c and d switching places). In all experiments, we require $p < 0.01$. A general problem for statistical pattern mining is the lack of an appropriate multiple test correction. We here however only use the test to *filter* candidates, false positive patterns passing the test are still evaluated in terms of MDL.

5.3. The PREMISE Algorithm

Algorithm 1 PREMISE

1: **Input:** D, significance threshold α
2: **Output:** approximation M of M^*
3: **repeat**
4: $\quad\Delta' \leftarrow 0$
5: $\quad M' \leftarrow M$
6: $\quad C \leftarrow$`createCandidates`(M)
7: $\quad$**for** $P \in C$ **do**
8: $\quad\quad\Delta \leftarrow L(D, M \oplus P) - L(D, M)$ *// (neg.) gain*
9: $\quad\quad p \leftarrow$`FisherExactTest`(P) *// p-value*
10: $\quad\quad$**if** $p < \alpha$ and $\Delta < \Delta'$ **then**
11: $\quad\quad\quad\Delta' \leftarrow \Delta$, $M' \leftarrow M \oplus P$
12: $\quad\quad$**end if**
13: $\quad$**end for**
14: $\quad M \leftarrow M'$
15: **until** $\Delta' = 0$

Combining the candidate generation and the MDL score from Section 4, we obtain PREMISE. We give the pseudocode in Algorithm 1. Starting with the empty model, we generate candidates and for each of those, we compute the (negative) gain in terms of MDL (line 7) as well as the pattern's p-value (line 8). We select the candidate below a significance threshold α that reaches the highest gain (line 9-11) and add it to the model. If we created the pattern through a merge, we remove its parent patterns from M. We repeat the process until no candidate provides further gain in codelength. As we are after a concise set of patterns that describe the given label, which MDL ensures to find, we analyze PREMISE's time complexity in terms of the output. To find a set of k patterns of maximum length l over a dataset of m items, PREMISE runs in time $O(k\,l\,(k\,l + m^2))$. For a full derivation and discussion of the algorithm's complexity, see Appendix, Section A.5.

5.4. Mutual Exclusivity

In our practical NLP applications, we are interested, among others, in finding clauses expressing words that are synonyms, that reflect similar concepts, or language variations, such as $\bigwedge$(*which,*$\bigotimes$(*color, colour*)) or $\bigotimes$(*could, can*). Such statements, however, require a pattern language beyond purely conjunctive patterns. We discussed above how to identify the best model over a pattern language of clauses in terms of MDL. Instead of enumerating all possible clauses exhaustively or searching for an XOR structure like in Fischer & Vreeken (2020), for NLP applications, we follow a more informed approach, taking into account information from pre-trained word embeddings. These classifier-independent embeddings indicate word relations such as similar concepts, synonyms, or writing styles, and hence express exactly the relationships that we are interested in. In our candidate search, we consider only those XOR terms that span words that are close in the embedding space (5 closest neighbours). We give details in Appendix, Section A.4.

6. Experiments

We evaluate our approach on synthetic data with known ground truth, as well as on real world NLP tasks to characterise misclassifications. We compare against significant pattern mining (SPuMANTE, Pellegrina et al., 2019), rule sets mining (GRAB, Fischer & Vreeken, 2019), rule lists (CLASSY, Proença & van Leeuwen, 2020), top-k subgroup discovery (SUBGROUP-DISCOVERY, Lemmerich & Becker, 2018), the subgroup discovery tool CORTANA (Duivesteijn & Thaele, 2014; Meeng & Knobbe, 2011) and class-specific matrix factorization (C-SALT, Hess & Morik, 2017). As representatives of interpretable, global machine learning models we consider the rule-learner (RIPPER, Cohen, 1995) and patterns derived from classification trees (TREE). Due to runtime issues, we compare to the local explainability

method (ANCHORS, Ribeiro et al., 2018) only in the NER experiment. For similar reason, we exclude SLICEFINDER (Chung et al., 2020), and disjunctive emerging patterns (Vimieiro, 2012); neither completed a single run within 12 hours. Further details are given in Appendix, Section A.6, with datasets and code available online.[2]

6.1. Synthetic Data

To evaluate against a known ground truth, we generate synthetic data where we insert patterns. Unless specified differently, for each of the experiments we generate a data matrix with $10\,000$ samples, half of which get label l_{-}. The set of items $\mathcal{I}$ has size 1000. We draw patterns of length 2-5 from $\mathcal{I}$ with replacement until 50% of items are covered. For each pattern we then draw $k \sim \mathcal{N}(150, 20)$ and set the items of the pattern in $.9k$ random transactions from D^{+}, and $.1k$ transaction from D^{-} to 1. This corresponds to a typical sparsity level for pattern mining problems. Additionally, for each item that is part in a pattern, we let it occur in $k \sim \mathcal{N}(50, 20)$ random transactions from D. For all items not part of a pattern, we let them occur in $k \sim \mathcal{N}(150, 20)$ transactions from D. Lastly, we introduce background noise by flipping $.1\%$ of the matrix values.

We evaluate all methods with respect to *scalability* (size of item sets $\mathcal{I}$), *label imbalance* (proportion of transactions having label l_{-}), *label shift* (patterns occurring not exclusively in one of the labels) and *robustness to background noise* (flipping a fraction of entries of the data matrix). The results are shown in Figure 3. The performance of most existing methods deteriorates already for data with several hundred items. We observe similar effects for increasing label imbalance which is e.g. encountered in misclassified samples that make up only a small fraction of overall samples. For label shift, we adapt the occurrence of patterns between 1, meaning the pattern occurs exclusively in one partition of the database, to .6, meaning that only 60% of the transaction where a pattern occurs have one label. Again, most baselines struggle with this setting. C-SALT yields relatively good precision for balanced data but its recall is close to 0. SUBGROUP-DISCOVERY, CORTANA and PREMISE are robust to these changes. In most cases, however, SUBGROUP-DISCOVERY and CORTANA yield (soft) F1 scores of less than .4 and .6, respectively, while PREMISE is close to 1.

Synthetic text data. For an evaluation with known ground truth more similar to the NLP application domain, we evaluate how well all methods cope with item – or word – distributions similar to real text. We report these experiments in Fig. 4 with more details in A.8. For most most baselines performance quickly deteriorates for longer patterns. GRAB is able to retrieve longer patterns and is resistant to shift and noise in the form of non-systematic label errors. PREMISE outperforms all competitors, achieving consistently high F1 scores beyond .92. For complex patterns consisting of conjunctive clauses of disjunctions, we verify that PREMISE is able to retrieve them even in the presence of noise.

[2] https://github.com/uds-lsv/premise

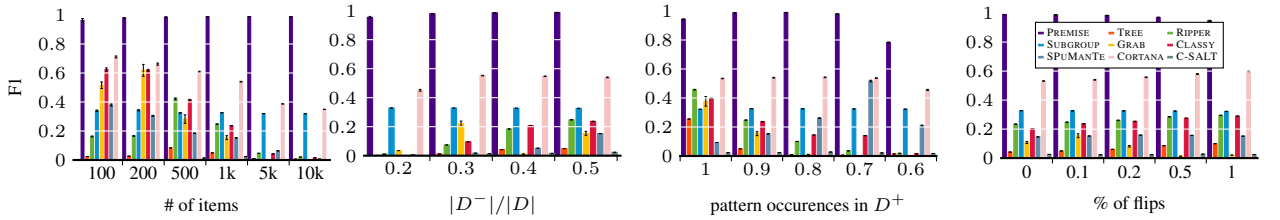
6.2. Real Data: VQA

Visual Question Answering (VQA) is the popular and challenging task of answering textual questions about a given image. We analyze the misclassification of Visual7W (Zhu et al., 2016) and the state-of-the-art LXMERT (Tan & Bansal, 2019), both specific architectures for different VQA tasks. Visual7W reaches 54% accuracy in 4-option multiple choice, LXMERT a validation score of 70%. Both classifiers perform far from optimal and thus serve as interesting applications for describing (misclassification) labels. We derive misclassification data sets from applying the classifiers to the development sets.

SPUMANTE as well as TREE retrieve several hundred or thousand patterns making it difficult to interpret the results. Furthermore, we know from the previous experiments that these methods find thousands of patterns even when there exist only few ground truth patterns. SUBGROUP-DISCOVERY requires the user to specify the number of patterns a-priori, which is not known. The discovered patterns are highly redundant with often ten or more patterns expressing the same cause for misclassification. It is thus hard to get a full description of what goes wrong, it lacks the power of set mining approaches that evaluate patterns *together*. CORTANA filters more strongly, the discovered patterns are, however, still redundant. C-SALT finds a few, partially redundant patterns. Most patterns found by CLASSY consist of only one token, GRAB and RIPPER fail to retrieve meaningful results. In Appendix, Table 3, we provide statistics about the data and retrieved patterns, a full list of retrieved patterns for all methods can be found in the online material.

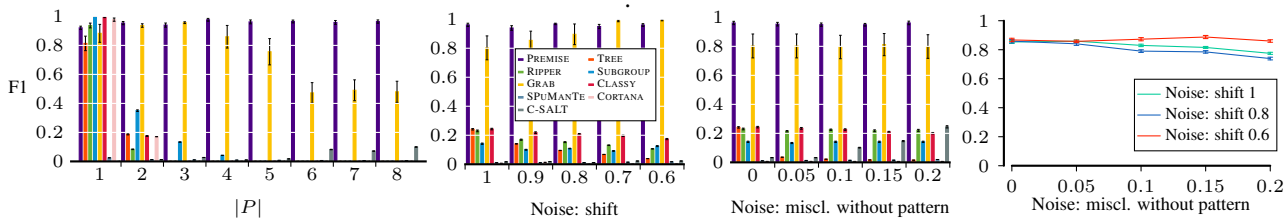
The patterns found by PREMISE (see Tab. 1, full results online), highlight the advantage of the richer pattern language, allowing to find patterns with related concepts such as $\bigwedge$(*what,* $\bigotimes$(*color, colors, colour*)). Generally, the patterns found by PREMISE highlight different types of wrongly answered questions, including counting questions, identification of objects and their colors, spatial reasoning, and higher reasoning tasks like reading signs. Furthermore, PREMISE retrieves both frequent patterns, such as $\bigwedge$(*how, many*) and rare patterns such as $\bigwedge$(*on, wall, hanging*).

PREMISE also discovers patterns that are biased towards correct classification, which can indicate issues with the dataset. For instance, $\bigwedge$(*who, took,* $\bigotimes$(*photo, picture, pic, photos, photograph*)), although a difficult question, is nearly always answered by "photographer" and thus easy to learn.



(a) Varying number of items (b) Varying label ratio (c) Varying label shift (d) Varying background noise

Figure 3. Synthetic data results. As competitors only recover fragments of patterns, the results are in terms of a soft F1 score, which also rewards the discovery of fragments, as defined in Appendix, Section A.7.



(a) Varying pattern length (b) Varying shift noise (c) Varying label noise (d) Complex clauses

Figure 4. Synthetic text data results. On synthetic text data, varying the number of items per pattern (a), the amount of *shift noise* (b), and the amount of *label noise* (c), we visualize the results in terms of F1 score with respect to the ground truth for existing methods and PREMISE. We additionally provide results of PREMISE on data containing patterns of mutual exclusive clauses for varying *shift noise* (d).

Another problematic question is indicated by the pattern ⩓(*clock, time*), where usually the answer is "UNK" – the unknown word token – due to the limited vocabulary of Visual7W. The pattern hence indicates a setting where the VQA classifier undeservedly gets a good score.

By adding additional information as items to each instance, it is possible to gain further insights. Appending the correct output to each instance, we observe for the question when the picture was taken two different trends. On the one hand, the discovered pattern ⩓(*when,* ⊗(*daytime, nighttime*)) is associated with correct classification, the pattern ⩓(*when,* ⊗(*evening, morning, afternoon, lunchtime*)), on the other hand, points towards misclassification. This is intuitively consistent as the answers "daytime" and "nighttime" are easier to choose based on a picture.

We observe in the discovered patterns that the Visual7W and LXMERT classifiers share certain issues, like the counting questions. However, no patterns regarding color or spatial position are retrieved. This seems to indicate that the more recent LXMERT classifier can handle these better.

6.3. Real Data: NER

A machine learning classifier might perform well during development, its performance when deployed "in the wild" however is often much worse. Understanding the difference is important for being able to improve the classifier. Here, we investigate the popular LSTM+CNN+CRF architecture (Ma & Hovy, 2016) for Named Entity Recognition (NER). The classifier is trained on the standard NER dataset CoNLL03, where it achieves a good performance (F1-score of 0.93). On OntoNotes, a dataset covering a wider range of topics, the performance drops to 0.61 F1 on the development set. We evaluate on this split of the data consisting of 16k sentences and 23k unique items.

Table 1. VQA example patterns for Visual7W found by PREMISE. More examples in Table 2 in the Appendix.

pattern	example from the dataset
UNK	how are the UNK covered
⩓(*how, many*)	how many elephants are there
⩓(*what,* ⊗(*color, colors, colour*))	what color is the bench
⩓(*on, top, of*)	what is on the top of the cake
⩓(*left, to*)	what can be seen to the left
⩓(*on, wall, hanging*)	what is hanging on the wall
⩓(*how, does, look*)	how does the woman look
⩓(*what, does,* ⊗(*say, like, think, know, want*))	what does the sign say

ANCHORS allows to obtain conjunctive patterns to explain NLP instances locally. It took, however, several days to analyze all misclassifications on modern GPU hardware due to the necessary, repeated queries to the NER classifier. ANCHORS finds 4.1k unique patterns, many of which are redundant, overly long, and specific. As expected from a local method, the patterns are highly specific and thus

identify problems of the model for particular instances rather than identifying the the general issues. PREMISE retrieves a concise set of 190 patterns. An example is ⩓(*-LRB-, -RRB-*) that indicates different preprocessing of the text. Patterns also indicate problems with differing labeling conventions, e.g. the patterns ⩓(*'s*) and ⩓(*Wall, Street*), which are handled differently for entities in OntoNotes. We can also isolate issues with OntoNotes alone, which contains bible excerpts that are not labeled at all. We discover this through patterns that describe this domain (*God*, *Jesus*, *Samuel*).

To empirically validate that the found patterns affect the classifier's performance, we select the top 50 patterns according to gain in MDL and for each pattern sample 5 sentences containing it uniformly at random from the OntoNotes training data. The CoNLL03 classifier is then fine-tuned on this data. Sampling and fine-tuning is repeated 20 times with different seeds. Using the pattern-guided data, the performance is improved to 0.67 mean F1 score (SE 0.003) compared to sampling fully at random where only a small improvement to 0.62 (SE 0.005) is achieved. This shows that the patterns discovered by PREMISE provide actionable insights into how a classifier can be improved.

7. Discussion

Experiments show that PREMISE provides concise and interpretable descriptions of labeled data. On synthetic data, we find that the state-of-the-art methods across different fields have severe difficulties finding the ground truth pattern set that describe the given labels. PREMISE is the only approach that is at the same time robust to noise, label imbalance, and easily scaling to thousands of items. When considering as label the information on whether a sample has been misclassified or not by a particular method, these approaches allow to find descriptions of which patterns in the data are correlated with misclassifications. For tasks like characterising misclassifications of NLP models, however, the labels are inherently imbalanced and the sets of items – in this case tokens – is large. To capture the structures of word associations, we further need a richer pattern language capturing mutual exclusiveness, which only PREMISE provides.

On two models for VQA, we set for characterising their misclassifications. While some of the competing methods retrieve reasonable explanations, these are highly redundant with several hundred or thousand patterns. Moreover, important concepts, such as patterns that are similar across related words or synonyms, are completely missed. PREMISE, on the other hand, discovers succinct sets of patterns that provide interesting characterizations, revealing that models struggle with counting, spatial orientation, reading, and identifies shortcomings in training data. For a popular NER classifier, we consider a model applied to text of a different source and characterize the resulting classification errors. Compared to the local explanation method ANCHORS, PREMISE retrieves a more succinct set of patterns in less time and we show that the obtained insights are actionable.

To show that the pattern sets are not only interpretable and give interesting insights, but also actionable, we analyze a popular classifier for Named Entity Recognition. In particular, we consider a model applied to text of a different source and characterize the resulting classification errors with PREMISE, and compare it with the recent local explanation method ANCHORS. While PREMISE is able to retrieve a pattern set swiftly in few hours on commodity hardware, ANCHORS requires several days on a modern GPU to deliver results. Inspecting the retrieved patterns confirms that also for NER models PREMISE is able to retrieve meaningful patterns explaining misclassification, while ANCHORS finds a large set of overly long and redundant patterns. Furthermore, as expected from a local method, the patterns describe instance-specific problems of the model rather than identifying the general issues that the model has.

8. Conclusion

We considered the problem of finding interpretable and non-redundant descriptions of a given label, and proposed to discover pattern sets to describe the labels based on the Minimum Description Length Principle. To solve this formulation in practice, we suggested an efficient bottom-up heuristic PREMISE. Our method showed to be the only approach that scales well to data typical in real world problem settings, while at the same time being robust to noise, and label imbalance. With these abilities, combined with a more expressive pattern language compared to state-of-the-art, capturing mutual exclusive relationship, PREMISE discovered succinct, informative, and actionable pattern sets that characterize misclassifications of NLP models in two challenging settings, which capture general problems of the model rather than instance specific (local) issues. It, hence, fills the gap of a robust approach to describe labels in terms of human-interpretable patterns in practice.

While our approach scales already to tens of thousands of features, it makes for engaging future work to scale it even further towards hundreds of thousands of features or to extend the work on characterizing misclassifications incorporating elements of the classifier itself, such as neuron activations (Fischer et al., 2021).

Acknowledgment

This work is partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 232722074 - SFB 1102.

References

Agrawal, R., Imielinksi, T., and Swami, A. Mining association rules between sets of items in large databases. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pp. 207–216. ACM, 1993.

Atzmueller, M. Subgroup discovery. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(1): 35–49, 2015.

Barakat, N. and Diederich, J. Eclectic rule-extraction from support vector machines. *International Journal of Computational Intelligence*, 2(1):59–62, 2005.

Budhathoki, K. and Vreeken, J. The difference and the norm – characterising similarities and differences between databases. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD)*. Springer, 2015.

Califf, M. E. and Mooney, R. J. Relational learning of pattern-match rules for information extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 328–334, 1999.

Chung, Y., Kraska, T., Polyzotis, N., Tae, K. H., and Whang, S. E. Automated data slicing for model validation: A big data - AI integration approach. *IEEE Transactions on Knowledge and Data Engineering*, 32(12):2284–2296, 2020.

Cohen, W. W. Fast effective rule induction. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 115–123, 1995.

Danilevsky, M., Qian, K., Aharonov, R., Katsis, Y., Kawas, B., and Sen, P. A survey of the state of explainable AI for natural language processing. In Wong, K., Knight, K., and Wu, H. (eds.), *AACL/IJCNLP 2020*, pp. 447–459, 2020.

Dong, G. and Li, J. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 43–52, 1999.

Duivesteijn, W. and Thaele, J. Understanding where your classifier does (not) work – the SCaPE model class for EMM. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pp. 809–814, 2014.

Fischer, J. and Vreeken, J. Sets of robust rules, and how to find them. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD)*. Springer, 2019.

Fischer, J. and Vreeken, J. Discovering succinct pattern sets expressing co-occurrence and mutual exclusivity. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 813–823, 2020.

Fischer, J., Olah, A., and Vreeken, J. What's in the box? Exploring the inner life of neural networks with robust rules. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 3352–3362, 2021.

Fisher, R. A. On the interpretation of χ 2 from contingency tables, and the calculation of p. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 85 (1):87–94, 1922.

Frosst, N. and Hinton, G. Distilling a neural network into a soft decision tree. In *Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML*, 2017.

García-Vico, A., Carmona, C., Martín, D., García-Borroto, M., and del Jesus, M. An overview of emerging pattern mining in supervised descriptive rule discovery: taxonomy, empirical study, trends, and prospects. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(1):e1231, 2018.

Gardner, M., Artzi, Y., Basmova, V., Berant, J., Bogin, B., Chen, S., Dasigi, P., Dua, D., Elazar, Y., Gottumukkala, A., Gupta, N., Hajishirzi, H., Ilharco, G., Khashabi, D., Lin, K., Liu, J., Liu, N. F., Mulcaire, P., Ning, Q., Singh, S., Smith, N. A., Subramanian, S., Tsarfaty, R., Wallace, E., Zhang, A., and Zhou, B. Evaluating models' local decision boundaries via contrast sets. In *Findings of the Association for Computational Linguistics: EMNLP*, pp. 1307–1323, 2020.

Grave, E., Bojanowski, P., Gupta, P., Joulin, A., and Mikolov, T. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation*, 2018.

Grünwald, P. *The Minimum Description Length Principle*. MIT Press, 2007.

Hämäläinen, W. Kingfisher: an efficient algorithm for searching for both positive and negative dependency rules with statistical significance measures. *Knowledge and Information Systems*, 32(2):383–414, 2012.

Hedderich, M. A., Adelani, D., Zhu, D., Alabi, J., Markus, U., and Klakow, D. Transfer learning and distant supervision for multilingual transformer models: A study on African languages. In *Findings of the Association for Computational Linguistics: EMNLP*, pp. 2580–2591, 2020.

Henelius, A., Puolamäki, K., Boström, H., Asker, L., and Papapetrou, P. A peek into the black box: Exploring classifiers by randomization. *Data Min. Knowl. Discov.*, 28(5–6):1503–1529, 2014.

Hess, S. and Morik, K. C-SALT: mining class-specific alterations in boolean matrix factorization. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD)*, pp. 547–563. Springer, 2017.

Kontkanen, P. and Myllymäki, P. A linear-time algorithm for computing the multinomial stochastic complexity. *Information Processing Letters*, 103(6):227–233, 2007.

Lakkaraju, H., Kamar, E., Caruana, R., and Leskovec, J. Interpretable & explorable approximations of black box models. *arXiv preprint arXiv:1707.01154*, 2017.

Lee, G., Kim, S., and Hwang, S. Qadiver: Interactive framework for diagnosing QA models. In *AAAI*, pp. 9861–9862, 2019.

Lemmerich, F. and Becker, M. pysubgroup: Easy-to-use subgroup discovery in python. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD)*, pp. 658–662, 2018.

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

Llinares-López, F., Sugiyama, M., Papaxanthos, L., and Borgwardt, K. Fast and memory-efficient significant pattern mining via permutation testing. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 725–734. ACM, 2015.

Ma, X. and Hovy, E. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *ACL*, pp. 1064–1074, 2016.

Meeng, M. and Knobbe, A. J. Flexible enrichment with cortana – software demo. In *Proceedings of Benelearn*, 2011.

Miettinen, P. On finding joint subspace Boolean matrix factorizations. In *Proceedings of the 12th SIAM International Conference on Data Mining (SDM), Anaheim, CA*, pp. 954–965. SIAM, 2012.

Miller, G. A. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

Novak, P. K., Lavrac, N., and Webb, G. I. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *Journal of Machine Learning Research*, 10:377–403, 2009.

Papaxanthos, L., Llinares-López, F., Bodenham, D. A., and Borgwardt, K. M. Finding significant combinations of features in the presence of categorical covariates. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2271–2279, 2016.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Pellegrina, L. and Vandin, F. Efficient mining of the most significant patterns with permutation testing. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 2070–2079, 2018.

Pellegrina, L., Riondato, M., and Vandin, F. SPuManTE: Significant pattern mining with unconditional testing. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 1528–1538. ACM, 2019.

Proença, H. M. and van Leeuwen, M. Interpretable multi-class classification by mdl-based rule lists. *Inf. Sci.*, 512: 1372–1393, 2020.

Ribeiro, M. T., Singh, S., and Guestrin, C. "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 1135–1144. ACM, 2016.

Ribeiro, M. T., Singh, S., and Guestrin, C. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1527–1535. AAAI Press, 2018.

Ribeiro, M. T., Wu, T., Guestrin, C., and Singh, S. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 4902–4912, 2020.

Riloff, E. and Wiebe, J. Learning extraction patterns for subjective expressions. In *Findings of the Association for Computational Linguistics: EMNLP*, 2003.

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

Rissanen, J. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, 11(2):416–431, 1983.

Rondeau, M.-A. and Hazen, T. J. Systematic error analysis of the Stanford question answering dataset. In *Workshop on Machine Reading for Question Answering*, pp. 12–20, 2018.

Tan, H. and Bansal, M. LXMERT: Learning cross-modality encoder representations from transformers. In *Findings of the Association for Computational Linguistics: EMNLP*, pp. 5100–5111, 2019.

van Leeuwen, M. and Knobbe, A. J. Diverse subgroup set discovery. *Data Mining and Knowledge Discovery*, 25 (2):208–242, 2012.

Vimieiro, R. *Mining disjunctive patterns in biomedical data sets*. PhD thesis, The University of Newcastle, Australia, 2012.

Webb, G. I. Discovering significant patterns. *Machine Learning*, 68(1):1–33, 2007.

Wrobel, S. An algorithm for multi-relational discovery of subgroups. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 78–87. Springer, 1997.

Wu, T., Ribeiro, M. T., Heer, J., and Weld, D. Errudite: Scalable, reproducible, and testable error analysis. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 747–763, 2019.

Zhu, Y., Groth, O., Bernstein, M., and Fei-Fei, L. Visual7W: Grounded Question Answering in Images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

A. Appendix

A.1. Pseudocode

We, here, provide the pseudocode for the function to create candidate patterns (Alg. 2).

Algorithm 2 createCandidates

Input: D, patterns $\mathcal{P}$ in current M, max neighbour distance K
Output: C, a set of candidate patterns $\mathcal{P}$
// Define $nb(I, 0) = I$ for simplicity
$C \leftarrow \{\}$
// Single item and its neighbours
for $I \in \mathcal{I}$ **do**
  $A \leftarrow \{\}$
  for $k \in \{0, \ldots, K\}$ **do**
    $A \leftarrow A \cup \{nb(I, k)\}$
    $C \leftarrow C \cup \{\otimes(A)\}$
  end for
end for
// Pairs of items and their neighbours
for $(I_1, I_2) \in \mathcal{I} \times \mathcal{I}$ **do**
  $A_1 \leftarrow \{\}$
  for $k_1 \in \{0, \ldots, K\}$ **do**
    $A_1 \leftarrow A_1 \cup \{nb(I_1, k_1)\}$
    $A_2 \leftarrow \{\}$
    for $k_2 \in \{0, \ldots, K\}$ **do**
      $A_2 \leftarrow A_2 \cup \{nb(I_2, k_2)\}$
      $C \leftarrow C \cup \{\owedge(\otimes(A_1), \otimes(A_2))\}$
    end for
  end for
end for
// Pattern + item and its neighbours
for P in $\mathcal{P}$ **do**
  for $I \in \mathcal{I}$ **do**
    $A \leftarrow \{\}$
    for $k \in \{0, \ldots, K\}$ **do**
      $A \leftarrow A \cup \{nb(I, k)\}$
      $C \leftarrow C \cup \{\owedge(\gamma(P) \cup \{A\})\}$
    end for
  end for
end for
// Pattern + Pattern
for $(P_1, P_2) \in \mathcal{P} \times \mathcal{P}$ **do**
  $C \leftarrow C \cup \{\owedge(\gamma(P_1) \cup \gamma(P_2)\}$
end for
// see Sections 4 and 5 for filter criteria
$C \leftarrow \texttt{Filter}(C)$

Table 2. VQA example patterns. Our method discovers meaningful and easily interpretable patterns. For the LXMERT dataset, we show a subset of the patterns highlighting different reasons for misclassification along with examples from the corresponding datasets. The full list of retrieved patterns for all methods is given in the Supplementary Material.

pattern	example
$\owedge$(*How, many*)	How many kites are flying?
$\owedge$(*hanging, from*)	What is hanging from a hook?
$\owedge$($\otimes$(*kind, sort*), of)	What kind of birds are these?
$\owedge$($\otimes$(*would, could, might, can*), *you*)	How would you describe the decor?
$\owedge$(*name, of*)	What is the name of this restaurant?
number	What is the pitchers number?
$\otimes$(*letter, letters*)	What letter appears on the box?
$\owedge$(*How, much,* $\otimes$(*cost,costs*))	How much does the fruit cost?

A.2. Proof: Order of Items

Here, we provide a proof that the codelength is independent on the order of items in mutual exclusive clauses. The proof closely follows that of Fischer & Vreeken (Fischer & Vreeken, 2020).

Given a clause $cl = \otimes(i, j, k)$ with corresponding margins n_i, n_j, n_k, it does not matter in which order we transmit where the items hold. We show that we can flip the item order without changing the cost. Assume a new order $P = \otimes(k, i, j)$, then we show

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

With the definition of the binomial using factorials and standard math, adding new terms that add up to 0, we show

Table 3. VQA data statistics. For the two VQA classifiers, we provide general statistics about data dimensions, and for each method the number of discovered patterns ($k = |P|$) or if applicable number of patterns explaining misclassification ($k^- = |P^-|$), respectively correct classification ($k^+ = |P^+|$) and the average pattern length $\overline{|p|}$.

			PREMISE			TREE		RIPP.		SUBGR.		SPUM.		CLAS.		GRAB		CORTA.		CSAL.	
Dataset	$\lvert\mathcal{I}\rvert$	$\lvert D\rvert$	k^-	k^+	$\overline{\lvert p\rvert}$	k	$\overline{\lvert p\rvert}$	k	$\overline{\lvert p\rvert}$	k	$\overline{\lvert p\rvert}$	k	$\overline{\lvert p\rvert}$	k	$\overline{\lvert p\rvert}$	k	$\overline{\lvert p\rvert}$	k	$\overline{\lvert p\rvert}$	k	$\overline{\lvert p\rvert}$
Visual7W	2429	28032	29	26	3.4	4309	3.6	0	0.0	100	2.3	575	2.9	19	1.3	1	1	15	2.3	10	3.9
LXMERT	5351	25994	41	34	2.7	3371	2.7	3	3.0	100	2.5	951	3.9	36	1.3	1	1	2	3.0	11	4.2

that the above equation hold.

$$
\begin{aligned}
&\log\frac{n!}{(n-n_i)!n_i!} + \log\frac{(n-n_i)!}{(n-n_i-n_j)!n_j!} \\
&\quad + \log\frac{(n-n_i-n_j)!}{(n-n_i-n_j-n_k)!n_k!} \\
&= \log(n!) - \cancel{\log((n-n_i)!)} - \log(n_i!) + \cancel{\log((n-n_i)!)} \\
&\quad - \cancel{\log((n-n_i-n_j)!)} - \log(n_j!) + \cancel{\log((n-n_i-n_j)!)} \\
&\quad - \log((n-n_i-n_j-n_k)!) - \log(n_k!) \\
&\quad \underbrace{+\log((n-n_k)!) - \log((n-n_k)!)}_{=0} \\
&\quad \underbrace{+\log((n-n_i-n_k)!) - \log((n-n_i-n_k)!)}_{=0} \\
&= \log\frac{n!}{(n-n_k)!n_k!} + \log\frac{(n-n_k)!}{(n-n_i-n_k)!n_i!} \\
&\quad + \log\frac{(n-n_i-n_k)!}{(n-n_i-n_j-n_k)!n_j!}.
\end{aligned}
$$

Other permutations and larger clauses follow the same reasoning.

A.3. Size of the Model Space

We here briefly describe the derivation for the size of the model space. In particular, the size of the model space is

$$|\mathcal{M}| = 2^{\sum_{i=1}^{|\mathcal{I}|}\binom{|\mathcal{I}|}{i}\times\sum_{j=1}^{i}\left\{{i \atop j}\right\}},$$

where the first term in the summation specifies the number of possible item combinations in a pattern of length i, the second term counts the number of possible ways to separate them into j different clauses via the Stirling number of the second kind and the exponent is introduced as a model M consists of arbitrary combinations of patterns. The MDL score for such complex model classes does not lend itself for easy-to-exploit structure such as monotonicity. Hence, we resort to an efficient bottom-up search heuristic for discovering good models which we introduce in the next section.

A.4. Mutual Exclusivity and Word Neighbors

For the clauses of mutually exclusive items, we are interested in finding words that are synonyms or that reflect similar concepts, such as $\otimes$(*color, colour*) or $\otimes$(*could, can*). Research in NLP has proposed various techniques for identifying such pairs including manually created ontologies such as WordNet (Miller, 1995) or word embeddings that are learned through co-occurrences in text and map words to vector representations. This information about related words can be used to guide the search for mutually exclusive patterns. Using such pretrained embeddings rather than deriving them from the given input data has the advantage that we are independent of the size of the input data set, and receive reliable embeddings, which were trained on very large, domain independent text corpora.

While our approach is independent of the specific method, we have chosen FastText word embeddings trained on CommonCrawl and Wikipedia (Grave et al., 2018). In contrast to word ontologies, word embeddings have a broader vocabulary coverage. They also do not impose strict restrictions such as a particular definition of synonyms and instead reflect relatedness concepts learned from the text. FastText embeddings have the additional benefit that they use subword information, removing the issue of out-of-vocabulary words. The word embeddings are independent of the machine learning classifier we study. As measure of relatedness m between two items I_1, I_2, we use cosine similarity, i.e. $m = cos(emb(I_1), emb(I_2))$ where emb is the mapping between an item/word and its vector representation. We define $nb(I, k)$ as the $I' \in \mathcal{I}$ for which $m(I, I')$ is the k-highest. Examples for words and their neighbours in FastText embeddings are given in Table 4.

Based on the information of the embedding, we derive $\otimes$-clauses. For each item I, we explore mutual exclusivity in its $1 \dots K$ closest neighbors, i.e. from $\otimes(I, nb(I, 1))$ until $\otimes(I, nb(I, 1), \dots, nb(I, K))$ where K is the maximum neighborhood size. For that, we adapt the `createCandidates` algorithm from Section 5.1 so that whenever we consider merging with an item I, we also consider merging with the $\otimes$-clauses containing additionally the $1, 2, \dots K$ closest neighbours (see Alg. 2).

Since not all words have K neighbors that represent similar words, we additionally filter neighbourhoods such that $\frac{\bigcap_I \sigma_I(D)}{\bigcup_I \sigma_I(D)} < a$ and $m(I, nb(I, k)) > b_k$ for all items I in the clause, i.e. we require that their transactions barely overlap

Word	5-nearest neighborhood
photo	photograph, photos, picture, pic, pictures
color	colour, colors, purple, colored, gray
can	could, will, may, might, able
say	know, think, tell, mean, want

Table 4. Words and their nearest neighbors on *Visual7W*.

(mutual exclusivity), and that their embeddings are reasonably close. In all experiments we set $K = 5$, $a = 0.05$ and b_k to the 3rd quartile of $\{m(I, nb(I, k)) \mid I \in \mathcal{I}\}$.

In the general case for arbitrary labeled data, we could follow the proposal of Fischer & Vreeken (2020) to search for potential XOR structure, which however would lead to a much increased search space and hence computational costs, without any benefits for the specific applications.

A.5. Complexity

While it is common to consider the complexity in terms of the size of the input, the bound it would give – which is exponential in the number of items as discussed in the theory section – is neither helpful nor tight considering the discovery of small models. we thus analyze the complexity of PREMISE in terms of the size of the model.

Consider PREMISE finds k conjunctive patterns of maximum length l for a dataset with m items. Since in every round either a new singleton or pair is generated that belongs to one of the k final patterns, or two existing patterns are merged, the algorithm runs $O(k\,l)$ rounds. In each round, the dominating factor is the candidate generation, out of which there are $O(m)$ potential singletons, $O(m^2)$ pairs, and at maximum $O(k\,l)$ pattern merges, corresponding to the case that all parts of the final patterns exist as singleton patterns in the current round. Hence, we get a worst case time complexity of $O(k\,l\,(k\,l + m^2))$.

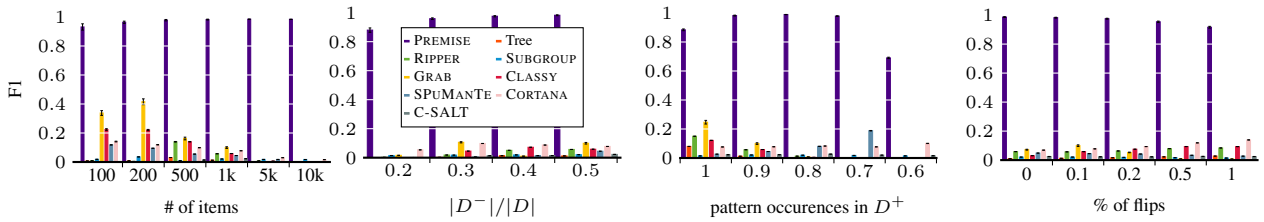
For clauses containing mutual exclusivity, for all practical applications we consider XOR statements of the c closest words in a given embedding, where c is a small constant. We hence consider $O(m\,c)$ single XOR clauses, $O((m\,c)^2)$ pairs, and at maximum $O(k\,l)$ pattern merges, where again this corresponds to the case that all parts of the final patterns exist as singleton patterns in the current round. Hence we get a worst case time complexity of $O(k\,l\,(k\,l + (m\,c)^2))$. For the general case, when searching for arbitrary AND and XOR combinations, we refer to Fischer & Vreeken (Fischer & Vreeken, 2020).

A.6. Experimental Details

Experiments were performed on an Intel i7-7700 machine with 31GB RAM running Linux. For the single-threaded C++ implementation of PREMISE, all synthetic data experiments finished within minutes for the moderately sized data sets, and within hours for the larger datasets with 5k and 10k items. On the VQA datasets PREMISE finished within 20 minutes and on the NER data within 4 hours.

For TREE, patterns are exracted from a decision tree trained on the misclassification data. Each of the tree's inner nodes is a binary decision regarding the presence of an item and a pattern is the conjunctive path from the tree's root to one of its leafs. The model is trained with Gini impurity as decision criterion in the implementation from scikit-learn (Pedregosa et al., 2011). For SUBGROUP-DISCOVERY, the PySubgroup library is used (Lemmerich & Becker, 2018) with depth-first search and StandardQF as quality function. The size of the result set and the maximum depth are set to the ground truth for the synthetic data. On the synthetic data, it hence has an advantage over all other approaches which would not hold in a real-world scenario. For the VQA datasets, the values are set to 100 and 5 respectively. SPUMANTE is used with the authors' default parameters, setting its sample size to the dataset size. For GRAB we use the publicly available implementation by the authors, which we tailored for the task at hand by restricting the possible rule-heads to the labels only, but allowing tails over all other items. CORTANA (Meeng & Knobbe, 2011) is a subgroup discovery tool. It is used by Duivesteijn & Thaele (2014) where two numeric labels are expected, one for ground truth and one for prediction probability. We, therefore, split the misclassification label into two labels that disagree if an instance is misclassified. As quality measure, we use negative r and we follow the authors approach of only considering subgroups that cover 1% of the data to prevent overfitting. The maximum depth is set to ground truth for the synthetic data and to 5 for VQA. The beam width is kept to 100 and the quality threshold to 0.2 following the default settings. For CLASSY we used the publicly available implementation by the authors as used in the original publication. Minimum support is set to 1 and maximum rule length to the ground truth for the synthetic data and 5 for the VQA datasets. For C-SALT (Hess & Morik, 2017), we use the authors' implementation and the parameters of their test script, i.e. rank increment 10, 10k iterations and threshold 0.05. We set the maximum rank to ground truth for the synthetic data and to 100 on the VQA data.

For Visual7W and LXMERT, we use the published, pretrained models by the corresponding authors. For LXMERT, the minival version of the development set is used. For the LSTM+CNN+CRF classifier (Ma & Hovy, 2016) for NER, we follow the specific set-up from Hedderich et al. (2020) with English FastText embeddings. OntoNotes was split and preprocessed using the script from `https://github.com/yuchenlin/OntoNotes-5.0-NER-BIO`. The fine-tuning data consists of 240 instances/sentences as two



(a) Varying number of items (b) Varying label ratio (c) Varying label shift (d) Varying background noise

Figure 5. Synthetic data results (F1 score). We visualize results on synthetic data with varying number of items (a), label ratio (b), label shift (c) and amount of background noise (d). The results are in terms of F1 score with respect to the ground truth.

patterns did not match any training data. Fine-tuning on the additional data is performed for 30 epochs. As labels, the intersection between CoNLL03 and OntoNotes is used (PER, LOC, ORG) in the BIO2 format.

A.7. F1 Metric

A standard metric to evaluate success of a model is the F1 score – the harmonic mean between precision and recall – which for discovered pattern set P_d and ground truth pattern set P_g is defined as $\text{F1}(P_d, P_g) = |P_d \cap P_g| / \left(|P_d \cap P_g| + \frac{1}{2}|P_d \Delta P_g|\right)$, where Δ is the symmetric difference between two sets. As competitors only recover fragments of patterns and hence they obtain very low F1 scores, we instead report a soft F1 score that rewards also fragments. We define it as harmonic mean between a soft precision and a soft recall:

$$\text{SoftPrec}(P_d, P_g) = \sum_{p_d \in P_d} \operatorname*{argmax}_{p_g \in P_g} \frac{|p_d \cap p_g|}{|p_g|} ,$$

$$\text{SoftRec}(P_d, P_g) = \sum_{p_g \in P_g} \operatorname*{argmax}_{p_d \in P_d} \frac{|p_d \cap p_g|}{|p_g|} ,$$

$$\text{F1}(P_d, P_g) = \frac{2 * \text{SoftPrec} * \text{SoftRec}}{\text{SoftPrec} + \text{SoftRec}} .$$

Results with the original F1 score are given in Figure 5.

A.8. Synthetic Text Data Experiments

To obtain a synthetic data set with similar item/token distributions as natural language text, we derive transactions/instances from the around 3.4k sentences in the development set of the PennTreebank Corpus. In particular, we draw 12 distinct patterns, for each pattern choosing items from the vocabulary tokens at random. To ensure that we introduce only new patterns into the data, we verify that none of the items in the patterns co-occur in the original data. We then insert each pattern into a random subset of the PennTreebank instances, where the number of instances to be covered is drawn from a normal $\mathcal{N}(150, 20)$. The data contains 6k unique items. To evaluate settings typical for classification, we then vary two types of noise. *Shift noise* indicates the percentage of instances with a pattern that are actually labeled as misclassifications, lower values mean that the model is still able to predict correctly in some of the instances – e.g. because a network leverages additional information in the data. The second type of noise is labeling instances as misclassification although there is no pattern occurrence – i.e. non-systematic errors – which we refer to as *label noise*. For all samples with pattern occurences, we label a fraction of those as misclassification according to the *shift noise*, and then introduce *label noise*.

Experimental setups We generate four different sets of experiments. In the first set, we introduce conjunctive patterns varying pattern length of the introduced patterns between 1 and 8 without noise. In the second set of experiments we vary the amount of *shift noise*, introducing shifts of $\{0.6, 0.7, 0.8, 0.9, 1\}$, and choosing pattern length uniformly in 1 to 5. In the third set we instead change the amount *label noise*, varying in $\{0, 0.05, 0.1, 0.15, 0.2\}$. In the fourth set of experiments, we introduce patterns consisting of conjunctions of mutual exclusive itemsets. The number of clauses per pattern and the number of items for each clause is chosen uniformly at random between 1 and 5. A pattern is only added to an instance if this would not break the mutual exclusivity assumptions of all patterns. For the word neighborhoods, items in the same clause obtain embeddings located around a randomly chosen centroid. All other items obtain random embeddings. We repeat all experiments 10 times and report the F1 score – the harmonic mean between precision and recall – as average across repetitions.

Results For the first experiment set (Fig. 4a) of varying pattern length, we observe that SUBGROUP-DISCOVERY and CORTANA are able to retrieve short patterns well, failing however to discover any larger patterns, instead retrieving large sets of redundant patterns. Decision trees perform similarly due to overfitting, finding a plethora of highly redundant patterns. SPUMANTE, which although based on statistical testing, consistently finds thousands of redundant patterns, performing worst of all in this regard. The rule set miner GRAB recovers small patterns well, it performs

however much poorer in retrieving patterns of larger size. PREMISE is the only approach to consistently recover the ground truth in all data sets.

For both noise experiments, visualized in Fig. 4b and 4c, the tree based method completely breaks down already for moderate amounts of noise. SUBGROUP-DISCOVERY and SPUMANTE both perform consistently bad with F1 scores below .2. Out of the existing approaches, only GRAB is able to recover the ground truth well. PREMISE outperforms all existing methods in each of our noise experiments, achieving consistently high F1 scores beyond .92.

As ablation, we ran PREMISE without the noise filter introduced in Section 5.2. A score around $0.3 - 0.4$ is achieved, which is better than most baselines, but much lower than the score achieved with the complete PREMISE method.

Since most baselines do not support discovering mutual exclusivity or proved to fail in the more simple setup of conjunctions, we only evaluate our proposed method on the fourth set of experiments. We observe that PREMISE is still able to retrieve patterns even in this challenging setup of complex clauses, with F1 scores close to .9, and is able to discover clauses in the presence of noise (Fig. 4d).