Characterizing Misclassifications of Deep NLP Models

Anonymous Preprint

ABSTRACT

Understanding the reasons for misclassification is a critical point to improve a black-box classifier's performance. We propose a method to characterize these classification errors while considering the challenges that NLP applications pose, such as sparse and high dimensional discrete input spaces. Our approach discovers patterns over the input of a model that strongly correlate with the correctness of the classification. This allows identifying the systematic errors made by the models. We formalize the problem in terms of the Minimum Description Length principle to obtain non-redundant and easily interpretable results, and we propose the Premise algorithm to find good patterns in practice. The discovered patterns allow the user to take action and improve the model, e.g. through changes to the training data or model definition. On synthetic data and two real-world NLP tasks, we show that Premise performs well in practice. For two Visual Question Answering classifiers, we discover that they struggle with aspects like counting, location and reading, and for a Named Entity Recognition model, we leverage the found patterns to improve the F1 performance by almost 10% through targeted fine-tuning.

CCS CONCEPTS

• **Information systems → Data mining**.

KEYWORDS

pattern mining, explainability, misclassification, MDL

1 INTRODUCTION

State-of-the-art deep learning achieves human-like performance on many computer vision and natural language processing tasks. However, just as much as 'to err is human', these models make errors too. Some of the errors they make are relatively benign as they are due to noise in the process we're trying to model, i.e. they are random. As there is no structure to these errors, adding more training data will not or only barely change the model's overall performance. Systematic errors, on the other hand, such as those due to misspecification of the model or bias in the training data, are much more serious as they lead to mistreatment of entire groups of instances. At the same time, these errors are nice in the sense that once we are aware of them, we can actively intervene and either adapt the model, or augment the training data in a targeted fashion to improve the overall performance. Before we can do so, we first need to identify whether a model makes systematic errors, and, how to characterize those errors in easily understandable terms. How to do so is precisely what we answer in this paper.

We study the problem of characterizing the errors of a black box classification model, with a specific focus on the setting where we have sparse high dimensional discrete input spaces, such as found in natural language processing. In particular, we propose to discover patterns over the input of these models, such that these strongly correlate with the correctness of answers. We are, for example, interested in discovering that state-of-the-art visual question answering (VQA) models significantly often incorrectly answer questions involving the words *how* and *many*, signifying that they are bad at counting. We are not just interested in conjunctive patterns, but also want to handle synonyms, and therefore consider a rich pattern language of noise-robust conjunctions and mutual exclusivity. This allows us, for the same VQA to discover the misclassification pattern ⦲(*what,* ⊗(*color, colors, colours*)), which is to say, it is also bad at answering questions about colors in the image.

To ensure that the results we discover are useful, we aim for small and non-redundant sets of highly informative patterns. We formalize this problem in information-theoretic terms using the Minimum Description Length (MDL) principle, by which we define the set of patterns that best characterizes the classification behaviour of the model as the one that best compresses its errors and successes. The search space over such pattern sets is twice exponential, however, and as it does not exhibit any easy-to-exploit structure that can steer us to the optimal result, we therefore propose Premise, an efficient bottom-up heuristic to discover high quality sets of patterns that explain the conditions, the *premises*, under which the model under inspection tends to make classification errors.

We evaluate Premise on synthetic data, as well as on two real-world NLP tasks. The experiments show that unlike the state of the art, Premise is robust against noise and gives clear insights into the biases of two VQA classifiers. Its results clearly show that these models have issues with different aspects including counting, relative location and higher reasoning tasks like reading. For Named Entity Recognition (NER) we show that we can use the patterns that Premise discovers to actively intervene and fine-tune the classifier with additional training data to improve its F1-score performance by 9.8% from 0.61 to 0.67.

The remainder of this paper is organized as usual. We provide additional detail for reproducibility in the supplementary, and make our code and data publicly available for research purposes.[1]

2 RELATED WORK

We propose to mine those patterns that best characterize the classification behaviour of the model under consideration, and hence work in pattern mining is strongly related. Frequent pattern mining was first proposed by Agrawal and Srikant [1], after which research focused on more efficient algorithms [15, 46] and succinct representations [3, 4, 15, 27]. By measuring interestingness per pattern,

[1] https://bit.ly/39TAl7a

and asking for all patterns that meet an interestingness threshold, the results of traditional approaches are extremely large and redundant [42]. Pattern set mining circumvents this by asking for a small set of non-redundant patterns that together generalize the data well [23, 25, 37]. All of the above are unsupervised in nature, i.e. they do not take label information into account and are hence not directly applicable to the task at hand.

Supervised pattern mining is known under different names, of which subgroup discovery [44] and emerging pattern mining [6] are the most well-known [28]. The task is to discover those patterns that correlate strongly with the class labels. Emerging pattern mining returns all patterns that meet a user-specified 'growth' threshold, and hence suffers from the same problems as frequent pattern mining. Subgroup discovery instead 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 [40]. For recent surveys we refer to Atzmueller [2], García-Vico et al. [10].

Statistical pattern mining avoids discovery of spurious results by reporting only patterns that correlate *significantly* to a class label [21, 29, 31]. While it is (relatively) easy to test one pattern for significance, in practice we have to evaluate many millions of candidates, and hence multiple hypothesis testing becomes a serious problem: these methods tend to discover millions of 'significant' patterns even from small data, making the results hard to use.

Rule mining aims to discover rules of the form $X \rightarrow Y$ [1, 14], and could hence also be used to explain misclassifications. Like the methods above, most existing methods again evaluate patterns individually, and therewith run into similar problems as frequent itemset mining, discovering millions of rules even if the data is pure noise. Grab [7] combines the idea of pattern set mining with rule mining, with the goal of discovering a small set of rules that together summarize the data well. We compare to Grab in the experiments. We can already reveal that it does not work well in our setting: association rules reflect that the consequence is likely to co-occur with the antecedent, and unlikely to occur otherwise. As a result, rule mining techniques such as Grab fail to pick up subtle patterns, e.g. one that occurs in 70% of the misclassified instances, and 40% of the correctly classified instances.

Besides the problem of redundancy and being able to detect subtle patterns, all above approaches are restricted to conjuctions. Natural language data is sparse, and contains many synonyms and language variations. We therefore build upon and extend the pattern language Fischer et al. [8] recently proposed for the unsupervised setting, as it allows us to express noise-robust conjunctions and mutual exclusive relationships between words.

There also exists work in NLP research for identifying the issues leading to classification errors. Challenge and contrast sets are test sets to evaluate models on difficult instances or concepts unseen in the training distribution [11, 33]. They require manual creation and some of them do not directly indicate reasons for the misclassification, hence are more suited for benchmarking. Approaches exist to verify user-provided hypotheses for misclassification causes [18, 36, 45]. SliceFinder [5] is a method to find small sets of slices that describe where the model loss is high. They focus on numerical and categorical features and—as we confirm in our experiments—their method does not scale to the large input spaces of text data.

Here, we propose to mine sets of patterns that provide a succinct and descriptive representation of the misclassified instances. We formulate the problem based on the Minimum Description Length principle and propose an efficient heuristic to discover pattern sets that describe misclassification in practice. We compare our approach to the state-of-the-art in rule mining, subgroup discovery, emerging pattern mining, and statistical pattern mining.

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 datasets of sequences X of length $|X|$ over an alphabet $\mathcal{I}$, i.e. $X \in \mathcal{I}^{|X|}$, along with a classification label y for each sequence. Additionally, we consider indicator $\ell_f(X, y)$ that specifies whether machine learning model f classifies a sample correctly or not,

$$\ell_f(X, y) = \begin{cases} l_- & \text{if } f(X) \neq y \text{ (misclassification),} \\ l_+ & \text{otherwise (correct classification).} \end{cases}$$

We transform the sequence data into a binary transaction database D over the set of items $\mathcal{I}$. Additionally, we define a mapping of transactions to corresponding misclassification labels $\lambda(t) \in \{l_-, l_+\}$, and define the partition of D by label as $D^+ = \{t \in D \mid \lambda(t) = l_+\}$, respectively $D^- = \{t \in D \mid \lambda(t) = l_-\}$. 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 given as $\pi_X(D) = \{t \cap X \mid t \in D\}$.

We use the definition by Fischer and Vreeken [8], hence for a logical condition c, we consider a selection operator

$$\sigma_c(D) = \{t \in D \mid c(t) \equiv \top\}.$$

For an item $I \in \mathcal{I}$, it holds that $[c_I(t) \equiv \top \leftrightarrow I \in t]$. The k-ary AND operator $\bigotimes(c_1, \ldots, c_k)$ describes patterns of co-occurrence and holds iff all its conditions hold, i.e.

$$\left(\bigotimes_{c_1,\ldots,c_k}(t) \equiv \top\right) \leftrightarrow \left(\forall_{i=1}^{k} c_i(t) \equiv \top\right).$$

Similarly, the k-ary XOR operator $\otimes$ describes patterns of mutual exclusivity, where

$$\left(\otimes_{c_1,\ldots,c_k}(t) \equiv \top\right) \leftrightarrow \left(\exists_{i \in \{1,\ldots,k\}} c_i(t) \equiv \top \land \forall_{j \neq i} c_j(t) \equiv \bot\right).$$

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. $\bigotimes(\otimes_{c_1,\ldots,c_k}, \ldots, \otimes_{c'_1,\ldots,c'_k})(t)$. An XOR operation is called clause, $\gamma(c)$ lists all clauses in conjunctive condition c.

To simplify notation, we drop t where it is obvious from context and write I for conditions on a single item $c(I)$. In the text, condition and pattern are used interchangeably.

3.2 Minimum Description Length

The Minimum Description Length (MDL) principle [34] is an approximation of Kolmogorov complexity [20] 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 [13], 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. When applying MDL, we are only interested in the codelengths and not the actual code. Codelength is measured in bits, hence all log operations are base 2 and we define $0 \log 0 = 0$.

4 THEORY

To discover patterns characterizing classification error using MDL, we here introduce the class of models $\mathcal{M}$ and correpsonding codelength functions that yield the number of bits required to encode a model, respectively the number of bits needed to encode data given a model. Before we define these formally, we give the intuition behind the model class $\mathcal{M}$ and why it naturally captures our problem.

4.1 The Problem, informally

Given a dataset of binarized input and prediction error observed from a model, we aim to find a set of patterns that together identify the systematic errors made by the model. Focusing on NLP tasks, we are specifically interested in patterns that capture combinations of words that describe a label, e.g. $\owedge$(*how, many*) occuring predominantly when a misclassifcation happens, but are also capable of expressing synonyms or different writing styles. Those we express as mutual exclusive patterns, e.g. $\otimes$(*color, colour*). The pattern language we use here is a combination of the two, namely conjunctions of mutual exclusive clauses such as $\owedge$(*what*, $\otimes$(*color, colour*)). We provide an example in Figure 1.

Thus, we define a model $M \in \mathcal{M}$ as a set of patterns $\mathcal{P}$ containing all patterns that help to characterise the labels. Additionally, M contains all singleton words $I \in \mathcal{I}$, describing the entire data D label unspecific, and hence serve as a baseline assuming that there is no structural error that led to misclassification. Furthermore, these singletons ensure that we can always encode any data over the set of words using our model. Whenever there is a structural error in the labels that can be explained by a pattern, we transmit the data using that pattern for correctly, respectively incorrectly, classified data separately. This allows us to more succinctly transmit where this pattern holds.

Consider the example in Figure 1, we would first send $\owedge(A, \otimes(B, C))$ occurrences in D^+, and then its occurrences in D^-. Thus, we identify where A, C, and D hold at once, and we leverage the fact that $\owedge(A, \otimes(B, C))$ 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. Due to the one-to-one correspondance between codelength and probabilities, we hence transmit more efficiently by sending the pattern separately for D^+ and D^- if there is a large deviation between these two probabilities.

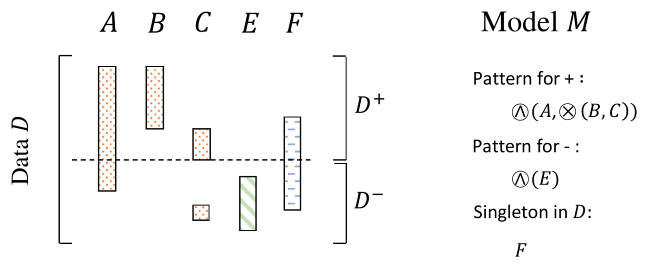


Figure 1: *Example database and model.* A toy database D over a set of items, separated by misclassification labels into D^+ and D^-, is given on the left. The corresponding model M containing patterns that explain misclassification (D^-) and correct classification (D^+) is given on the right.

Coming back to the example, F however occurs similarly often in both labels — 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 to transmit. 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 misclassification in easily understandable terms. We are hence after the model $M^* \in \mathcal{M}$ that minimizes the cost of transmitting the data and the model.

4.2 MDL for Misclassification Patterns

We now formalize the above intuition using an MDL score to identify the best pattern set $M^* \in \mathcal{M}$ to describe classification errors in a given dataset. We will first detail how to compute the encoding costs for the data given the model and then for the model itself.

Cost of Data Given Model. We will 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 [20]. For the data cost, we thus obtain

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

Taking into account the split of D along the label, $D^- = \{t \in D \mid \lambda(t) = l_-\}$ and D^+ correspondingly, we encode I separately:

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

Due to the properties of the binomial, this code length will be shorter than the baseline code if there is a difference between the unconditional probability – i.e. frequency in D – of I and the conditional probability of I conditioned on the label – i.e. frequency in D^- respectively D^+ . It models the property that we are interested in; a pattern that is specific to an (in)correct classification. It is straightforward to extend to patterns of co-occurring items $P =$

$\owedge(X_1, \ldots, X_k)$ by selecting on transaction where the pattern holds

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

There might be transactions where individual items of P are present, but not the full pattern holds. To ensure a lossless encoding, the singleton code $L(\pi_I(D) \mid I)$ is adapted to cover all item occurrences left unexplained after transmitting all patterns. Hence, we get

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

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. That we do one by one, as when we know that the pattern holds and A is present, B cannot be present too. Hence, with each item of the clause that is transmitted, the codelength needed to identify transactions of the remaining items is reduced. More formally, the codelength for a pattern P of conjunctions of clauses is given as

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

assuming an arbitrary order on $\mathcal{I}$. With clauses of only length 1 we arrive at a simple conjunctive pattern, where the second term evaluates to 0 and thus 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 a proof for the case of 3 items in App. A.2, the case for an arbitrary number of l items follows the same reasoning.

This concludes the definition of codelength functions for transmitting the data, and we can define the overall cost of transmitting the data D given a model M as

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

Cost of the Model. To transmit the model M of patterns $\mathcal{P}$, first, the number of patterns $|\mathcal{P}|$ are transmitted using the MDL-optimal code for integers $L_{\mathbb{N}}(|\mathcal{P}|)$. It is defined as $L_{\mathbb{N}}(n) = \log^* n + logc_0$ with $\log^* n = \log n + \log\log n + \ldots$ and c_0 being a constant so that $L_{\mathbb{N}}(n)$ satisfies the Krafft-inequality [35]. Then, for each pattern P, the number of clauses is transmitted via $L_{\mathbb{N}}(|\gamma(P)|)$. For each such clause the items it contains are transmitted 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 and Myllymäki [17]. Lastly, we transmit the parametric complexities of all binomials used in the data encoding.

Combining the above, the overall cost of the model is

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

by which we have a lossless MDL score.

The Problem. We can now formally state the problem.

Minimal Misclassification Description Problem *Given data D over $\mathcal{I}$ and a split into D^- and D^+, find the best model M our of all models $\mathcal{M}$ that minimizes the codelength $L(M, D)$.*

Solving this problem through enumeration of all possible models is computationally infeasible as the model space is too large. Specifically, the size of the model space is given by

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

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 an M consists of arbitrary combinations of patterns. Next, we introduce an efficient bottom-up search heuristic that—in terms of MDL—identifies a good pattern set.

5 PREMISE

To find a good pattern set in practice, we present Premise, which discovers *P*atterns *RE*constructing *MIS*classification *E*rrors. Instead of enumerating all possible patterns, it efficiently explores the search space in a bottom-up heuristical fashion.

5.1 Creating and Merging Patterns

Premise iteratively improves the model by adding, extending, and merging patterns until no more gain in the MDL score can be achieved. To ease the explanation, we will first introduce the setting without mutual exclusivity, but only conjunctive patterns. We start with an empty set of patterns M, the dataset is initially encoded only using singletons. We then search for candidate patterns that improve $L(M, D)$. These can be created in the following ways:

- *single items*: a single item $I \in \mathcal{I}$ 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 $\owedge(P, I)$ obtained by merging an existing pattern $P \in M$ with an item $I \in \mathcal{I}$,
- *pairs of patterns*: a new conjunctive pattern $\owedge(P_1, P_2)$ obtained by merging two existing patterns $P_1, P_2 \in M$.

Pairs of items for which the transaction sets barely overlap do not make for conjunctive patterns that compress well, hence, we introduce a minimum overlap threshold of 0.05 in all experiments, to speed up the search by pruning infrequent and therewith uninteresting patterns. We so have 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.

5.2 Filtering Noise

Additionally to the MDL score, Fischer and Vreeken [8] proposed to use Fisher's exact test as a filter for spurious patterns in their setting. 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 [9]. 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 that points towards misclassification. For patterns that point towards correct classification, the other tail of the distribution is tested (with a and b as well as c and d switching places). As with statistical pattern mining, an appropriate multiple test correction is not available. We however only use the test to *filter* candidates, hence false positive patterns passing the test are still evaluated in terms of MDL.

5.3 The Premise Algorithm

Combining the candidate generation and the MDL score from Section 4.2, we obtain Premise. We give the pseudo-code in Algorithm 1. Starting with the empty model, we generate candidates and for each of those, we compute the gain in terms of MDL (line 6) as well as the pattern's p-value (line 7). We select the candidate below a significance threshold α that reaches the highest gain (line 8-10) 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.

Algorithm 1: Premise

```
input: $D$ with $D^-$ and $D^+$, significance threshold $\alpha$
output: Heuristic approximation $M$ of $M^*$
1  do
2      $\Delta' \leftarrow 0$;
3      $M' \leftarrow M$;
4      $C \leftarrow$ createCandidates($M$);
5      for $P \in C$ do
6          $\Delta \leftarrow L(D, M \oplus P) - L(D, M)$ ;        // gain
7          $p \leftarrow$ FishersExactTest($P$);                  // p-value
8          if $p < \alpha$ and $\Delta > \Delta'$ then
9              $\Delta' \leftarrow \Delta$;
10             $M' \leftarrow M \oplus P$;
11         end
12     end
13     $M \leftarrow M'$;
14 while $\Delta' > 0$;
15 return $M$
```

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 1: Words and their nearest neighbors on *Visual7W*.

5.4 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 [26] 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 [12]. 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 1.

Based on the information of the embedding, we derive $\otimes$-clauses. For each item I, we explore mutual exclusivity in its $1 \ldots K$ closest neighbors, i.e. from $\otimes(I, nb(I, 1))$ until $\otimes(I, nb(I, 1), \ldots, nb(I, K))$ 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 $1, 2, \ldots K$ closest neighbours. We give the full algorithm in App. A.1.

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 (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 $\{m(I, nb(I, k)) \mid I \in \mathcal{I}\}$.

6 EXPERIMENTS

We evaluate and compare our approach on synthetic data with known ground truth as well as on real world classification settings. We compare against several methods that lend themselves for our setting. In particular, we compare to patterns derived from classification trees obtained from predicting the misclassification flag from the input tokens, subgroup discovery with misclassification flags as target variable, using weighted relative accuracy as quality function [2], the recent significant pattern miner SPuManTe [30], and the state of the art rule miner Grab, which similar to Premise is based on MDL [7]. We exclude SliceFinder [5] and the miner of disjunctive emerging patterns using hypergraphs by Vimieiro [41] from the experiments as they did not complete within 12 hours for a single run. If not specified otherwise, English FastText embeddings trained on CommonCrawl and Wikipedia are used [12]. Further training details are given in the Appendix. All experiments were performed on an Intel i7-7700 machine with 31GB RAM running Linux. The single-threaded C++ implementation of Premise required less then 10 minutes on all synthetic datasets. On the VQA datasets it finished within 20 minutes and on the NER data within 4 hours. For all existing methods we use the publicly available code, and make our code available for research purposes.[2]

6.1 Synthetic Data

To obtain a synthetic data set with similar token distributions as natural language text, we derive samples from the around 3.4k instances in the development set of the PennTreebank Corpus [24]. 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 $\mathcal{N}(150, 20)$. Each instance containing a pattern is then flagged as misclassification. Furthermore, we introduce two types of noise. *Shift* indicates the percentage of instances with a pattern that are actually labeled as misclassifications. The second type of noise is labeling of instances as misclassification although there is no pattern occurrence, which we refer to as *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 chosing 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. Patterns are created and planted in the data as in the previous experiments. Additionally, 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.

[2]https://bit.ly/39TAl7a

Results. For the first experiment set (Fig. 2a) of varying pattern length, we observe that subgroup discovery is 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. The statistical testing based SPuManTe 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. 2b and 2c, the tree based method completely breaks down even for just moderate amounts of noise. Subgroup Discovery and SPuManTe both perform consistently bad with F1 scores below .2. Out of the existing approaches, only Grab– due to its explicit noise encoding – 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.

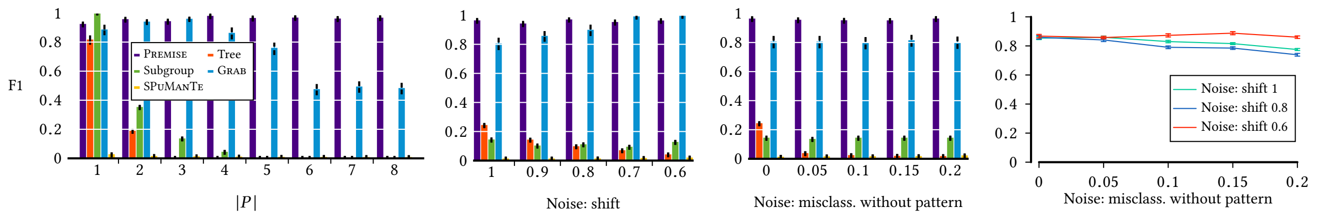
Since most baselines do not support discovering mutual exclusivity or proved to fail in the more simple setup of conjunctions, we only evaluate our proposed method on the fourth set of experiments. We observe that Premise is still able to retrieve patterns even in this challenging setup of complex clauses of length up to 20, with F1 scores close to .9, and is able to discover clauses in the presence of noise (Fig. 2d).

6.2 Real Data: VQA

Visual Question Answering (VQA) is the task of answering textual questions about a given image. It is a popular and challenging task at the intersection of vision and natural language processing. In this section, we analyze the misclassification of Visual7W [47] and the state-of-the-art LXMERT [38], both specific architectures for different VQA tasks. The pretrained Visual7W reaches 54% accuracy in 4-option multiple choice, LXMERT a validation score of 70% on their minival split. Both classifiers perform far from optimal and thus serve as interesting applications for our setting. Here, we derive misclassification data sets from application of the classifiers to the development sets.

In Tab. 2 we provide statistics about the data and retrieved patterns. Both the tree based method and SPuManTe 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. The subgroup discovery approach requires the user to specify the number of patterns a-priori, which is not known, hence we search for the top 100 patterns to get a succinct set of patterns. For the retrieved results, there are some patterns showing reasons for misclassification. However, the 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 a set mining approach that evaluates patterns *together.* Grab fails to retrieve meaningful results, likely due to the heuristic search.

Only Premise is able to provide a succinct and non-redundant description of misclassification, and is the only method to recover



(a) Varying pattern length

(b) Varying shift noise

(c) Varying label noise

(d) Complex clauses

Figure 2: *Synthetic data results.* On synthetic 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 the results of PREMISE on data containing patterns of mutual exclusive clauses for varying amounts of *shift noise* (d).

			PREMISE (this paper)			Tree		Subgroup		SPuManTe		GRAB	
Dataset	$\lvert\mathcal{I}\rvert$	$\lvert D\rvert$	$\lvert P^-\rvert$	$\lvert P^+\rvert$	$\overline{\lvert p\rvert}$	$\lvert P\rvert$	$\overline{\lvert p\rvert}$	$\lvert P\rvert$	$\overline{\lvert p\rvert}$	$\lvert P\rvert$	$\overline{\lvert p\rvert}$	$\lvert P\rvert$	$\overline{\lvert p\rvert}$
Visual7W	2429	28032	29	26	3.38	4309	3.55	100	2.32	575	2.92	1	1
LXMERT	5351	25994	41	34	2.69	3371	2.71	100	2.52	951	3.90	1	1

Table 2: *VQA data statistics.* For the two VQA classifiers, we provide general statistics about data dimensions, and for each method the number of discovered patterns ($\lvert P\rvert$) or if applicable number of patterns explaining misclassification ($\lvert P^-\rvert$), respectively correct classification ($\lvert P^+\rvert$). Additionally, we provide the average pattern length $\overline{\lvert p\rvert}$.

patterns that reflect synonyms or different styles of writings. In Tab. 3a, we list patterns found by PREMISE for the Visual7W classifier, where we can clearly see the advantage of the richer pattern language, allowing to find patterns such as $\owedge$(*what*, ⊗(*color, colors, colour*)). Generally, the patterns found by PREMISE highlight different types of wrongly answered questions, including counting questions, identification of specific objects and their colors, spatial reasoning, as well as higher reasoning tasks like reading signs. Furthermore, PREMISE retrieves both frequent patterns, such as $\owedge$(*how, many*), rare and highly specific patterns, such as $\owedge$(*on, wall, hanging*), and patterns containing related concepts and synonyms.

PREMISE also finds patterns explaining correct classification. These could indicate questions that are easy to answer but also questions that are just memorized by the network, since within the data set the answer is always the same. For instance, $\owedge$(*who, took,* ⊗(*photo, picture, pic, photos, photograph*)), although a difficult question, is nearly always answered by ”photographer“. Thus, patterns that are biased towards correct classification can indicate issues with the dataset. Another type of problematic questions is indicated by the pattern $\owedge$(*clock, time*), where usually the answer is ”UNK“, the actual time being replaced with the unknown word token by 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 for example the correct output to each instance, we observe for instances regarding the question when the picture was taken two different trends. On the one hand, the discovered pattern $\owedge$(*when,* ⊗(*daytime, nighttime*)) is associated with correct classification, the pattern $\owedge$(*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.

For the LXMERT classifier, a similar set of patterns is discovered by PREMISE, with examples given in Table 3b. We observe that both 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 modern LXMERT classifier can handle these better. Instead, many pattern indicate setting that require advanced capabilities like noticing fine-grained details or reading text on the images.

For the considered VQA classifiers, existing methods do not give succinct descriptions, and can not handle the richer language over conjunctions and mutual exclusivity. Such a language, however, provides a deeper understanding of misclassification errors. In contrast, the patterns discovered by PREMISE give intuitive and informative descriptions over such a richer language that allow to understand both, the issues of the dataset, as well as limits of the VQA classifier and hence can be used to improve classifier performance.

6.3 Real Data: NER

A machine learning classifier might perform well on the training and test data, its performance when deployed ”in the wild“ however is often much worse. Understanding the difference between the restricted training set-up and the open application of the classifier in real-life is important for being able to adapt and improve. Here, we investigate the popular LSTM+CNN+CRF architecture [22] for Named Entity Recognition (NER). NER is a sequence labeling task that identifies entities such as persons, locations and organizations in text and it is the basis for more advanced tasks like text search or virtual assistants. The classifier is trained on the standard NER

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

(a) Visual7W

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?

(b) LXMERT

Table 3: Our method discovers meaningful and easily interpretable patterns. For Visual7W (left) and LXMERT (right), 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 additional material.

dataset CoNLL03 [39], where it achieves an F1-score of 0.93, a performance good enough for a production setting. When evaluated on OntoNotes [43], a dataset which covers a wider range of text domains than CoNLL03, similar to how an open application of the classifier would encounter different text contexts, the performance drops to 0.61 F1-score on the development set.

We take each sentence as one instance and label it as misclassification if at least one token in the sentence is predicted incorrectly. This results in a misclassification dataset with about 16k instances and 23k unique items. Premise discovers patterns such as $\owedge$(*-LRB-*, *-RRB-*) that indicate different preprocessing of the text, where *-LRB-* is an alternative form for the opening bracket, which is specific to the OntoNotes data, and thus should be properly handled by the NER classifier. Patterns also indicate problems with differing labeling conventions. For example, we find the patterns $\owedge$(*'s*) and $\owedge$(*Wall, Street*), which turn out to be handled differently for entities in OntoNotes, respectively CoNLL03. Apart from patterns that highlight dataset differences, we can also isolate issues with OntoNotes alone, which contains bible excerpts that are not labeled at all. We discover this via several found patterns that describe this domain (*God*, *Jesus*, *Samuel*).

While these patterns do give insight in what is going wrong when applying a classifier to a different dataset, it is left to show that we can act upon them. Hence, we select the top 50 patterns according to gain in MDL and sample for each pattern 5 sentences containing the pattern uniformly at random from the OntoNotes training data. Using this data, we fine-tune the CoNLL03 classifier and compare it to fine-tuning on the same number of random sentences from the training data. The sampling and fine-tuning is repeated 20 times with different seeds. Using the pattern-guided data, the performance is improved by 9.8% to a mean F1-score of 0.67 (SE 0.003). The randomly chosen instances only result in a small improvement of 1.6% percent to a mean F1 score of 0.62 (SE 0.005), showing that we can act on patterns discovered by Premise, using them to guide fine-tuning based on the original training data to improve performance on deployment.

7 DISCUSSION & CONCLUSION

We considered the problem of characterizing classification errors of machine learning models, in particular for NLP settings with high dimensional, sparse input spaces. In particular, we are interested in easily interpretable statements about the data that describe where it went wrong. Here, we formulated this problems in terms of the Minimum Description Length Principle to discover a succinct set of interpretable patterns that describe the observed classification error. Considering a pattern language expressing conjunctions of mutual exclusive tokens, these patterns are human interpretable and easy to act upon, while capturing important features of natural language, such as related words or synonyms. To find related words in a robust and classifier-agnostic way, we leverage information from pretrained word embeddings to guide the search for good sets of mutual exclusive tokens. To discover pattern sets efficiently, we proposed Premise, which employs a bottom-up heuristic search scheme and showed to yield good approximations to the optimal model on data with known ground truth.

In contrast to the state-of-the-art, Premise can model both conjunctions and mutual exclusivity, while at the same time being robust to noise and scalable to the large input spaces encountered in NLP tasks. In experiments on synthetic data, we showed that Premise consistently recovers the ground truth patterns across varying length, and different types of noises. Existing methods work only in very limited settings of extremely low amounts of noise and for short patterns.

On real data, we compared the existing methods in a case study, analyzing the misclassifications of two modern Visual Question Answering classifiers. While some of the competing methods did retrieve reasonable explanations, these were highly redundant and barely interpretable for human experts 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 of classification errors, revealing that models struggle with counting, spatial orientation, reading, and identifies shortcomings in training data.

To show that the pattern sets is not only interpretable and gives interesting insights, but also actionable, we analyzed 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. Inspecting the retrieved patterns confirms that also for NER models Premise is able to retrieve meaningful patterns explaining misclassification. To show actionability, we fine-tune the NER model using samples from the training data that contain discovered patterns, increasing F1 classifier performance by almost 10%, whereas fine-tuning with arbitrary training samples does not yield much improvement.

Premise discovers interpretable and actionable relation between input and classification errors. It makes for engaging future work to also consider elements of the classifier itself – such as neuron activations – to search for interesting patterns, to relate elements of the model itself with the classification error. While this would give interesting insights, it is much harder to act upon, and requires a model specific approach. In that line, considering the hundreds of thousands of neuron activations of an entire network at once poses additional challenges regarding scalability, which again serve as an interesting direction of future research.

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. Atzmueller. 2015. Subgroup discovery. *WIREs Data Mining and Knowledge Discovery* 5, 1 (2015), 35–49.

[3] R. Bayardo. 1998. Efficiently mining long patterns from databases. In *SIGMOD*. 85–93.

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

[5] Y. Chung, T. Kraska, N. Polyzotis, K. H. Tae, and S. E. Whang. 2020. Automated Data Slicing for Model Validation: A Big Data - AI Integration Approach. *IEEE Trans. Knowl. Data Eng.* 32, 12 (2020), 2284–2296.

[6] G. Dong and J. Li. 1999. Efficient mining of emerging patterns: Discovering trends and differences. In *KDD*. ACM New York, NY, USA, 43–52.

[7] J. Fischer and J. Vreeken. 2019. Sets of Robust Rules, and How to Find Them. In *ECML PKDD*. Springer.

[8] J. Fischer and J. Vreeken. 2020. Discovering Succinct Pattern Sets Expressing Co-Occurrence and Mutual Exclusivity. In *KDD*. 813–823.

[9] R. A. Fisher. 1922. On the interpretation of χ 2 from contingency tables, and the calculation of P. *Journal of the Royal Statistical Society* 85, 1 (1922), 87–94.

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

[11] M. Gardner, Y. Artzi, V. Basmova, J. Berant, B. Bogin, S. Chen, P. Dasigi, D. Dua, Y. Elazar, A. Gottumukkala, N. Gupta, H. Hajishirzi, G. Ilharco, D. Khashabi, K. Lin, J. Liu, N. F. Liu, P. Mulcaire, Q. Ning, S. Singh, N. A. Smith, S. Subramanian, R. Tsarfaty, E. Wallace, A. Zhang, and B. Zhou. 2020. Evaluating Models' Local Decision Boundaries via Contrast Sets. In *EMNLP*. 1307–1323.

[12] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov. 2018. Learning Word Vectors for 157 Languages. In *LREC*.

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

[14] 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.

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

[16] M. A. Hedderich, D. Adelani, D. Zhu, J. Alabi, U. Markus, and D. Klakow. 2020. Transfer Learning and Distant Supervision for Multilingual Transformer Models: A Study on African Languages. In *EMNLP*. 2580–2591.

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

[18] G. Lee, S. Kim, and S. Hwang. 2019. QADiver: Interactive Framework for Diagnosing QA Models. In *AAAI*. 9861–9862.

[19] F. Lemmerich and M. Becker. 2018. pysubgroup: Easy-to-use subgroup discovery in python. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 658–662.

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

[21] F. Llinares-López, M. Sugiyama, L. Papaxanthos, and K. Borgwardt. 2015. Fast and Memory-Efficient Significant Pattern Mining via Permutation Testing. In *KDD (KDD '15)*. 725âĂŞ734.

[22] X. Ma and E. Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In *ACL*. 1064–1074.

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

[24] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Comput. Linguist.* 19, 2 (June 1993), 313–330.

[25] P. Miettinen. 2010. Sparse Boolean Matrix Factorizations. In *ICDM*. 935–940.

[26] G. A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM* 38, 11 (1995), 39–41.

[27] 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.

[28] P. K. Novak, N. Lavrac, and G. I. Webb. 2009. Supervised Descriptive Rule Discovery: A Unifying Survey of Contrast Set, Emerging Pattern and Subgroup Mining. *JMLR* 10 (2009), 377–403.

[29] 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.

[30] L. Pellegrina, M. Riondato, and F. Vandin. 2019. SPuManTE: Significant Pattern Mining with Unconditional Testing. In *KDD*. ACM, 1528âĂŞ1538.

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

[32] S. Pradhan, A. Moschitti, N. Xue, H. T. Ng, A. Björkelund, O. Uryupina, Y. Zhang, and Z. Zhong. 2013. Towards Robust Linguistic Analysis using OntoNotes. In *CoNLL*.

[33] M. T. Ribeiro, T. Wu, C. Guestrin, and S. Singh. 2020. Beyond Accuracy: Behavioral Testing of NLP Models with CheckList. In *ACL*. 4902–4912.

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

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

[36] M.-A. Rondeau and T. J. Hazen. 2018. Systematic Error Analysis of the Stanford Question Answering Dataset. In *Proceedings of the Workshop on Machine Reading for Question Answering*. 12–20.

[37] K. Smets and J. Vreeken. 2012. Slim: Directly Mining Descriptive Patterns. In *SDM*. SIAM, 236–247.

[38] H. Tan and M. Bansal. 2019. LXMERT: Learning Cross-Modality Encoder Representations from Transformers. In *EMNLP*. 5100–5111.

[39] E. F. Tjong Kim Sang and F. De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *CoNLL*. 142âĂŞ147.

[40] M. van Leeuwen and A. J. Knobbe. 2012. Diverse subgroup set discovery. *Data Min. Knowl. Disc.* 25, 2 (2012), 208–242.

[41] R. Vimieiro. 2012. *Mining disjunctive patterns in biomedical data sets*. Ph.D. Dissertation. The University of Newcastle, Australia.

[42] J. Vreeken and N. Tatti. 2014. *Interesting Patterns*. Springer, 105–134.

[43] R. Weischedel, M. Palmer, M. Marcus, E. Hovy, S. Pradhan, L. Ramshaw, N. Xue, A. Taylor, J. Kaufman, M. Franchini, M. El-Bachouti, R. Belvin, and A. Houston. 2013. OntoNotes Release 5.0 LDC2013T19. Web Download, Linguistic Data Consortium. (2013).

[44] S. Wrobel. 1997. An algorithm for multi-relational discovery of subgroups. In *Principles of Data Mining and Knowledge Discovery*, Jan Komorowski and Jan Zytkow (Eds.). Springer, 78–87.

[45] T. Wu, M. T. Ribeiro, J. Heer, and D. Weld. 2019. Errudite: Scalable, Reproducible, and Testable Error Analysis. In *ACL*. 747–763.

[46] M. J. Zaki. 2000. Scalable Algorithms for Association Mining. *IEEE TKDE* 12, 3 (2000), 372–390.

[47] Y. Zhu, O. Groth, M. Bernstein, and L. Fei-Fei. 2016. Visual7W: Grounded Question Answering in Images. In *IEEE CVPR*.

A APPENDIX

A.1 Algorithm `createCandidates`

Algorithm 2: createCandidates

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

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 and Vreeken [8].

Theorem A.1. *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.*

Proof. 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

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

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

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

Other permutations and larger clauses follow the same reasoning.

□

A.3 Experimental Details

For the decision tree, patterns are exracted from a 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.

For the subgroup discovery, the implementation by Lemmerich and Becker [19] is used with depth-first search. The size of the result set and the maximum depth are set to the ground truth for the synthetic data and to 100 and 5 for the VQA datasets. SPuManTe is used with the authors' default parameters, setting its sample size to the dataset size. Grab is used in the authors' implementation but restricting the head to the labels and the tail to all other items.

For Visual7W and LXMERT, we use the published, pretrained models by the corresponding authors. For the LSTM+CNN+CRF classifier for NER, we follow the specific set-up from [16] with the English FastText embeddings. For OntoNotes, the data split by Pradhan et al. [32] is used. The fine-tuning data consists of 240 instances/sentences as two patterns did not match any training data. Fine-tuning on the additional data is performed for 30 epochs.