ATLANTIS PRESS

Atlantis Studies in Uncertainty Modelling, volume 1

11th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2019)

Mining data quality rules based on T-dependence

Toon Boeckling[a], **Antoon Bronselaer**[b] **and Guy De Tré**[c]
[a] Department of Telecommunications and Information Processing,
Ghent University, Sint-Pietersnieuwstraat 41, 9000 Ghent, Belgium, toon.boeckling@ugent.be
[b] Department of Telecommunications and Information Processing,
Ghent University, Sint-Pietersnieuwstraat 41, 9000 Ghent, Belgium, antoon.bronselaer@ugent.be
[c] Department of Telecommunications and Information Processing,
Ghent University, Sint-Pietersnieuwstraat 41, 9000 Ghent, Belgium, guy.detre@ugent.be

Abstract

Since their introduction in 1976, edit rules have been a standard tool in statistical analysis. Basically, edit rules are a compact representation of non-permitted combinations of values in a dataset. In previous work, edit rules are mined automatically as value-combinations showing strong negative correlation tested under stochastic independence using the traditional notion of lift. In this paper, we generalize the traditional notion of lift to that of T-lift, where stochastic independence is generalized to T-dependence. We show several interesting properties of edit rules under different T-lift measures and proof that edit rules under the minimum t-norm can be computed efficiently by use of frequent pattern trees. Experiments support this result and show that there is a weak to medium correlation in the rank order of edit rules obtained under the minimum and product t-norms.

Keywords: Data Quality, Pattern Mining, Consistency, Triangular Norms

1 Introduction

Assessment of the *quality of data* [6, 23] remains an important topic with the ever growing potential of using data for a variety of tasks. As a lot of research has been done in the past dealing with data quality measurement, often recognized as a multi-dimensional problem [5, 21], a common way to assess the quality of data is by assuming a set of data quality rules (e.g. non-permitted value combinations) to which the data must comply. From this point of view, it is assumed that data is of the best possible quality if it satisfies all rules in this set. This assumption is formally stated in [8].

A recurring question is how to construct a set of data quality rules. One possible approach is to manually define a set of rules independently of the underlying data source in which the data must satisfy those rules. This requires that internal knowledge of the data semantics is available and that meaningful rules can be derived based upon this knowledge. An advantage of manually defining data quality rules is that this set of rules does not change when data is updated over time. In contrast to the manual definition of data quality rules, one can also apply a data-driven approach in which rules are derived by analyzing the underlying data source. Since this approach does not require any domain-specific knowledge and the time to collect a set of rules is limited, recent interest shifted to the autonomous mining and evaluation of data quality rules derived from a collection of data objects. If consistency is assumed as dimension, a large portion of the autonomous methods focuses on generating rules assessing the mutual (in)dependence of attribute-values in a data object. Examples of such quality rules are functional dependencies [1, 17], conditional functional dependencies [7, 9, 12, 13] and approximate functional dependencies [17, 19]. A more recent approach for the generation of data quality rules uses association analysis [2]. This method allows to uncover very common [4] as well as illegal [22] combinations of values in a collection of data objects.

In [14], Fellegi and Holt introduced *edit rules* as illegal combinations of values used to detect errors in census data. Rammelaere et al. define in [22] forbidden itemsets as a specialization of edit rules for which the likeliness of the itemsets does not exceed a predefined threshold. The authors propose to use the *lift* of an itemset as likeliness function and, given a collection of data objects, they mine low lift itemsets by using a variant of the Eclat algorithm [25] (i.e., FBIMiner) that exploits properties of the lift measure. The idea behind low lift itemsets is to search for combinations of values that display a much stronger negative correlation than is to be expected under the assumption of stochastic independence. This negative correlation indicates that these combinations are interesting to investigate as possible edit rules in the sense of Fellegi and Holt [14].

age	relation	sex	support
31-50	Husband	Male	7
>50	Husband	Male	3
<18	Own-child	Female	1
<18	Own-child	Male	1
22-30	Husband	Male	1
22-30	Own-child	Female	1
22-30	Own-child	Male	1
22-30	Not-in-family	Male	1
31-50	Not-in-family	Female	1
31-50	Wife	Female	1
>50	**Husband**	**Female**	1
>50	Not-in-family	Female	1

Table 1: An example dataset of categorical data with three attributes and the number of rows for each combination. The bold values together yield an edit rule.

The main contribution of this paper is to generalize the notion of low lift edit rules introduced in [22] and investigate other types of negative correlation. More specifically, stochastic independence is generalized to T-dependence [10] based on a triangular norm (t-norm) T [18, 20]. It is shown that, besides the edit rules obtained through lift, there are other types of edit rules yielding interesting properties, which can be found efficiently. An example of a dataset that will be used as a running example throughout this paper is shown in Table 1, where the bold values together are an example of an edit rule and the number of times a row occurs in the dataset is given in the 'support' column.

The remainder of the paper is organized as follows. In Section 2, state-of-the-art work according to data quality rule mining is pointed out. Preliminary concepts involving association analysis (measures) are explained in Section 3. In Section 4, the lift measure is generalized and the properties of edit rules that are obtained under different t-norms are studied. In particular, the special role played by the minimum t-norm is pointed out. In Section 5, the minimum t-norm is further investigated and an efficient algorithm is proposed to mine the corresponding edit rules. After this, the rank correlations between different t-norms are empirically verified and the execution times of the proposed mining algorithm and the FBIMiner algorithm are compared in Section 6. Finally, in Section 7, concluding remarks and further research opportunities are provided.

2 Related work

Recently, much research has been done on the autonomous generation and evaluation of data quality rules [11]. Most approaches can be distinguished by the type of rules they aim to generate. A significant number of papers revolve around the discovery of *functional dependencies*, with the TANE algorithm as the most famous example [17]. In line with these approaches, variants or generalizations of functional dependencies have been studied. In particular, *conditional* functional dependencies [9, 13] and *approximate* functional dependencies have been investigated extensively [19].

Besides these approaches, there has also been a strong focus on the application of *association analysis* to find data quality rules. A lot of these approaches identify common patterns underlying a data source, such as frequent itemsets [16] or high-confidence association rules [4] and then label data that meet few of these patterns as possible violations. In a way, these approaches provide an effective way to find outliers in categorical datasets.

A third way of defining data quality rules is by use of *edit rules* [14], which are compact representations of non-permitted combinations of values in a dataset. Edit rules have been used as a standard tool in statistical processing for decades and come with a set of appealing properties. Very recently, an approach was given to automatically discover such edit rules [22]. In this paper, we build on the results from [22] and provide a more generic notion of edit rules.

3 Preliminaries

A finite dataset D with attributes $(A_1, \dots, A_n)$ defined over the universe of discourse $\mathcal{I}$ consists of n-dimensional data objects o. Every o is of the form $(j, I_j = [(A_1, v_1), \dots, (A_n, v_n)])$ with $j \in \mathbb{N}$ the unique identifier of the object and $v_i \in \text{dom}(A_i), \forall i \in \{1, \dots, n\}$. The set of all possible attribute-values (A, v) makes up $\mathcal{I}$.

An itemset I consists of a finite set of items (A, v). It contains at most one item for each A. A data object o_j is said to contain I if $I \subseteq I_j$. An association rule of an itemset I is a statement of the form $X \Rightarrow Y$ with X, Y both itemsets and the requirements that $X, Y \neq \emptyset$, $X \cup Y = I$ and $X \cap Y = \emptyset$

In association analysis, various measures are used to indicate the interestingness of an itemset or a rule. In this work, the support, frequency, confidence and lift will be used for this purpose. The support of an itemset I in a dataset D is given by:

$$\operatorname{supp}(I, D) = |\{o_j \mid o_j \in D \wedge I \subseteq I_j\}| \tag{1}$$

and indicates the absolute number of objects $o \in D$ containing the itemset I. For an association rule $X \Rightarrow Y$, the support of the rule in dataset D is defined by $\operatorname{supp}(X \cup Y, D)$. The frequency of an itemset in D, $\mathrm{F}(I, D)$, is given by $\operatorname{supp}(I, D)/|D|$. The confidence of an association rule $X \Rightarrow Y$ in D is defined by

$$\operatorname{conf}(X \Rightarrow Y, D) = \frac{\operatorname{supp}(X \cup Y, D)}{\operatorname{supp}(X, D)}. \tag{2}$$

and estimates the conditional probability $\Pr(Y|X)$. The lift measure of an itemset, used in the remainder, is defined by

$$\operatorname{lift}(I, D) = \frac{\mathrm{F}(I, D)}{\min\limits_{\emptyset \subset J \subset I} \{\mathrm{F}(J, D) \times \mathrm{F}(I \setminus J, D)\}}. \tag{3}$$

and is a modification of the basic definition of the lift of an itemset, assuming pairwise stochastic independence. Usually, lift is defined by the ratio of the observed joint probability of the items in this itemset to the expected joint probability assuming full stochastic independence among those items [26], but Rammelaere et al. use this modification to avoid a bias towards larger itemsets. Within this definition, every binary partition of the itemset is tested and the least value is adopted in the resulting lift measure of this itemset. An example application of the lift measure is given in Section 4.

4 Edit rules under T

In [22], Rammelaere et al. adopt the pairwise independent lift measure (i.e., Eq. (3)) as a way to evaluate the correlation between items. In this paper, a more general type of correlation is considered by assuming events to be T-dependent [10], where T is a triangular norm [18]. In what follows, we adopt the notations and definitions given in [18]: T_M (minimum), T_P (product), T_L (Łukasiewicz), T_D (drastic product), T_0^H (Hamacher product) and T_x^{SS} (Schweizer-Sklar family). This leads to the following definition.

Definition 1. *The T-lift of an itemset I in a dataset D based on a triangular norm T is defined as*

$$\lambda_T(I, D) = \frac{\mathrm{F}(I, D)}{\min\limits_{\emptyset \subset J \subset I} \{T\left(\mathrm{F}(J, D), \mathrm{F}(I \setminus J, D)\right)\}}. \tag{4}$$

In line with the approach in [22], an itemset I is an edit rule under T if and only if

$$\lambda_T(I, D) \leq \tau \tag{5}$$

Itemset	λ_{T_M}	λ_{T_P}	λ_{T_L}
{rel:husband, sex:female}	0.17	0.28	∞
{age:>50, sex:male}	0.60	0.86	∞

Table 2: Lift measures for different t-norms of two itemsets from the dataset in Table 1.

where $\tau \in]0, 1[$ is an appropriate threshold. In Table 2, the first itemset is suspicious under λ_{T_M} and λ_{T_P} given the dataset in Table 1.

In the remainder, we will show some interesting properties of edit rules mined under T-lift in general and under specific choices of T.

A first interesting aspect is that there are specific relations between edit rules obtained under different t-norms. More specifically, the following proposition holds.

Proposition 1. *If T_1 and T_2 are two t-norms such that $T_1 \leq T_2$, then any edit rule under T_1 is also an edit rule under T_2.*

Proposition 1 has some interesting consequences. Because $T_1 \leq T_2$, it is easy to see that $\lambda_{T_1} \geq \lambda_{T_2}$ for any itemset I. This indicates that, given a predefined τ, the collection of edit rules mined under T_1, called $\mathcal{E}_{T_1}$, is a subset of the collection of edit rules mined under T_2, $\mathcal{E}_{T_2}$. Therefore, according to the properties of t-norms, it can be said that the pointwise smallest t-norm, T_D, generates the smallest set of edit rules and the pointwise largest t-norm, T_M, generates the largest set of edit rules, being a superset of $\mathcal{E}_T$ for each $T \leq T_M$ and a fixed τ. Moreover, any (efficient) algorithm (e.g. FBIMiner for T_P) that exists for finding edit rules under T_2 can be used as a candidate selector for edit rules under T_1. Secondly, when one aims at mining a constant set of rules with a pointwise increasing T, τ can be decreased constantly, because $\lambda_{T_1} \geq \lambda_{T_2}$. As is shown in Section 6, this implies a faster execution time of the mining algorithm.

Let us now consider some specific (types of) t-norms and evaluate the semantics of their corresponding edit rules and the influence on the ability to find edit rules efficiently. It is trivial that under $T = T_P$, the framework as described in [22] is obtained, so we will not detail T_P further. The most extreme variant of edit rules are those obtained under T_D. λ_{T_D} will only yields a value if the frequency of at most one value in the itemset I under investigation is not 1. In any other case, there exists a binary partition in which both partitions have a frequency lower than 1, which results in $T_D = 0$ and therefore will be taken as the minimum in Eq. (4). A consequence is that there can exist at most one attribute in I with more than 1 value in the dataset D.

From a semantical point of view, T_D-dependence is a form of independence that is so weak that I can only be an edit rule under T_D if there is an enormous negative correlation. From a practical point of view, it means that, when searching for edit rules under T_D, only those itemsets containing at most 1 attribute with more than 1 value have to be considered.

In case of T_L, which is pointwise smaller than T_P, I can never be an edit rule if there exists some $J \subset I$ such that $\mathrm{F}(J) + \mathrm{F}(I \setminus J) \leq 1$. The reason is that this will result in $T_L = 0$ which yields no result for λ_{T_L}. A direct implication is that an itemset I can contain at most one value with a frequency lower than 0.5. In any other case, there exists a binary partition in which both partitions have a frequency lower than 0.5, satisfying $\mathrm{F}(J) + \mathrm{F}(I \setminus J) \leq 1$.

This ability to reduce the number of itemsets to check (prune) can be used by any nilpotent t-norm. Indeed, any nilpotent t-norm has zero divisors, which implies that an itemset I can only be an edit rule if its binary partitions are all sufficiently frequent and can not result in a t-norm value equal to 0. In general, the more zero divisors T has, the higher the negative correlation should be for I to be an edit rule. Important to note is that this can be done independent of the choice of τ.

Finally, the pointwise largest t-norm, T_M, is considered. For completeness, the definition of the λ_{T_M} is given by

$$\lambda_{T_M}(I, D) = \frac{\mathrm{F}(I, D)}{\min\limits_{\emptyset \subset J \subset I} \{\min(\mathrm{F}(J, D), \mathrm{F}(I \setminus J, D))\}}. \quad (6)$$

For T_M, the following proposition holds.

Proposition 2. *The definition of λ_{T_M} of an itemset I can be reduced to*

$$\lambda_{T_M}(I, D) = \frac{\mathrm{F}(I, D)}{\min\limits_{i \in I} \{\mathrm{F}(I \setminus \{i\}, D)\}}. \quad (7)$$

In other words, only the frequencies of the largest strict subsets of I have to be considered during the calculation of λ_{T_M}.

Proof. In Eq. (6), T_M is calculated for every binary partition consisting of strict subsets of I and λ_{T_M} will be characterized by the binary partition resulting in the lowest T_M. Because the frequency of an itemset is monotonically decreasing according to the $\subseteq$-relation, it can be said that T_M of a binary partition consisting of a partition p_1 with cardinality of $|I| - 1$ will always be lower than T_M of a binary partition consisting of a partition $p_2 \subset p_1$. Therefore, one should only evaluate the frequency of the largest strict subsets of I as given in Eq. (7). $\square$

A consequence of Proposition 2 is that not all binary combinations of strict subsets of I have to be evaluated for the calculation of λ_{T_M}, but only the largest strict subsets. This reduces the calculation to a linear problem instead of an exponential one. Semantically, Eq. (7) searches for the association rule of the form $I \setminus \{i\} \Rightarrow i$ which has the maximum confidence over all items $i \in I$. The reason for this is that the corresponding itemset I will be registered as an edit rule if and only if the confidence of each association rule of this form is low enough according to the threshold τ. If it turns out that I is an edit rule, the item $i \in I$ resulting in the lowest confidence of the association rule of the given form, will provide most credible proof of this itemset being non-permissible. This can be an interesting candidate for a value imputation later. Introducing these semantics of confidence establishes a very interesting connection between lift on the one hand and confidence on the other hand, as they both seem to be the natural consequence of a particular t-norm. Besides these semantics, there are however numerous other advantages of using T_M as will be explained in Section 5.

5 Efficient mining of edit rules under T_M

In this section, mining edit rules under T_M is further investigated. First, two additional properties held by the operator T_M are pointed out. After this, an algorithm exploiting those properties is proposed that can be used to efficiently mine edit rules under T_M given a dataset D.

5.1 Properties

As a first additional property of T_M, a lower bound is defined on the support of a strict subset J of an edit rule I mined under T_M.

Proposition 3. *If I is an edit rule under T_M in dataset D, it holds that, for each subset $J \subset I$,* $\mathrm{supp}(J, D) \geq \frac{1}{\tau}$.

Proof. Given an edit rule I under T_M in dataset D. According to the definition of edit rules (Eq. (5)) and λ_{T_M} (Eq. (6)), it is easy to see that for each subset $J \subset I$

$$\tau \geq \frac{\mathrm{F}(I, D)}{\mathrm{F}(J, D)} = \frac{\mathrm{supp}(I, D)}{\mathrm{supp}(J, D)} \geq \frac{1}{\mathrm{supp}(J, D)}. \quad (8)$$

$\square$

Proposition 3 shows that an itemset $J \subset I$ cannot generate an edit rule I under T_M if the support of J is lower than $\frac{1}{\tau}$. Vice versa, once I has a single subset J with a support lower than $\frac{1}{\tau}$, I cannot be an edit rule under T_M. This property can be exploited in several ways. An interesting strategy would be to find all itemsets that are $\frac{1}{\tau}$-frequent, which can be done in parallel by using the partition algorithm [24] that distributes the computation over different nodes.

The second proposition defines an upper bound on the frequency of an itemset I for it to become a candidate edit rule under T_M.

Proposition 4. *If I is an edit rule under T_M in dataset D then* $\mathrm{F}(I, D) \leq \tau$.

Proof. Proposition 4 follows automatically from the definition of edit rules (Eq. (5)), the definition of λ_{T_M} (Eq. (6)) and the property that the support of an itemset monotonically decreases according to the $\subseteq$ relation. □

A direct consequence is that any itemset I as well as all its subsets $J \subset I$ are no edit rules under T_M if the frequency of I is higher than τ. The upper bound proven in Proposition 4 can be further decreased to:

$$\mathrm{F}(I, D) \leq \tau * \mathrm{F}(J, D) \qquad (9)$$

for any strict subset $J \subset I$, in particular the subset with the lowest frequency.

5.2 An FP-tree algorithm

It is already pointed out that algorithms traditionally designed for finding frequent itemsets can be modified to search for edit rules. In [22], a variant of the Eclat-algorithm [25] (i.e., FBIMiner) is proposed to find all edit rules under T_P. A different approach is taken here by proposing a frequent pattern tree (FP-tree) based algorithm [15] to find all edit rules under T_M.

An FP-tree is basically a data structure that represents a set of itemsets in a compact manner by representing highly frequent patterns as a single branch in a tree. Each node represents an item and each path represents a combination of items. A node has a counter that indicates how many times the path consisting of that node and its ancestors, occurs in the dataset. For details on how to compute FP-trees, we refer to [15]. In contrast to algorithms such as Eclat [25] and Apriori [3], an FP-tree already contains all candidate itemsets that should be evaluated. Besides that, Propositions 2-4 can be easily exploited by using FP-trees. An initial FP-tree for finding edit rules under T_M for the example dataset from Table 1 with $\tau = 0.2$ is shown in Figure 1.

The algorithm used for the mining of edit rules $\mathcal{E}_{T_M}$ in D with threshold τ consists of two procedures, as illustrated in Algorithm 1. The procedure **makeTree**

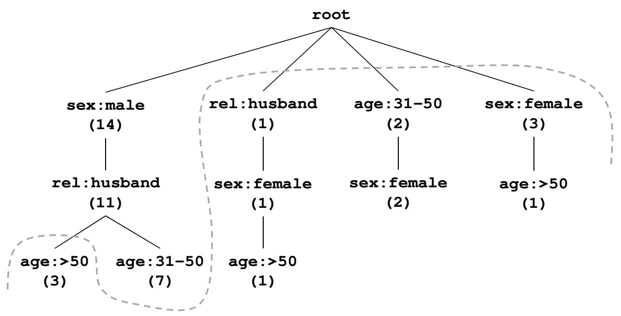

Figure 1: An FP-tree for finding edit rules under T_M for the dataset from Table 1 assuming $\tau = 0.2$. The grey dashed line indicates the candidate items for an empty prefix.

(Algorithm 2) generates an FP-tree $\mathcal{T}$ from the dataset D removing items with a support lower than or equal to minS (minimum support) and marking nodes in the tree with a count lower than maxS (maximum support). After the construction of the initial tree, the recursive procedure **searchTree** (Algorithm 3) mines edit rules $\mathcal{E}_{T_M}$ given $\mathcal{T}$ and a threshold τ. In the remainder, a detailed description of both procedures is given.

Algorithm 1 mineEditRules

```
1: function MINEEDITRULES($D$, $\tau$)
2:     $\mathcal{T}$ = MAKETREE($D$, $\lfloor \frac{1}{\tau} \rfloor$, $\lceil |D| * \tau \rceil$)
3:     return SEARCHTREE($\mathcal{T}$, $\emptyset$, $\tau$)
```

Basically, the **makeTree** procedure generates an FP-tree data structure comprising all data from D [15]. This function creates an FP-tree $\mathcal{T}$ containing all items with a support larger than minS. For the initial tree, minS equals $\lfloor \frac{1}{\tau} \rfloor$ because Proposition 3 shows that single items with a support lower than this bound are unable to generate edit rules and they can not be edit rules themselves. Because of this, all items from Table 1 with a count below $1/0.2 = 5$ are discarded in Figure 1. The value for maxS is initially equal to $\lceil |D| * \tau \rceil$ because of Proposition 4 and is used to identify possible candidate items that need to be further investigated. An item is a candidate if and only if there is a node containing this item with a count lower than or equal to the upper bound. In other words, if all nodes matching a certain item in $\mathcal{T}$ have a support higher than maxS, the item will not be denoted as candidate edit rule because of Proposition 4. This is an important advantage in comparison with the approach described in [22] where they use the upper bound on the frequency of an itemset only to limit lift computations but not to prune edit rules. By using an FP-tree data structure and the definition of λ_{T_M}, candidate reduction can be done easily. In our example, with

$|D| = 20$ and $\tau = 0.2$, we have initially that only items that occur somewhere in the tree with a count lower than or equal to 4, are candidate items. This candidate border is shown in Figure 1 for the initial tree as the grey dotted line. An itemset containing the item "sex:male" can thus never be an edit rule under T_M if $\tau = 0.2$.

Algorithm 2 makeTree

```
1: function MAKETREE($D$, minS, maxS)
2:     $\mathcal{T}$ = FPTREECONSTRUCT($D$, minS)
3:     $\mathcal{T}$.candidates ← $\emptyset$
4:     for all leaf $\in \mathcal{T}$.leaves do
5:         n ← leaf
6:         while n $\neq \mathcal{T}$.root $\wedge$ n.count $\leq$ maxS do
7:             $\mathcal{T}$.candidates.add(n.item)
8:             n ← n.parent
9:     return $\mathcal{T}$
```

The core of the mining algorithm is resided in the second procedure, **searchTree**. This procedure evaluates the lift measure of all possible candidates by recursively expanding candidate edit rules. The approach is in line with the FP-growth algorithm described in [15]. Since the purpose is not to find frequent itemsets but edit rules, it is necessary to adapt the algorithm in such a way that edit rules can be mined efficiently. First, it is not useful to test all items, but only possible candidate edit rules as described above (line 3). Besides that, if the support of the current itemset under investigation is too low according to Proposition 3, the itemset will not be further expanded (line 7). When generating a frequent pattern tree conditioned on the current itemset under investigation, a new upper bound is calculated according to Proposition 4 (line 9), so that in the recursive call, the evaluated candidates are limited. Finally, as already described as a consequence of Proposition 2, the calculation of λ_{T_M} reduces to a linear problem.

Algorithm 3 searchTree

```
1: function SEARCHTREE($\mathcal{T}$, $I_C$, $\tau$)
2:     $\mathcal{E} \leftarrow \emptyset$
3:     for all $i \in \mathcal{T}$.candidates do
4:         $I \leftarrow \{i\} \cup I_C$
5:         if $|I| > 1$ & $\lambda_{T_M}(I, D) <= \tau$ then
6:             $\mathcal{E} \leftarrow \mathcal{E} \cup \{I\}$
7:         if supp$(I, D) < \frac{1}{\tau}$ then
8:             continue
9:         maxS ← $\lceil \tau * \text{supp}(I, D) \rceil$
10:        $\mathcal{T}_{proj}$ ← MAKETREE(proj$(D, i)$, 1, maxS)
11:        if $\mathcal{T}_{proj}$.root.hasChildren then
12:            $\mathcal{E} \leftarrow \mathcal{E} \cup$ SEARCHTREE($\mathcal{T}_{proj}$, $I$, $\tau$)
13:    return $\mathcal{E}$
```

To conclude this section we note that, because of Proposition 1, the FP-tree algorithm can easily be adapted to work for any t-norm by simply replacing λ_{T_M} with λ_T on line 5 in Algorithm 3.

6 Results

In this section, we report the results of some experiments to show the usefulness of our contributions. More specifically, we are interested in answering two questions:

- How do rankings of edit rules obtained under different t-norms compare?
- How efficient is the FP-tree algorithm?

To answer the first question, the rank order correlation of edit rules based on their T-lift measure λ_T for different t-norms is calculated. This should provide insight in how similar the edit rules under different t-norms are. To answer the second question, we will report execution times of the FP-tree algorithm, described in Section 5, and compare them with execution times of FBIMiner [22].

6.1 Data and setup

To conduct experiments, two datasets from the UCI repository[1] are used. These datasets are the 'adult' (44842 data objects, 202 items over 11 attributes) and the 'census-income' (199524 data objects, 235 items over 12 attributes) dataset, both containing census data from the U.S. Census bureau. The main reason to use these datasets is to be able to make a fair comparison with the results from [22]. From that perspective, the same preprocessing steps were taken as those reported in [22]. An implementation of the FP-tree algorithm was made in Java. The applications are tested on an Intel Core i7 (1.80GHz) processor with 16GB RAM running Windows 10.

6.2 Correlations between edit rule rankings

In Tables 3 and 4, the Spearman's rank correlation coefficients are shown which are retrieved by comparing the rankings of edit rules (from the resp. the 'census-income' and 'adult' dataset) based on their λ_T under different t-norms. Those t-norms are generated by the family of Schweizer-Sklar t-norms for different parameter values. All rank correlations are statistically significant at 0.01 level. There exists a positive correlation between the rank orders of the edit rules obtained under all tested t-norms. However, we observe a clear

[1]http://archive.ics.uci.edu/ml/

	$T^{SS}_{-\infty} = T_M$	$T^{SS}_{-1} = T^H_0$	$T^{SS}_{0.5}$	$T^{SS}_0 = T_P$
T_M	1.000	0.961	0.845	0.523
T^H_0	0.961	1.000	0.940	0.546
$T^{SS}_{0.5}$	0.845	0.940	1.000	0.596
T_P	0.523	0.546	0.596	1.000

Table 3: Spearman's rank order correlation coefficients between λ_T for different Schweizer-Sklar t-norms calculated for the 21599 edit rules under T_M for $\tau = 0.04$ on the 'census-income' dataset.

	$T^{SS}_{-\infty} = T_M$	$T^{SS}_{-1} = T^H_0$	$T^{SS}_{0.5}$	$T^{SS}_0 = T_P$
T_M	1.000	0.981	0.901	0.280
T^H_0	0.981	1.000	0.955	0.315
$T^{SS}_{0.5}$	0.901	0.955	1.000	0.412
T_P	0.280	0.315	0.412	1.000

Table 4: Spearman's rank order correlation coefficients between λ_T for different Schweizer-Sklar t-norms calculated for the 16533 edit rules under T_M for $\tau = 0.1$ on the 'adult' dataset.

decrease in correlation when we shift from T_M towards T_P. These results were confirmed by more experiments using Frank t-norms and Hamacher t-norms[2]. If we recall that T_M induces the semantics of confidence and T_P induces the semantics of lift, then our results indicate that intermediary t-norms provide a mixture of confidence and lift. We can conclude that it is certainly useful to consider different t-norms for different situations where edit rules are searched in a dataset.

6.3 Execution times of the FP-tree algorithm

In a second experiment, the execution time of FBIMiner [22] is compared with the execution time of the FP-tree algorithm. In Figure 2 and Figure 3, the execution time is shown in function of both the threshold τ and the number of rules to mine. It is clear that, for a given τ value, the FBIMiner algorithm still outperforms our approach. Although this is the case, it is necessary to nuance this result. The reason is that, as Proposition 1 proves, the λ_{T_M} measure of an edit rule will always be lower than or equal to the λ_{T_P} measure of this edit rule. Therefore, when one wants to mine at least those edit rules under T_M that are also mined under T_P, the threshold value τ can be decreased such that the performance of the FP-tree algorithm improves. Besides that, the FP-tree algorithm executes the mining task of n edit rules fastest. This is an advantage because one typically tries to limit the number of edit rules to keep the result understandable.

[2]Results are omitted here due to space constraints.

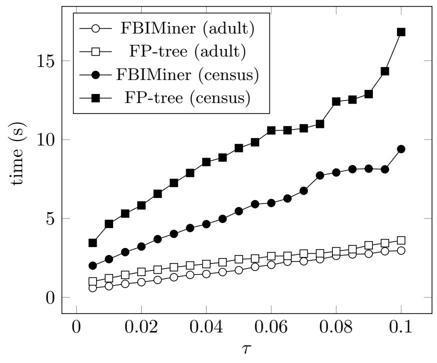

Figure 2: Runtime of FBIMiner vs. FP-tree in function of τ.

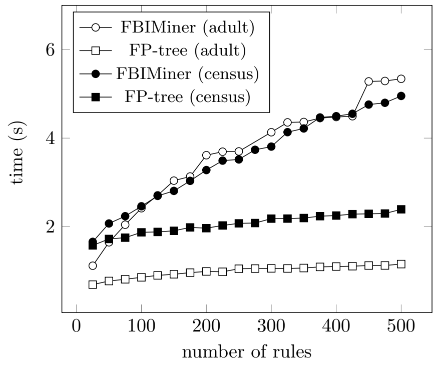

Figure 3: Runtime of FBIMiner vs. FP-tree in function of number of rules to mine.

7 Conclusion

In contrast to previous work on autonomous mining of edit rules, the definition of edit rules is generalized such that rules are generated based on different t-dependencies instead of the product t-norm. As we have shown, using different t-norms result in different set of edit rules, each with their own interpretation and remarkable properties. One particular t-norm that is investigated in detail is the minimum t-norm, T_M. The properties of T_M are exploited by an FP-tree based algorithm, used to mine edit rules under T_M. It can be said that, with the usefulness of edit rules in mind, generalizing the definition of edit rules has many advantages. In the future, the different interpretations and their impact on edit rule mining can be further examined. Further research can be done to generate the *complete* set of edit rules of D under different t-norms and to investigate in which ways data objects failing this set of edit rules can be efficiently imputed.

References

[1] S. Abiteboul, R. Hull, V. Vianu (Eds.), Foundations of Databases: The Logical Level, 1st Edi-

tion, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[2] R. Agrawal, T. Imieliński, A. Swami, Mining association rules between sets of items in large databases, SIGMOD Rec. 22 (2) (1993) 207–216.

[3] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, in: Proceedings of the 20th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc., 1994, pp. 487–499.

[4] P. Alpar, S. Winkelsträter, Assessment of data quality in accounting data with association rules, Expert Systems with Applications 41 (5) (2014) 2259–2268.

[5] C. Batini, C. Cappiello, C. Francalanci, A. Maurino, Methodologies for data quality assessment and improvement, ACM Comput. Surv. 41 (3) (2009) 16:1–16:52.

[6] C. Batini, M. Scannapieco, Data Quality: Concepts, Methodologies and Techniques (Data-Centric Systems and Applications), Springer-Verlag, Berlin, Heidelberg, 2006.

[7] P. Bohannon, W. Fan, F. Geerts, X. Jia, A. Kementsietsidis, Conditional functional dependencies for data cleaning, in: Proceedings of the 23rd International Conference on Data Engineering, 2007, pp. 746–755.

[8] A. Bronselaer, R. De Mol, G. De Tré, A measure-theoretic foundation for data quality, IEEE Transactions on Fuzzy Systems 26 (2) (2017) 627–639.

[9] F. Chiang, R. J. Miller, Discovering data quality rules, Proc. VLDB Endow. 1 (1) (2008) 1166–1177.

[10] G. De Cooman, Possibility theory, 3: possibilistic independence, International Journal of General Systems 25 (4) (1997) 353–371.

[11] W. Fan, F. Geerts, Foundations of Data Quality Management, Morgan & Claypool Publishers, 2012.

[12] W. Fan, F. Geerts, X. Jia, A. Kementsietsidis, Conditional functional dependencies for capturing data inconsistencies, ACM Transactions on Database Systems 33 (2) (2008) 6:1–6:48.

[13] W. Fan, F. Geerts, J. Li, M. Xiong, Discovering conditional functional dependencies, IEEE Transactions on Knowledge and Data Engineering 23 (5) (2011) 683–698.

[14] I. P. Fellegi, D. Holt, A systematic approach to automatic edit and imputation, Journal of the American Statistical Association 71 (353) (1976) 17–35.

[15] J. Han, J. Pei, Y. Yin, Mining frequent patterns without candidate generation, SIGMOD Rec. 29 (2) (2000) 1–12.

[16] Z. He, X. Xu, J. Z. Huang, S. Deng, Fp-outlier: frequent pattern based outlier detection, Tech. rep. (2002).

[17] Y. Huhtala, J. Krkkinen, P. Porkka, H. Toivonen, Tane: An efficient algorithm for discovering functional and approximate dependencies, The Computer Journal 42 (2) (1999) 100–111.

[18] E. P. Klement, R. Mesiar, E. Pap, Triangular Norms, 1st Edition, Springer, 2000.

[19] S. Kruse, F. Naumann, Efficient discovery of approximate dependencies, Proceedings of the VLDB Endowment 11 (7) (2018) 759–772.

[20] K. Menger, Statistical metrics, Proceedings of the National Academy of Sciences of the United States of America 28 (12) (1942) 535–537.

[21] L. L. Pipino, Y. W. Lee, R. Y. Wang, Data quality assessment, Commun. ACM 45 (4) (2002) 211–218.

[22] J. Rammelaere, F. Geerts, B. Goethals, Cleaning data with forbidden itemsets, in: 33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017, 2017, pp. 897–908.

[23] T. C. Redman, Data Quality for the Information Age, 1st Edition, Artech House, Inc., Norwood, MA, USA, 1997.

[24] A. Savasere, E. Omiecinski, S. B. Navathe, An efficient algorithm for mining association rules in large databases, in: Proceedings of the 21th International Conference on Very Large Data Bases, VLDB '95, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995, pp. 432–444.

[25] M. J. Zaki, Scalable algorithms for association mining, IEEE Transactions on Knowledge and Data Engineering 12 (3) (2000) 372–390.

[26] M. J. Zaki, W. M. Jr, Data Mining and Analysis: Fundamental Concepts and Algorithms, Cambridge University Press, New York, NY, USA, 2014.