Contrastive Antichains in Hierarchies

Anes Bendimerad
Univ Lyon, INSA, CNRS UMR 5205
F-69621 France
aabendim@liris.cnrs.fr

Jefrey Lijffijt
IDLab, Ghent University
Ghent, Belgium
jefrey.lijffijt@ugent.be

Marc Plantevit
Univ Lyon, UCBL, CNRS UMR 5205
F-69622 France
marc.plantevit@liris.cnrs.fr

Céline Robardet
Univ Lyon, INSA, CNRS UMR 5205
F-69621 France
celine.robardet@liris.cnrs.fr

Tijl De Bie
IDLab, Ghent University
Ghent, Belgium
tijl.debie@ugent.be

ABSTRACT

Concepts are often described in terms of positive integer-valued attributes that are organized in a hierarchy. For example, cities can be described in terms of how many places there are of various types (e.g. nightlife spots, residences, food venues), and these places are organized in a hierarchy (e.g. a Portuguese restaurant is a type of food venue). This hierarchy imposes particular constraints on the values of related attributes—e.g. there cannot be more Portuguese restaurants than food venues. Moreover, knowing that a city has many food venues makes it less surprising that it also has many Portuguese restaurants, and vice versa.

In the present paper, we attempt to characterize such concepts in terms of so-called *contrastive antichains*: particular kinds of subsets of their attributes and their values. We address the question of when a contrastive antichain is interesting, in the sense that it concisely describes the unique aspects of the concept, and this while duly taking into account the known attribute dependencies implied by the hierarchy. Our approach is capable of accounting for previously identified contrastive antichains, making iterative mining possible. Besides the interestingness measure, we also present an algorithm that scales well in practice, and demonstrate the usefulness of the method in an extensive empirical results section.

CCS CONCEPTS

• **Mathematics of computing** → *Information theory*; Probabilistic representations; • **Computing methodologies** → *Knowledge representation and reasoning*; *Learning in probabilistic graphical models*; Unsupervised learning.

KEYWORDS

Antichains; Subjective Interestingness; Pattern Mining.

ACM Reference Format:
Anes Bendimerad, Jefrey Lijffijt, Marc Plantevit, Céline Robardet, and Tijl De Bie. 2019. Contrastive Antichains in Hierarchies. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3292500.3330954

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08.
https://doi.org/10.1145/3292500.3330954

1 INTRODUCTION AND MOTIVATION

Humans typically understand concepts in terms of how they differ from the norm. For example, one might characterize the city of Amsterdam as a city of trams, snack places, and marihuana dispensaries. These are uniquely defining features of Amsterdam, even though it has lots of other features that are far more common (e.g. food venues) but less specific to the city. Another example would be to contrast the American with the French food culture. One might then find that popcorn and Aloe-vera-drinks are highly specific for the American food culture, with fresh foods, pies, and tabbouleh more defining for the French food culture.

In these examples, the *concepts* of interest (cities, food cultures) are described by values of particular positive integer-valued attributes, to which we will refer as *counters*. The number of facilities of various types a city has (e.g. number of bars, nightlife spots more generally, bus stops, etc.) is documented e.g. by FourSquare. The extent to which a national food culture relies on various food categories (e.g. beverages, alcoholic beverages more specifically, sugary snacks, etc.) can be found in the OpenFoodFact database.

Such data offers interesting opportunities to uniquely and objectively characterize concepts in terms of such counters. Most immediately, one could use the aggregate of these counters over all concepts, and contrast the counter of a particular concept with that average. The stronger it differs (positively or negatively), the stronger it should characterize that counter. In this way, one could obtain a set of unique aspects of each concept.

A challenge with this approach, however, is that counters are often organized in a hierarchy—a toy example is shown in Fig. 1 for illustration. Indeed, bars are examples of nightlife spots in the FourSquare taxonomy, and alcoholic beverages are particular kinds of beverages in the OpenFoodFact dataset. This means that the characterization can be highly redundant. Moreover, information about one counter can affect one's expectation about another (e.g. the fact that there are lots of bars can be explained by a high number of nightlife spots more generally), which may influence one's expectation about other counters (e.g. this may inflate one's expectation about night clubs).

Hierarchies have been extensively investigated in the literature for decades [1, 9, 16, 18]. Most studies have only considered hierarchies to restrain the pattern syntax (i.e., to avoid redundancy) or to take advantage of their subsumption power [1]. Surprisingly, there

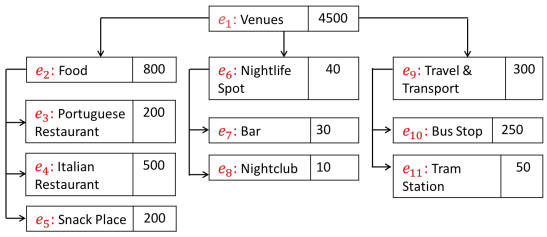

Figure 1: Example of hierarchy with counts (counts of venues in a city).

has been little discussion about the use of the hierarchical relationships for reasoning in the knowledge discovery task, especially to estimate some counters on some levels of the hierarchy knowing the values of some ancestors.

In this paper, we introduce the problem of characterizing a concept that is described in terms of a set of hierarchically-organized counters, by means of a subset of these counters with distinctive values. Hereby, distinctiveness is assessed with respect to some reference, e.g. the sum total of all concepts, or one concept in particular. To account for the dependencies between the various counters implied by the hierarchy, we propose to focus on subsets that are antichains (sets of incomparable elements) of the hierarchy, called *contrastive antichains* in this paper. These contrastive antichains pinpoint non-redundant concepts of the hierarchy that are at the same time compact and unexpected. In Figure 1, the set {Food, Bar} is an antichain, because none of these two items is a predecessor of the other one (they are not comparable). This antichain informs the user about the number of Food and Bar venues in the studied dataset. This antichain is informative if the number of Food and Bar venues are different from the expected values by the user.

The main contributions are as follows:

- Section 2 introduces and formalizes the problem of mining *contrastive antichains* in hierarchically organized sets of counters.
- Section 3 formalizes the *interestingness* of a contrastive antichain with respect to prior beliefs about the values of these counters. These prior beliefs can be derived from a concept with which the concept under investigation needs to be contrasted, or from an aggregate count over all concepts. Section 3.3 in particular shows how the measure of interestingness can take into account the knowledge of previously found contrastive antichains, such that *iteratively mined* contrastive antichains can be made non-redundant.
- Section 4 presents an *algorithm* to efficiently mine the most contrastive antichains.
- Section 5 discusses the most important related work.
- Section 6 empirically evaluates the potential usefulness of contrastive antichains as a tool to gain insight into such data, and evaluate the effectiveness and scalability of the algorithm for mining them.

2 CONTRASTIVE ANTICHAINS

Here we define the problem setting, along the way introducing the necessary concepts and notation. First we define the kind of data considered, and then the kinds of pattern we are interested in finding in such data.

2.1 The data: concepts described as sets of hierarchically related counters

As discussed in the introduction, we consider the relatively common situation where concepts of interest are defined in terms of a set of positive integer-valued attributes, whereby these attributes are organized in a hierarchy. We refer to such attributes as *counters*. Indeed, often they represent a count of something in relation to that concept, e.g. the number of food venues (the attribute), in a particular city (the concept of interest).

A particular concern in this paper is the fact that these counters are often related in a hierarchy, defined as follows.

Definition 1 (Hierarchy). *A hierarchy (or a tree) $\mathcal{H}$ is defined as a tuple $\mathcal{H} = (E, \leq, e_1)$ where:*

- *$E = \{e_1, ..., e_n\}$ is a set of items,*
- *$\leq$ is a partial order relation defined over this set,*
- *$\forall e \in E : e_1 \leq e$ (the item e_1 is called the root of $\mathcal{H}$), and*
- *there is only one path from e_1 to any other item:*

$$\forall e_i, e_j, e_k \in E : e_i \leq e_k \wedge e_j \leq e_k \implies e_i \leq e_j \vee e_j \leq e_i.$$

In Figure 1, the following relations hold: $e_1 \leq e_2$, $e_1 \leq e_3$, and $e_2 \leq e_3$. If $e_i \leq e_j$, then e_i is a predecessor of e_j, but e_i is not necessarily the direct parent of e_i. Note that a hierarchy is a special case of a partially ordered set. We assume such a hierarchy, where the items are counters, imposes certain constraints on the values these counters may have. Specifically, we assume that the value of a counter is never strictly larger than the value of a hierarchically smaller counter in the partial order.

For convenience, we introduce the following operations over a hierarchy.

Definition 2 (Basic operations). *Given $S \subseteq E$:*

- *Precedessors operator $\Uparrow$, and successors operator $\Downarrow$ as:*

$$\Uparrow S = \{e \in E \mid \exists e' \in S : e \leq e'\},$$
$$\Downarrow S = \{e \in E \mid \exists e' \in S : e' \leq e\},$$

- *Strict predecessor relation: $e_i \prec e_j \Leftrightarrow e_i \leq e_j \wedge e_i \neq e_j$,*
- *The direct successor relation $\prec$ as: $e_i \prec e_j \Longleftrightarrow \Downarrow e_i \cap \Uparrow e_j = \{e_i, e_j\}$. Also, if $e_i \prec e_j$, we use the notation $\pi_i = j$ to refer to the index of the direct parent of e_i ($e_j = e_{\pi_i}$),*
- *Direct predecessor operator $\uparrow$, and direct successors operator $\downarrow$ as:*

$$\uparrow S = \{e \in E \mid \exists e' \in S : e \prec e'\},$$
$$\downarrow S = \{e \in E \mid \exists e' \in S : e' \prec e\},$$

- *the leaves $L(\mathcal{H}) = \{e \in E \mid \nexists e' : e \prec e'\}$,*
- *The mimimum of S:* $\min(S) = \{e \in S \mid \nexists e' \in S : (e' \prec e)\}$.

The value of each counter $e_i \in E$ is denoted using the variable x_i, and when a particular empirical value is meant it will be denoted as $\hat{x}_i$. Note that this means that $\forall i \in [\![2, n]\!] : x_i \leq x_{\pi_i}$ as well as $\hat{x}_i \leq \hat{x}_{\pi_i}$.

2.2 Contrastive antichains as patterns

We aim to inform the analyst about the values of a subset of the counters of a concept. As formalized in Section 3, it is our goal to ensure that the patterns of this type are as informative to the data analyst as possible, taking into account the fact that the data analyst has certain prior beliefs about these counters, e.g. based on one or a set of other well-understood concepts with the same sets of counters. As information on a counter directly affects one's expectations about comparable (i.e. hierarchically related) counters (see Section 3 where we make this rigorous), we made the choice of including into the same pattern only non-comparable counters—i.e. no counter is a predecessor or successor of another, and the set of counters forms an *antichain* of the hierarchy. This ensures in a constructive manner that the information provided by the different counters in the pattern is not too redundant, and ensures that the patterns provide information about a larger part of the hierarchy.

For each counter $e_i \in P$, a contrastive antichain pattern informs the data analyst about the value $\hat{x}_i$. The question arises whether it is of interest to inform the analyst about its precise value. We argue that in many practical cases the precise value gives no more insight than an order of magnitude indication (although probably more detailed than orders of 10). Thus, instead of informing the analyst about the precise value, in this paper we consider the case where the pattern describes just the scale of the count: $\lfloor \log(\hat{x}_i) \rfloor$.

A contrastive antichain pattern can thus be formally defined as:

Definition 3 (Contrastive antichains). *Given a concept and the value $\hat{x}_i$ for each of a set of counters $e_i \in E$ which are organized in a hierarchy $\mathcal{H} = (E, \leq, e_1)$. A contrastive antichain pattern P is specified by a subset of the counters ($P \subseteq E$) which forms an antichain w.r.t. $\leq$, i.e., $\forall e_i, e_j \in P: e_i \leq e_j \implies e_i = e_j$, along with the integers $\lfloor \log(\hat{x}_i) \rfloor$ describing the scale of the values of these counters.*

3 THE INTERESTINGNESS OF A CONTRASTIVE ANTICHAIN

For a hierarchy $\mathcal{H} = (E, \leq, e_1)$, we aim to represent the prior beliefs of the user about the counters $e_i \in E$. This will be represented by a probability distribution Pr for the set of random variables X_i. We call this distribution the *background distribution*. In practice, the prior beliefs are derived from one or a number of example concepts and the values of their counters, as explained next.

3.1 The background distribution

In this study, we consider that the analyst has some reference that will determine her prior expectations about the counters for the concept under investigation. For instance, if we consider cities as concepts, a Londoner who would like to gain an understanding of the city of Amsterdam in terms of its counters, will typically have prior beliefs determined by the values of those counters in London. More specifically, we assume that the analyst has an expectation of the value x_i of each counter $e_i \in E$ that is equal to $\bar{x}_i$, determined by the values of the respective counter in the reference concept. In addition, we assume that the analyst has specific expectations about the value of each counter (except for e_1) relative to its parent counter's value. This can be formalized using the following two types of constraints:

- **Expectations**: the expectation of each random variable X_i is $\bar{x}_i$:
$$\forall i \in [\![1, n]\!] : \sum_{x_i} \Pr(x_i) \cdot x_i = \bar{x}_i, \tag{1}$$
- **Conditional expectations**: The expectation of the ratio of a counter's value over the parent's counter's value, *conditional on* that parent counter's value, is equal to the observed ratio of these values:
$$\forall i \in [\![2, n]\!] : \sum_{x_i} \Pr(x_i | x_{\pi_i}) \cdot \frac{x_i}{x_{\pi_i}} = \frac{\bar{x}_i}{\bar{x}_{\pi_i}}. \tag{2}$$

For $i > 1$, the second type of constraint is arguably a more accurate encoding of the analyst's prior expectations, as it explicitly encodes dependencies of counters conditioned on parent counters. For instance, for cities as concepts and the venues hierarchy, the analyst may expect that 75% of nightlife spots are bars, and knowing the number of nightlife spots will arguably affect the expectation of the number of bars. Moreover, the following property shows that the second constraint essentially subsumes the former, i.e. it is strictly stronger than the former.

Property 1. *If the two following sets of conditions hold:*

(1) ***Expectations for only the root****: $\sum_{x_1} Pr(x_1) \cdot x_1 = \bar{x}_1$,*

(2) ***Conditional expectations for other nodes****: $\forall i \in [\![2, n]\!] : \sum_{x_i} Pr(x_i | x_{\pi_i}) \cdot \frac{x_i}{x_{\pi_i}} = \frac{\bar{x}_i}{\bar{x}_{\pi_i}}$,*

Then it follows that $\forall i > 1, \sum_{x_i} Pr(x_i) \cdot x_i = \bar{x}_i$.

This property can be recursively proven, by starting from the first level (nodes e_i for which $e_{\pi_i} = e_1$), and using the fact that $\sum_{x_{\pi_i}} \Pr(x_{\pi_i}) \cdot x_{\pi_i} = \bar{x}_{\pi_i}$. The detailed proof is given in Appendix A.1 (Supplementary Materials). This means that it suffices to consider the expectation constraint for only the root, in addition to conditional expectation constraints for the other nodes.

Thus we assume that the joint distribution can be written as:

$$\Pr(x_1, x_2, \ldots, x_n) = \Pr(x_1) \prod_{i \in [\![2,n]\!]} \Pr(x_i, x_{\pi_i}). \tag{3}$$

The constraints on the probability for x_1 and the conditional probabilities of x_i conditioned on x_{π_i} do not uniquely define the joint probability distribution of all counter values. As proposed in [7], we use distributions that maximize the entropy, as any other distribution effectively makes additional assumptions that reduce the entropy. This means that these marginal distributions are found as the solutions of the following optimization problem:

- For $i = 1$ (x_i is the root):
$$\max_{\Pr(x_i)} -\sum_{x_i} \Pr(x_i) \log \Pr(x_i),$$
$$\text{s.t.} \sum_{x_i} \Pr(x_i) x_i = \bar{x}_i,$$
$$\sum_{x_i} \Pr(x_i) = 1.$$
- For $i > 1$, the distribution $\Pr(x_i | x_{\pi_i})$ needs to maximize the entropy conditional on x_{π_i}. Here, we must however recognize that the value x_i is derived from its parent counter x_{π_i} by determining which of the elements counted as part of x_{π_i} also contribute to x_i. I.e., the prior expectation is

about the tendency of any element contributing to x_{π_i} to also contribute to x_i. In terms of such decisions for individual elements, there are many ways to arrive at the same count, namely $Q(x_i, x_{\pi_i}) = \binom{x_{\pi_i}}{x_i}$ ways. Thus, we should use $Q(x_i, x_{\pi_i})$, the number of ways to realize this counter value, as a base measure, or equivalently, maximize the Kullback-Leibler divergence between the conditional probability $\Pr(x_i|x_{\pi_i})$ and $Q(x_i, x_{\pi_i})$:

$$\max_{\Pr(x_i|x_{\pi_i})} -\sum_{x_i} \Pr(x_i|x_{\pi_i}) \log \frac{\Pr(x_i|x_{\pi_i})}{Q(x_i, x_{\pi_i})},$$
$$\text{s.t.} \sum_{x_i} \Pr(x_i|x_{\pi_i}) x_i = \frac{\bar{x}_i}{\bar{x}_{\pi_i}} \cdot x_{\pi_i},$$
$$\sum_{x_i} \Pr(x_i|x_{\pi_i}) = 1.$$

The solution to the first problem is a *geometric distribution* [7] having an expectation equal to $\bar{x}_1$:

$$\Pr(x_1) = \left(1 - \frac{1}{1+\bar{x}_1}\right)^{x_1} \cdot \frac{1}{1+\bar{x}_1} = (1-p_1)^{x_1} \cdot p_1,$$

with $p_1 = \frac{1}{1+\bar{x}_1}$.

The solution to the second problem for the other random variables x_i ($i > 1$), is a *binomial distribution* with an average $\frac{\bar{x}_i}{\bar{x}_{\pi_i}} \cdot x_{\pi_i}$ [12]:

$$\Pr(x_i|x_{\pi_i}) = \binom{x_{\pi_i}}{x_i} \cdot \left(\frac{\bar{x}_i}{\bar{x}_{\pi_i}}\right)^{x_i} \cdot \left(1 - \frac{\bar{x}_i}{\bar{x}_{\pi_i}}\right)^{x_{\pi_i} - x_i} = \binom{x_{\pi_i}}{x_i} \cdot b_i^{x_i} \cdot (1-b_i)^{x_{\pi_i} - x_i},$$

where $b_i = \frac{\bar{x}_i}{\bar{x}_{\pi_i}}$ is the binomial parameter.

Property 2. *The marginal distribution for each random variable x_i is geometric, and it is:*

$$Pr(x_i) = \left(1 - \frac{1}{1+\bar{x}_i}\right)^{x_i} \cdot \frac{1}{1+\bar{x}_i}.$$

We give the proof of Property 2 in Appendix A.1 (Supplementary Materials).

3.2 The interestingness of an antichain

We use the framework of subjective interestingness *SI* proposed in [7], that defines $SI(P)$ as the ratio between the informativeness of a pattern and its description length (cost of communication to the user):

$$SI(P) = \frac{IC(P)}{DL(P)}.$$

Information content *IC*: The information carried by a pattern is quantified by the information content [7], a quantity also known as the self-information or surprisal [5]. In particular, it is equal to the reduction of uncertainty about the data caused by the knowledge of the pattern, and is defined as follows:

$$IC(P) = -\log(\Pr(P)).$$

As previously discussed, the information content of a pattern is shared with the user by transmitting the scales of the counts appearing in the pattern, instead of the exact values, which can be overwhelming and difficult to remember. Let us define the random variable $Y_i = \lfloor \log(X_i) \rfloor$, whose true value is $\hat{y}_i = \lfloor \log(\hat{x}_i) \rfloor$. The probability associated to a pattern P is thus:

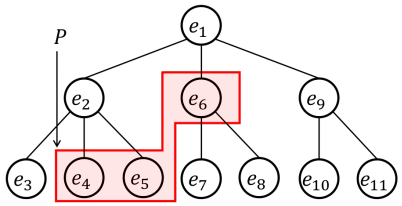

Figure 2: Example of an antichain pattern $P = \{e_4, e_5, e_6\}$

$$\Pr(P) = \prod_{e_i \in P} \Pr(\hat{y}_i) = \prod_{e_i \in P} \Pr(2^{\hat{y}_i} \leq x_i < 2^{\hat{y}_i+1}),$$
$$= \prod_{e_i \in P} \left((1-p_i)^{2^{\hat{y}_i}} - (1-p_i)^{2^{\hat{y}_i+1}}\right).$$

Description length *DL*: The description length measures the complexity of communicating a pattern P to the user. In general, the closer an item $e \in E$ is to the root, the more likely the user is to know it. For example in Foursquare hierarchy, it is simpler to communicate the node "Asian Restaurant" (2^{nd} level) than "Acehnese Restaurant" (4^{th} level) which is a successor of "Indonesian Restaurant". The idea behind the formulation of $DL(P)$ is to measure the length of the encodings of paths from the root to items $e \in P$ such that (1) these paths cover as few as possible items that are not in P, and (2) the paths are as short as possible. To construct the set of paths, we start from the root node e_1 and recursively move to the node children until reaching each item of P. Let S be the set of nodes of the paths. The cost associated to an item $e_i \in S$ is computed as follows:

(1) If e_i is an internal node of the hierarchy and the end-point of a path, its encoding cost is $\log(\alpha_1)$, with α_1 is the probability that the path "*stops*" on that node. We encode with the value $\log(1-\alpha_1)$ the fact that e_i is an internal node of a path.
(2) Let α_2 be the probability that $e_i \in P$ and $1-\alpha_2$ the probability that e_i is not in P. In the case where $e_i \in P$ we must also encode the value $\hat{y}_i$. This requires $\log(A)$ bits, where $A \geq \hat{y}_1$ is an upper bound on values $\hat{y}_i$.

By multiplying these quantities with the respective cardinalities ($|\Uparrow P \setminus P|$ for the number of internal nodes of paths and $|\downarrow(\uparrow P)|$ for the end-points of the paths), the total encoding cost of P is:

$$DL(P) = -|\Uparrow P \setminus P| \cdot \log(1-\alpha_1) - |\downarrow(\uparrow P) \setminus L(\mathcal{H})| \cdot \log(\alpha_1)$$
$$-|\downarrow(\uparrow P) \setminus P| \cdot \log(1-\alpha_2) - |P| \cdot (\log(\alpha_2) + \log(A)).$$

α_1 and α_2 are user specified parameters. If the user prefers to have patterns P with deeper nodes from the hierarchy (nodes which are closer to the leaves), she can decrease the value of α_1. On the other hand, if she likes to have patterns P with larger number of items, she can increase α_2.

In what follows, we detail, as an example, the computation of $DL(\{e_4, e_5, e_6\})$ given in Figure 2:

- Starting from the root e_1, we explore its children $\{e_2, e_6, e_9\}$. Then, we consider the node e_2 and explore its children. This requires a cost of $-2 \times \log(1-\alpha_1)$ since there are two internal nodes on these paths.

- At that time, the description covers the items $\{e_3, e_4, e_5, e_6, e_9\}$ and thus contains P. Since e_6 and e_9 are not leaves, we need to specify that we stop on them. This takes a cost of $-2\times\log(\alpha_1)$.
- We also specify that e_3, e_9 do not belong to P (with a cost of $-2\times\log(1-\alpha_2)$) and that e_4, e_5, e_6 belong to P (with a cost of $-3\times\log(\alpha_2)$). Finally, we add the cost to communicate the $\hat{y}_i$ values for each item of P (with a cost of $3\times\log(A)$).

The overal description length of P is then: $DL(P) = -2\times\log(1-\alpha_1) - 2\times\log(\alpha_1) - 2\times\log(1-\alpha_2) - 3\times(\log(\alpha_2)+\log(A))$.

3.3 Updating the background knowledge

When a rational user observes some patterns, her background knowledge may change to take into account these new pieces of information. It results that the observed patterns become expected by her and the background knowledge has to be updated as well. Let us denote by $O \subseteq E$ the set of already observed nodes. The quality of a pattern P can thus be assessed using $SI(P|O)$, the subjective interestingness of P conditioned to the already observed values from O:

$$SI(P|O) = \frac{IC(P|O)}{DL(P)} = \frac{-\log(\Pr(P|O))}{DL(P)}.$$

$\Pr(P|O)$ is the probability that P is present in the data given that the scales of items $e_i \in O$ are equal to their observed values $\hat{y}_i$:

$$\Pr(P|O) = \prod_{e_i\in P} \Pr(\hat{y}_i | \bigwedge_{e_j\in O} \hat{y}_j).$$

The conditional probabilities $\Pr(\hat{y_i}|\bigwedge_{e_j\in O}\hat{y_j})$ can be computed using the conditional constraints between each x_i and its direct ancestor x_{π_i}, as expressed by Equation 2. These dependencies between variables Y_i constitute a *Bayesian tree* (i.e., a graphical model representation [4]) which states that the joint probability between two values of the tree is independent conditionally to the value of their direct ancestor. The probability of a given state $(y_1, ..., y_n)$ is thus:

$$\Pr(y_1, ..., y_n) = \prod_{i\in[\![1,n]\!]} \Pr(y_i|y_{\pi_i}).$$

The *sum-product inference* algorithm [4] can be used to update the background model. In order to use this algorithm, we first need to derive the corresponding *factor graph*. This can be constructed by making a factor function f_1 defined on Y_1, and a factor function f_{i,π_i} for each pair (y_i,y_{π_i}). The values of these functions are: $f_1(y_1) = \Pr(y_1)$, and $f_{i,\pi_i}(y_i, y_{\pi_i}) = \Pr(y_i|y_{\pi_i})$ for each (y_i, y_{π_i}). Then, the probability of a given state $(y_1, ..., y_n)$ of this graph is:

$$\Pr(y_1, ..., y_n) = f_1(y_1)\cdot\prod_{i\in[\![2,n]\!]} f_{i,\pi_i}(y_i, y_{\pi_i}).$$

Figure 3 displays the factor graph of the hierarchy from Figure 1.

The sum-product algorithm works only when the random variables have a finite number of states, while in our case y_i take their values in $\mathbb{N}$. To make the use of sum-product possible, we can limit the values of x_i to an upper bound $A > \max(\hat{x}_1, \bar{x}_1)$. If we set $A = 2^{20}$, x_i can variate from 0 to 2^{20}, and y_i from 0 to 20. For example, if values of $\bar{x}_i$ and $\hat{x}_i$ are all lower than 2^{15}, it can be sufficient to set $A = 2^{20}$, since $Pr(y_i > 20)$ will be extremely small and negligible. Before running sum-product, we prepare the values

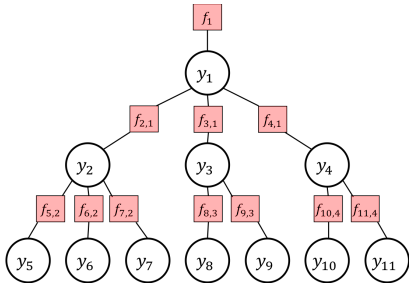

Figure 3: Factor graph of the tree in Figure 1.

of f_1 for each $y_1 \in [\![0, \log(A)]\!]$, and the value of each f_{i,π_i} for each $(y_i, y_{\pi_i}) \in [\![0, \log(A)]\!]^2$:

- For the factor f_1, the computation can be done in constant time using the CDF of the geometric distribution:

$$f_1(y_1) = \Pr(2^{y_1} \le x_1 < 2^{y_1+1}) = (1-p_1)^{2^{y_1}} - (1-p_1)^{2^{y_1+1}}.$$

- For the other factor functions, we have:

$$f_{i,\pi_i}(y_i, y_{\pi_i}) = \Pr(y_i|y_{\pi_i}) = \frac{\Pr(y_i \wedge y_{\pi_i})}{\Pr(y_{\pi_i})},$$
$$= \frac{\sum_{x_{\pi_i}=2^{y_{\pi_i}}}^{2^{y_{\pi_i}+1}-1} \Pr(x_{\pi_i})\cdot\Pr(2^{y_i} \le x_i < 2^{y_i+1}|x_{\pi_i})}{\Pr(2^{y_{\pi_i}} \le x_{\pi_i} < 2^{y_{\pi_i}+1})}.$$

And:

$$\Pr(2^{y_{\pi_i}} \le x_{\pi_i} < 2^{y_{\pi_i}+1}) = (1-p_i)^{2^{y_i}} - (1-p_i)^{2^{y_i+1}},$$
$$\Pr(2^{y_i} \le x_i < 2^{y_i+1}|x_{\pi_i}) = I_{1-b_i}(x_{\pi_i} - 2^{y_i+1} - 1, 2^{y_i+1}) - I_{1-b_i}(x_{\pi_i} - 2^{y_i} - 1, 2^{y_i}),$$

with $I_\alpha(m-k, k+1)$ is the regularized incomplete beta function which can be efficiently approximated [17].

The complexity for computing all the factor function values is in $O(|E|\cdot A\cdot\log(A))$, and the memory complexity is $O(|E|\cdot\log(A)^2)$. The computation of factor functions can be done once for the studied hierarchy, in order to be used repeatedly by sum-product each time we need to update the model. Sum-product can be applied as explained in [4]. The time complexity of this algorithm is $O(|E|\cdot\log(A)^2)$, we recall that $|E|$ is the number of random variables Y_i, and $\log(A)^2$ is the number of possible values of each pair (y_i, y_{π_i}).

4 FINDING THE MOST INTERESTING CONTRASTIVE ANTICHAINS

Finding contrastive antichains is computationally hard: the number of antichains in a rooted tree of order n is at most $2^{n-1}+1$, as shown by Klazar in [13]. Also, the interestingness of a contrastive antichain is a ratio of two measures, *IC* and *DL*, that vary unpredictably from one pattern to another and for which it is not straightforward to derive non-trivial bounds on their values to prune some uninteresting antichains. Nonetheless, we derive MICA-Miner, a heuristic algorithm based on a greedy strategy, which has a polynomial worst case complexity (i.e., $O\left(|E|^2\cdot\max(log(A)^2, |L(\mathcal{H})|)\right)$, see Appendix A.2) and whose effectiveness is demonstrated experimentally. Before running this algorithm, we consider that the factor functions

Algorithm 1: MICA-Miner

Input: $\mathcal{H}$: the hierarchical dataset, pre-computed values of factor functions f_1 and f_{i,π_i} for each $i \in [\![2, n]\!]$.
Output: $\mathcal{R}$: the antichains sorted based on iteratively updated SI.

$R \leftarrow \langle\rangle$
$O \leftarrow \emptyset$
repeat
  // Update the model with the sum-product algorithm
  // and derive the probabilities $\Pr(\hat{y}_i|O)$ for each y_i:
  Sum-product($\mathcal{H}$, O)
  // Compute an antichain with a greedy strategy:
  $P \leftarrow$ GreedySearch($\mathcal{H}$)
  if $P \neq \emptyset$ **then**
    $R.append(P)$
    $O \leftarrow O \cup P$
until $P = \emptyset$;

Algorithm 2: GreedySearch

Input: $\mathcal{H}$: the hierarchical dataset, values of conditional probabilities $\Pr(\hat{y}_i|O)$ for each y_i
Output: P: A greedily constructed antichain

$P \leftarrow \emptyset, C \leftarrow E, Q \leftarrow \emptyset$
while $C \neq \emptyset$ **do**
  $e^\star \leftarrow \text{argmax}_{e \in C} \frac{IC(P \cup \{e\}|O)}{\overline{\overline{DL}}(P \cup \{e\}, Q \cup \{e\})}$
  $f^\star \leftarrow \text{argmax}_{e \in \min(C)} \frac{IC(P|O)}{\overline{\overline{DL}}(P, Q \cup \{e\})}$
  if $\frac{IC(P \cup \{e^\star\}|O)}{\overline{\overline{DL}}(P \cup \{e^\star\}, Q \cup \{e^\star\})} \geq \frac{IC(P|O)}{\overline{\overline{DL}}(P, Q \cup \{f^\star\})}$ **then**
    $P \leftarrow P \cup \{e^\star\}$
    $Q \leftarrow Q \cup \{e^\star\}$
    $C \leftarrow \{e \in C \mid e \not\preceq e^\star \wedge e^\star \not\preceq e\}$
  else
    $Q \leftarrow Q \cup \{f^\star\}$
    $C \leftarrow \{e \in C \mid f^\star \not\preceq e\}$
return P

f_1 and f_{i,π_i} (for each $i \in [\![2, n]\!]$) are already computed based on the approach explained in Section 3.3.

MICA-Miner is an iterative algorithm that, at each iteration, updates the model using the sum-product algorithm for integrating the previously observed nodes O. It then produces an antichain P based on GreedySearch. MICA-Miner continues until GreedySearch returns an empty pattern, which means that $IC(e) = 0\ \forall e \in E$.

In order to greedily build an antichain, GreedySearch starts from an empty pattern $P = \emptyset$ and a set of candidates $C = E$. It also uses a set Q that contains the current paths endpoints, i.e., Q contains the already known part of $\downarrow(\uparrow P)$, and Q will be equal to $\downarrow(\uparrow P)$ at the end of GreedySearch. In each step, GreedySearch chooses an item that will be added to Q and possibly to P, and removed from C (as well as its predecessors and successors). In order to decide which item to pick in each step, GreedySearch uses a heuristic $\frac{IC}{\overline{DL}}$ where $\overline{DL}$ is defined for an antichain P and a set Q s.t. $P \subseteq Q \subseteq \downarrow(\uparrow P)$. $\overline{DL}$ is similar to DL, the difference is that $\overline{DL}$ does not account for the cost of all the paths endpoints $\downarrow(\uparrow P)$, but only for Q, the already known part of the final $\downarrow(\uparrow P)$:

$$\overline{DL}(P, Q) = -|\Uparrow P \setminus P| \cdot \log(1-\alpha_1) - |Q \setminus L(\mathcal{H})| \cdot \log(\alpha_1) - |Q \setminus P| \cdot \log(1-\alpha_2) - |P| \cdot (\log(\alpha_2) + \log(A)).$$

At each iteration, GreedySearch selects two items: $e^\star \in C$ and $f^\star \in \min(C)$ such that regarding to $\frac{IC}{\overline{\overline{DL}}}$: (1) $e^\star$ is the best item to add to P , (2) $f^\star$ is the best item to add to $Q \setminus P$ (i.e., $f^\star$ is the best candidate to remove from C without add it to P). From $e^\star$ and $f^\star$, GreedySearch chooses the one that has the highest $\frac{IC}{\overline{\overline{DL}}}$. If $e^\star$ is chosen, P is extended by this item. Otherwise, $f^\star$ and all its successors are removed from the set of candidates. When C becomes empty, the antichain P is returned.

5 RELATED WORK

The use of hierarchies (ontologies, taxonomies, etc.) in KDD tasks has been extensively studied. In [1], the authors explained the high and promissing utility of ontologies in different steps of the KDD process, and proposed an association mining tool that benefits from ontologies in different stages of the mining process (data understanding, task design, etc.). The generic problem of Semantic Data Mining has been defined in [16, 18]: given a set of objects annotated with ontology terms, the goal is to find hypothesis, expressed by domain ontology terms, explaining the given empirical data. Specifically in [15, 18, 19], the Semantic Subgroup Discovery problem is studied: given a dataset where each object is annotated with ontology terms and belongs to a specific class, the goal is to find a conjunction of ontology terms (a conjunctive rule) that corresponds to a set of object discriminating a specific class. To evaluate the discriminativity of a conjunctive rule, these approaches mainly use the $wWRAcc$ heuristic based on the $WRAcc$[1] measure. This heuristic has been initially proposed for the generic task of Subgroup Discovery [14]. A similar method has been proposed in [1] to take benefit from ontology structures in order to optimize and reduce the redundancy in the search of conjunctive rules that satisfy some user specified constraints. In [2], this problem of rule learning is tackled based on ILP (Inductive Logic Programming), and ontology structures are integrated by providing additional clauses to the ILP solver. Enumerating hierarchical attributes has been also integrated in the Exceptional Model Mining problem [3]. The common point between these works is the use of ontologies in the rule discovery to provide conjunctive clauses which cover a part of data that satisfy some constraints. In our work, the goal is to characterize the dataset based on an antichain that does not necessarily cover the same part of the data, but provides a good characterization of the overal dataset. Another specificity of our work is the explicit use of hierarchies to represent user background knowledge and integrate it in the interestingness measure.

Several works have also used hierarchies to improve the prediction task. In [11], the authors propose an algorithms that, given a training set of objects described with ontology concepts, allows to select an antichain whose nodes will be used as features to learn a predictive model. Other works proposed different ontology-aware classification approaches [8, 20, 21] based on well known models

[1] $WRAcc$: Weighted Relative Accuracy measure

(decision trees, bayesian networks). While these works uses ontology structures to provide better prediction models, our approach aims to provide knowledge about a dataset to a specific user. Ontology structure has been used in other KDD tasks (ontology-based clustering, information extraction, recommender systems, etc.). In the survey [9] a large spectrum of these works is reviewed.

In this paper, we use an interestingness measure inspired by the FORSIED framework [6, 7], which defines the SI of a pattern as the ratio between the IC and the DL. The IC is the amount of information contained in the pattern. The quantification is based on the gain from a Maximum Entropy background model that depicts the current knowledge of a user, hence it is subjective, i.e., particular to the modeled belief state. Regarding the previous studies based on FORSIED framework, the novelty of our work is the incorporation of dependencies between user expectations about different attributes, using available hierarchy structures of the data.

6 EMPIRICAL RESULTS

In this section, we report our experimental results. These experiments aim to evaluate the performance and the quality of results provided by MICA-Miner in real world datasets. The method was implemented in Java and the experiments were performed on a machine equipped with Intel(R) Xeon(R) CPU @ 4.00GHz, and 128GB main memory, running Debian GNU/Linux 9.6. The code and the data are available[2].

Datasets and aims. We conduct experiments in three real world datasets whose main characteristics are given in Table 1:

- **FV**: Foursquare venues [10], the nodes e_i in this hierarchical dataset are categories of venues[3] (food, restaurant, event, etc.). We aim to study the distribution of venue categories in Amsterdam and London, and discover which kind of categories are surprisingly frequent/rare in each of these cities. We then consider two case studies: (1) **FV-Ams** for Amsterdam: the observed values $\hat{x}_i$ are the counters of categories in Amsterdam, (2) **FV-Lon** for London: the observed values $\hat{x}_i$ are the counters of categories in London. For both cases, the priors $\bar{x}_i$ are the aggregation of categories counters throughout 20 well known cities from Europe and USA (London, Barcelona, New York, etc.).
- **OFF**: Open Food Facts[4] is a free food product database that gathers information and data on food products around the world. Each product is described by its food category (Plant-based food, ...) and its country. The categories represent the nodes e_i of the hierarchy. We want to compare the distribution of food categories between France and USA, the two countries with the most contributions. Then, the values $\hat{x}_i$ (resp. $\bar{x}_i$) are the number of products for each category in France (resp. USA).
- **EP_Abst**: European Parliament[5] is a dataset that depicts ballots of the European Parliament between 2014 and 2019. Each ballot is described by the corresponding topics and the votes (for, against, abstain) of each deputy, whose political groups are given. Topics are organized following a hierarchy (e.g., the topic "5.10 Economic Union" is the strict predecessor of "5.10.02 Price policy, price stabilisation"). We build a hierarchical dataset for each political group where the values $\hat{x}_i$ correspond to the number of abstentions: the number of times a deputy from this group abstains in the topic e_i. The priors $\bar{x}_i$ are constructed from the total number of ballots by topic.

[2]https://tinyurl.com/y2jkmduv
[3]https://developer.foursquare.com/docs/resources/categories
[4]https://world.openfoodfacts.org/
[5]http://parltrack.euwiki.org/

Table 1: Description of the real-world datasets.

Datasets	$\|E\|$	$\|\hat{x}_1\|$	# levels	branching factor
FV-Ams	846	9030	5	13
FV-Lon	846	25220	5	13
OFF	2000	147,411	10	3.6
EP_Abs	358	[7470; 30, 673]	5	5.17

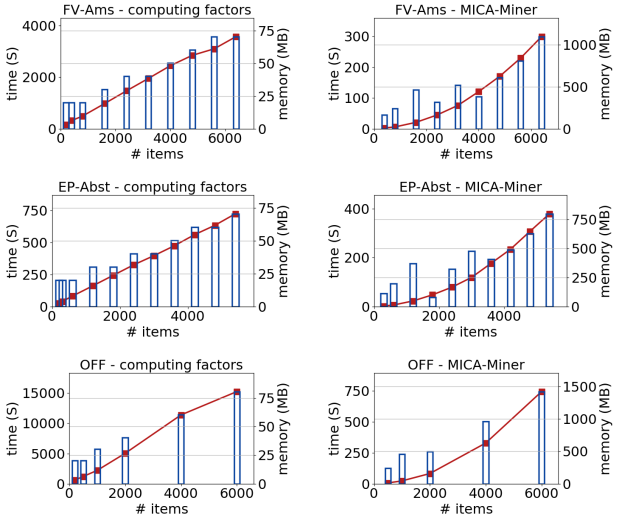

Figure 4: Time (line) and memory consumption (bars) of the factor functions computation (columns 1), and MICA-Miner (columns 2) in the studied datasets w.r.t. $|E|$.

In this experimental study, we aim to answer the following questions: How much time and memory does MICA-Miner consume in practice? How does the size of the hierarchy $|E|$ impact the performance of MICA-Miner? What is the impact of model updating on the found antichains? Are the antichains found by MICA-Miner relevant and informative?

Quantitative evaluation. We study the runtime and memory consumption for (1) the computation of factor functions (see Section 3.3), and (2) MICA-Miner. We vary the number of items in the real world datasets as follows: in order to decrease the number of items, we iteratively remove leaves that are the furthest from the root, and in order to increase the number of items, we duplicate the tree as many times as we need and then we add a new root that links all the duplications. Results are given in Figure 5, column 1 for the factor functions, and column 2 for MICA-Miner. As it was theoretically expected, the computation of factors requires a time and memory that linearly increase with the number of items. Moreover, the time increases for MICA-Miner in a quadratic manner (a fitted model $a \cdot |E|^2 + b$ approximates the time with an average square error below $7(s^2)$), its memory consumption also follows the same quadratic trend.

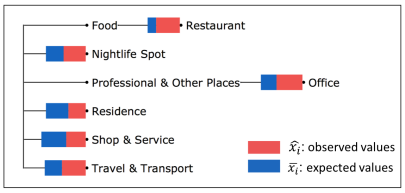

Figure 5: Top 1 pattern based on *WRAcc* measure FV-Ams.

Comparative evaluation. There is no approach in the literature that supports the discovery of subjectively interesting antichains in hierarchies. Nevertheless, we identify some baselines that we consider in our study to highlight the characteristics of the patterns found with MICA-Miner: (1) *SI without update* which returns the best results according to the *SI* measure without updating the background knowledge model; (2) *WRAcc* that uses the *WRAcc* measure [14]. This measure has been largely used in Subgroup Discovery in order to evaluate the prevalence of a given class of objects in a specific subset of data (a subgroup). We adapted it to our problem in order to evaluate the prevalence/absence of nodes $e \in P$ in the studied dataset. We consider that a subgroup is defined with the antichain P, the size of positive class is $\hat{x}_i$, and the size of negative class is $\bar{x}_i$. We compare the following properties computed in the top k patterns returned in each configuration:

- *Average normalized values of counters*: we show the average values of $\frac{\hat{x}_i}{\hat{x}_1}$ corresponding to nodes in the top patterns.
- *Average contrast*: we want to evaluate how much the values of $\hat{x}_i$ are contrastive comparing with their corresponding $\bar{x}_i$. We propose to use the following contrast measure defined for an antichain $P \subseteq E$: $contrast(P) = \sum_{e_i \in P} \frac{max(\hat{x}_i, \bar{x}_i) - min(\hat{x}_i, \bar{x}_i)}{max(\hat{x}_i, \bar{x}_i)}$.
- *Redundancy*: An important goal of the iterative updating of the model is to avoid communicating to the user similar information several times. For example, after informing the user that e_i ="Restaurant" is prevalent, it is less informative to tell her that e_{π_i} ="Food" is also prevalent. We aim to measure this kind of redundancy between patterns returned by each approach based on the following measure defined for a set of patterns R:

$$redund(R) = \frac{\sum_{P,P' \in R, P \neq P'} |\{e_i \in P | \exists e_{\pi_i} \in P' : (\hat{x}_i - \bar{x}_i) \cdot (\hat{x}_{\pi_i} - \bar{x}_{\pi_i}) \geq 0\}|}{|R| \times \sum_{P \in R} |P|}.$$

Figure 6 reports the values of these properties in each of the three configurations (1) *SI* (our approach), (2) *WRAcc* measure, and (3) *SI* without update. These properties are computed based on the top 5 patterns in **FV-Ams** and **EP-Abst**, and on top 20 in **OFF** (since it is a much larger hierarchy). The average values of counters in *WRAcc* results are significantly large, its top patterns generally contains nodes with the highest counters in the dataset. However, its average contrast is remarkably low. Figure 5 shows the first pattern found by *WRAcc* in **FV-Ams**. Most of its nodes are not significantly contrastive, i.e., there is not a high difference between the observed value and the expected value. We can also notice that the values of *redund* are the lowest for the results of *SI*, and those of the *WRAcc* have the highest *redund*. The value of *redund* for *SI* is clearly lower than its value for "*SI* without update" in **FV-Ams** and **EP-Abst**. The difference is not remarkable for the **OFF** dataset, indeed, since it is a larger dataset, there is less chance to have redundancies in the results. To sum up, our method allows to discover more contrastive antichains than methods that do not take into account the prior beliefs. Furthermore, updating the background knowledge at each iteration makes it possible to provide less redundant results.

Illustrative results. We report in Figure 7 the top 3 contrastive antichains discovered by MICA-Miner in **FV-Ams**, **FV-Lon**, and **OFF** datasets. The blue color quantifies the expected value by the user (based on her beliefs over 20 cities), and the red color is the observed counter in Amsterdam (resp. London). The first antichain in Amsterdam informs that American Restaurants, Residential Buildings, and Airports are much less frequent than expected, while Marijuana Dispensary and Tram Station are significantly over-expressed. This last type of venue is indeed characteristic of Amsterdam, because it is authorized in this city while it is generally illegal in other cities. London is characterized with a high number of Fish & Chip Shops, Portuguese Restaurants, and Pubs, and an under-expression of Residences and Airports. We point out that Airport venues can correspond to any place related to Airport (Airport Food Court, Airport Gate, Airport Tram, etc.). After assimilating this first antichain, the user will change her expectations about the rest of the data. For example, she may assume that in general there is a high presence of Food venues and Bars in London, which is also incorporated in our model using the conditional expectation. Due to this hypothesis, the second antichain will notify the user that despite of the high observed values in the first antichain, there is some specific types of Food venues and Bars that are under expressed (American Restaurant, Donut Shop, etc.). For **OFF** dataset, the Red color quantifies the observed counts of products from France, and blue color corresponds to the expected values derived from products distributions from USA. The first antichain can be interpreted as: *From an American point of view, the French food is characterized with a high number of fresh meals, refrigerated meals, fresh foods, and a low presence of groceries and popcorn products.*

7 CONCLUSION

We have introduced the novel problem of mining contrastive antichains in hierarchically organized sets of counters. Prior beliefs are used to assess how contrastive an antichain is. Furthermore, the hierarchical relations are fully exploited to both propagate the beliefs and to update the background knowledge . We have proposed a greedy algorithm that efficiently and iteratively returns the most contrastive antichains. Extensive empirical results on several real-world datasets confirm that the contrastive antichains are intuitive and they capture insights that cannot be done with other interestingness measure or without updating the background knowledge. This paper opens up several avenues for research such as the integration of this model into the subgroup discovery / exceptional model mining framework.

Acknowledgements. This work was supported by the ERC under the EU's Seventh Framework Programme (FP7/2007-2013) / ERC Grant Agreement no. 615517, FWO (project no. G091017N, G0F9816N), the EU's Horizon 2020 research and innovation programme and the FWO under the Marie Skłodowska-Curie Grant Agreement no. 665501, Group Image Mining (GIM) which joins researchers of THALES Group and LIRIS Lab, and by the ACADEMICS grant of the IDEXLYON, project of the Université of Lyon, PIA operated by ANR-16-IDEX-0005.

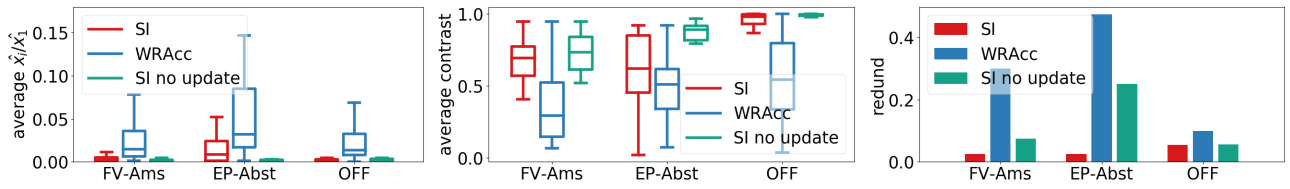

Figure 6: Comparison of (1) normalized supports, (2) contrast, (3) redundancy between the top antichains of the three measures: *SI*, *WRAcc*, and *SI* without model updating.

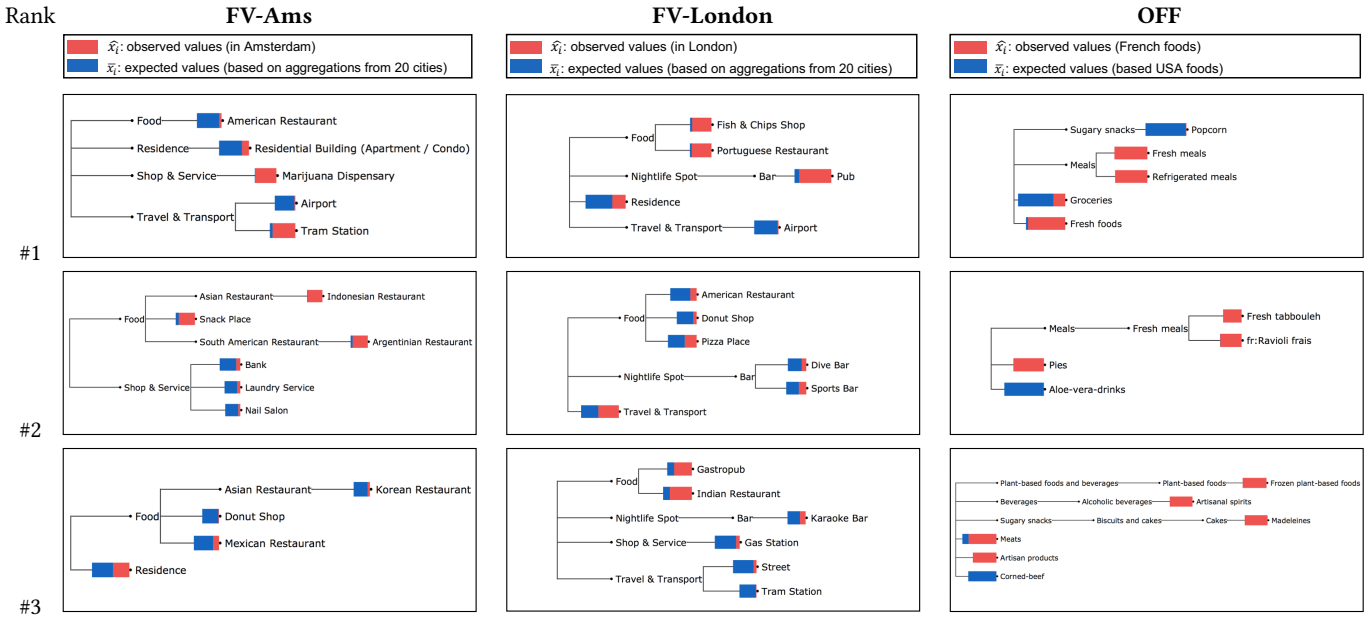

Figure 7: Top 3 antichains in Amsterdam, London (based on the prior beliefs over 20 cities) and Open Food Fact dataset (comparison of French products and USA products).

REFERENCES

[1] John M. Aronis, Foster J. Provost, and Bruce G. Buchanan. 1996. Exploiting Background Knowledge in Automated Discovery. In *KDD*. 355–358.

[2] Tim De Beéck, Arjen Hommersom, Jan Van Haaren, Maarten van der Heijden, Jesse Davis, Peter Lucas, Lucy Overbeek, and Iris Nagtegaal. 2015. Mining Hierarchical Pathology Data Using Inductive Logic Programming. In *AIME*. 76–85.

[3] Adnene Belfodil, Sylvie Cazalens, Philippe Lamarre, and Marc Plantevit. 2017. Flash Points: Discovering Exceptional Pairwise Behaviors in Vote or Rating Data. In *ECMLPKDD*. 442–458.

[4] Christopher M. Bishop. 2007. *Pattern Recognition and Machine Learning*. 359–422 pages.

[5] Thomas M Cover and Joy A Thomas. 1991. Entropy, relative entropy and mutual information. *Elements of information theory* 2 (1991), 1–55.

[6] Tijl De Bie. 2011. An information theoretic framework for data mining. In *KDD*. 564–572.

[7] Tijl De Bie. 2011. Maximum entropy models and subjective interestingness. *Data Mining and Knowledge Discovery* 23, 3 (2011), 407–446.

[8] Marie desJardins, Lise Getoor, and Daphne Koller. 2000. Using Feature Hierarchies in Bayesian Network Learning. In *Abstraction, Reformulation, and Approximation, 4th International Symposium, SARA 2000*. 260–270.

[9] Dejing Dou, Hao Wang, and Haishan Liu. 2015. Semantic data mining: A survey of ontology-based approaches. In *ICSC*. 244–251.

[10] Géraud Le Falher, Aristides Gionis, and Michael Mathioudakis. 2015. Where Is the Soho of Rome? Measures and Algorithms for Finding Similar Neighborhoods in Cities. In *ICWSM*. 228–237.

[11] Gemma C. Garriga, Antti Ukkonen, and Heikki Mannila. 2008. Feature Selection in Taxonomies with Applications to Paleontology. In *Discovery Science*. 112–123.

[12] Peter Harremoës. 2001. Binomial and Poisson distributions as maximum entropy distributions. *IEEE Trans. Information Theory* 47, 5 (2001), 2039–2041.

[13] M. Klazar. 1997. Twelve countings with rooted plane trees. *European J. Combin.* 18, 2 (1997), 195–210.

[14] Nada Lavrac, Branko Kavsek, Peter A. Flach, and Ljupco Todorovski. 2004. Subgroup Discovery with CN2-SD. *JMLR* 5 (2004), 153–188.

[15] Nada Lavrac, Anze Vavpetic, Larisa N. Soldatova, Igor Trajkovski, and Petra Kralj Novak. 2011. Using Ontologies in Semantic Data Mining with SEGS and g-SEGS. In *Discovery Science*. 165–178.

[16] Haishan Liu. 2010. Towards semantic data mining. In *ISWC*. 7–11.

[17] David M. Smith. 2001. Algorithm 814: Fortran 90 software for floating-point multiple precision arithmetic, gamma and related functions. *ACM Trans. Math. Softw.* (2001), 377–387.

[18] Anze Vavpetic and Nada Lavrac. 2013. Semantic Subgroup Discovery Systems and Workflows in the SDM-Toolkit. *Comput. J.* 56, 3 (2013), 304–320.

[19] Anze Vavpetic, Petra Kralj Novak, Miha Grcar, Igor Mozetic, and Nada Lavrac. 2013. Semantic Data Mining of Financial News Articles. In *Discovery Science*. 294–307.

[20] Jun Zhang, Doina Caragea, and Vasant G. Honavar. 2005. Learning Ontology-Aware Classifiers. In *Discovery Science*. 308–321.

[21] Jun Zhang, Dae-Ki Kang, Adrian Silvescu, and Vasant G. Honavar. 2006. Learning accurate and concise naïve Bayes classifiers from attribute value taxonomies and data. *Knowl. Inf. Syst.* 9, 2 (2006), 157–179.

A SUPPLEMENTARY MATERIALS

A.1 Proofs

Proof of Property 1. We consider that $\sum_{x_{\pi_i}} \Pr(x_{\pi_i}) \cdot x_{\pi_i} = \bar{x}_{\pi_i}$ and we prove that $\sum_{x_i} \Pr(x_i) \cdot x_i = \bar{x}_i$. The complete proof can be recursively established by starting from the root ($x_{\pi_i} = x_1$) for which we already know that $\sum_{x_1} \Pr(x_1) \cdot x_1 = \bar{x}_1$.

$$\begin{aligned}
\sum_{x_i} \Pr(x_i) \cdot x_i &= \sum_{x_i} \left(\sum_{x_{\pi_i}} \Pr(x_{\pi_i}) \cdot \Pr(x_i|x_{\pi_i}) \right) \cdot x_i, \\
&= \sum_{x_i} \sum_{x_{\pi_i}} \left(\Pr(x_{\pi_i}) \cdot \Pr(x_i|x_{\pi_i}) \cdot x_i \right), \\
&= \sum_{x_{\pi_i}} \sum_{x_i} \left(\Pr(x_{\pi_i}) \cdot \Pr(x_i|x_{\pi_i}) \cdot x_i \right), \\
&= \sum_{x_{\pi_i}} \Pr(x_{\pi_i}) \cdot \sum_{x_i} \left(\Pr(x_i|x_{\pi_i}) \cdot x_i \right), \\
&= \sum_{x_{\pi_i}} \Pr(x_{\pi_i}) \cdot \frac{\bar{x}_i}{\bar{x}_{\pi_i}} \cdot x_{\pi_i}, \\
&= \frac{\bar{x}_i}{\bar{x}_{\pi_i}} \cdot \sum_{x_{\pi_i}} \Pr(x_{\pi_i}) \cdot x_{\pi_i}, \\
&= \frac{\bar{x}_i}{\bar{x}_{\pi_i}} \cdot \bar{x}_{\pi_i}, \\
&= \bar{x}_i.
\end{aligned}$$

□

Proof of Property 2. This property states that the marginal distribution of each random variable x_i is geometric and it is:

$$\Pr(x_i) = (1 - \frac{1}{1+\bar{x}_i})^{x_i} \cdot \frac{1}{1+\bar{x}_i}.$$

For the root x_1, this is already the case. We then prove this property by recursion, we consider that x_{π_i} follows a geometric distribution with parameter $p_{\pi_i} = \frac{1}{1+\bar{x}_{\pi_i}}$ and we prove that x_i also follows a geometric distribution with a parameter $p_i = \frac{1}{1+\bar{x}_i}$:

$$\Pr(x_i) = \sum_{x_{\pi_i}=x_i}^{+\infty} \Pr(x_{\pi_i}) \cdot \Pr(x_i|x_{\pi_i}), \tag{4}$$

$$= \sum_{x_{\pi_i}=x_i}^{+\infty} (1-p_{\pi_i})^{x_{\pi_i}} \cdot p_{\pi_i} \cdot \binom{x_{\pi_i}}{x_i} \cdot b_i^{x_i} \cdot (1-b_i)^{x_{\pi_i}-x_i}, \tag{5}$$

$$= \frac{p_{\pi_i} \cdot b_i^{x_i}}{(1-b_i)^{x_i}} \cdot \sum_{x_{\pi_i}=x_i}^{+\infty} \binom{x_{\pi_i}}{x_i} \left((1-p_{\pi_i})(1-b_i)\right)^{x_{\pi_i}}, \tag{6}$$

$$= \frac{p_{\pi_i} \cdot b_i^{x_i}}{(1-b_i)^{x_i}} \cdot \sum_{k=0}^{+\infty} \binom{k+x_i}{x_i} \left((1-p_{\pi_i})(1-b_i)\right)^{k+x_i}, \tag{7}$$

$$= \frac{p_{\pi_i} \cdot b_i^{x_i}}{(1-b_i)^{x_i}} \cdot \left((1-p_{\pi_i})(1-b_i)\right)^{x_i} \cdot \sum_{k=0}^{+\infty} \binom{k+x_i}{x_i} \left((1-p_{\pi_i})(1-b_i)\right)^{k}, \tag{8}$$

$$= p_{\pi_i} \cdot b_i^{x_i} \cdot (1-p_{\pi_i})^{x_i} \cdot \sum_{k=0}^{+\infty} \binom{k+x_i}{k} \left((1-p_{\pi_i})(1-b_i)\right)^{k}, \tag{9}$$

The negative binomial infinite series can be simplified:

$$\begin{aligned}
\sum_{k=0}^{+\infty} \binom{k+x_i}{k} \left((1-p_{\pi_i})(1-b_i)\right)^{k} &= \left(1-(1-p_{\pi_i})(1-b_i)\right)^{-1-x_i}, \\
&= (b_i + p_{\pi_i} - b_i \cdot p_{\pi_i})^{-1-x_i},
\end{aligned}$$

then:

$$\Pr(x_i) = p_{\pi_i} \cdot b_i^{x_i} \cdot (1-p_{\pi_i})^{x_i} \cdot (b_i + p_{\pi_i} - b_i \cdot p_{\pi_i})^{-1-x_i}, \tag{10}$$

$$= \frac{p_{\pi_i}}{b_i + p_{\pi_i} - b_i \cdot p_{\pi_i}} \cdot (1 - \frac{p_{\pi_i}}{b_i + p_{\pi_i} - b_i \cdot p_{\pi_i}})^{x_i}. \tag{11}$$

After substituting $p_{\pi_i} = \frac{1}{1+\bar{x}_{\pi_i}}$ and $b_i = \frac{\bar{x}_i}{\bar{x}_{\pi_i}}$, we find that $\frac{p_{\pi_i}}{b_i + p_{\pi_i} - b_i \cdot p_{\pi_i}} = \frac{1}{1+\bar{x}_i}$. This means that:

$$\Pr(x_i) = (1 - \frac{1}{1+\bar{x}_i})^{x_i} \cdot \frac{1}{1+\bar{x}_i}.$$

This concludes the proof.

□

A.2 Theoretical complexity of MICA-Miner

We derive the worst case complexity of MICA-Miner, which depends on the complexity of sum-product and GreedySearch. As stated in Section 3.3, sum-product has a worst case complexity of $O(|E| \cdot log(A)^2)$ where $|E|$ is the number of items, and the parameter $A > max(\hat{x}_1, \bar{x}_1)$ is the considered number of possible values of x_i. In GreedySearch, each iteration costs at most $O(|E|)$, and the maximum number of iterations is equal to $|L(\mathcal{H})|$ which is the size of the largest antichain. Thus, GreedySearch has a worst case complexity of $O(|E| \cdot |L(\mathcal{H})|)$. Since the maximum number of iterations in MICA-Miner is $|E|$, we conclude that its worst case complexity is $O\left(|E|^2 \cdot \max(log(A)^2, |L(\mathcal{H})|)\right)$.

A.3 Supplementary experiments

Quantitative evaluation. We aim to study the influence of the parameter A on the runtime and memory consumption. A is an upper bound on the value of variables x_i, it needs to cover the used values $\bar{x}_i$ and $\hat{x}_i$, this means that $A \geq max(\hat{x}_i, \bar{x}_i)$. Figure 8 reports the results. The runtime grows linearly w.r.t A for the preprocessing step while the increasing of A does not significantly impact the runtime of MICA-Miner. The memory consumption slightly increases in general w.r.t. A in both cases.

A.4 Code and data

All code and data are available at https://tinyurl.com/y2jkmduv.

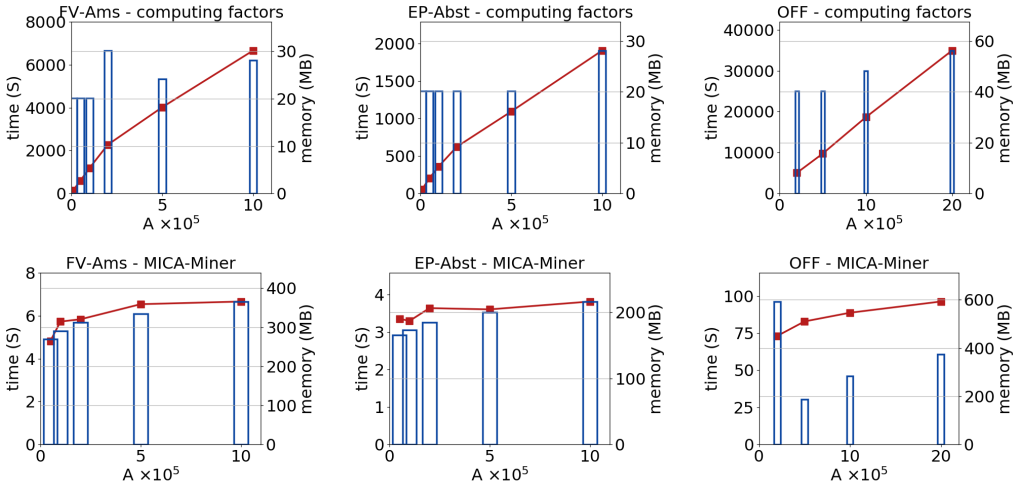

Figure 8: Supplementary experiments. Time (line) and memory consumption (bars) of the factor functions computation (row 1), and MICA-Miner (row 2) in the studied datasets w.r.t. A.