Multi-instance learning of pretopological spaces to model complex propagation phenomena: Application to lexical taxonomy learning

Gaëtan Caillaut, Guillaume Cleuziou

Version of Record: https://www.sciencedirect.com/science/article/pii/S0004370221001077
Manuscript_889ab71bc5785738b4ed906790273c42

Multi-instance Learning of Pretopological Spaces to Model Complex Propagation Phenomena: Application to Lexical Taxonomy Learning

G. Caillaut[a,b,*], G. Cleuziou[a,c]

[a] *University of Orléans, INSA Centre Val de Loire, LIFO EA 4022, France*
[b] *Le Mans University, LIUM EA 4023, France*
[c] *University of New Caledonia, ISEA, New Caledonia*

Abstract

This paper addresses the problem of learning the concept of *propagation* in the theoretical formalism of pretopology, and then applying this methodology for the well-known problem of learning Lexical Taxonomy. The theory of pretopology, among others, aims at modeling complex relations between sets of entities. The use of such fine-grained modeling implies limitations in terms of scalability. However, it allows for a more accurate capture of real-world relationships, such as the hypernymy relation, by modeling the task of relation extraction as a propagation model under certain structuring constraints, as opposed to traditional approaches that are limited to detecting relations between pairs of elements without considering knowledge on the expected structuring.

Our proposal is to define the pseudo-closure operator (modeling the concept of propagation) as a logical combination of heterogeneous neighborhoods, or sources. It allows the learning of models that exploit, for example, the knowledge acquired by both statistical and numerical approaches. We show that the learning of such an operator falls into the Multiple Instance (MI) framework, where the learning process is performed on bags of instances instead of individual instances. Although this framework is well suited for this task, using it for learning a pretopological space leads to a set of bags whose size is exponential. To overcome this problem, we propose a learning method (LPSMI) based on a low estimate of the bags covered by a concept under construction.

We first propose an experimental validation of our method, through the simulation of percolation processes (typically forest fires) learned with pretopological propagation models. It reveals that the proposed MI approach is particularly efficient on propagation model recognition task. We then provide a real-world contribution to the Lexical Taxonomy learning task, by modeling this task as a complex (semantic) propagation problem. We propose a very generic frame-

*Corresponding author
Email addresses: gaetan.caillaut@univ-lemans.fr (G. Caillaut), guillaume.cleuziou@unc.nc (G. Cleuziou)

work for training models combining various existing methods for learning Lexical Taxonomies (statistical, pattern-based and embedding-based).

Keywords: Multi-instance learning, Supervised learning, Pretopology, Complex propagation, Percolation, Lexical taxonomy

---

1. Introduction

Complex systems and complex networks have received increasing attention for several years. Such networks are used to model complex interactions between entities in a wide spectrum of domains such as organic ecosystems [49, 61], air pollution [8], energy management [5], world economy [33, 35] and of course with uses in artificial intelligence domains such as social networks analysis [50] or text mining [27] to name a few. All of these topics are difficult to model using standard tools but are essential to human development.

Complex systems can be approached using different theories, such as fuzzy logic [75], graph theory [63, 41], rough set theory [74] and pretopology [6] as major theoretical frameworks. The present work focuses on the theory of pretopology, known for its ability to finely model complex propagation phenomena. This theory is based on a propagation operator called *pseudo-closure* which forms the core of the theory of pretopology. It is usually defined on a collection of neighborhoods and subtly combines them. This powerful operator offers a formalism capable of describing, step by step, a non-trivial expansion process, such as coalition formation in game theory [9], the propagation of an epidemic [52] or the propagation of semantic relations in semantic networks [28]. The last application area is based on very challenging AI tasks, among which the exploitation problem [66] or the automatic or semi-automatic learning of Lexical Taxonomies (LT) from text corpora [25, 69].

Most of the work related to pretopology relies on a handcrafted pseudo-closure operator [3, 71, 20, 27]. The pseudo-closure operator is often defined as the conjunction of all neighborhoods. The problem is that such an operator limits us to a single pretopological space that may not be suitable for every application case. Moreover, in the common definition of the pseudo-closure operator, the same "credibility" and the same "utility" are given to each neighborhood. In applications where data are collected from multiple sources (or neighborhoods), this assumption is likely to be false.

Recent work addresses this problem by proposing more flexible solutions. Considering that a simple pretopological space cannot model all complex systems, Bui et al. [21] introduce the concepts of *weak* and *strong* pseudo-closure operators leading to *thick* (i.e. noisy) and *thin* pretopological spaces, respectively. At the same time, Galindo et al. [38] define a set of pseudo-closure operators, each referring to slightly different (topological) neighborhoods as a function of a parameter r such that the smaller r, the thinnest the resulting pretopological space is.

But these approaches to extending pretopology theory remain limited. For example, there is no way to ignore or limit the impact of erroneous neighborhoods in favor of more reliable ones. Existing work proposes to use multiple predefined pretopological spaces, but this does not guarantee that any one of these spaces, or even a combination of these spaces [29], is suitable for the application's needs. Therefore, we need a methodology for constructing a pseudo-closure operator that matches an expected behavior, given a collection of neighborhoods. Since, in most cases, it is humanly impossible to define such a complex pretopological space, the methodology must be automatic and our proposal is to learn the pseudo-closure operator.

In this paper we present a method for automatic learning of a V-type pretopological space. Our method learns the underlying pseudo-closure operator, which defines the pretopological space. Cleuziou and Dias [28] were the first to tackle the problem of learning a pseudo-closure operator from observations. For AI purposes, they designed a semi-supervised method for learning a pretopological space from which a lexical taxonomy is derived. To this end, the authors first model the pseudo-closure operator as a linear combination of neighborhoods and then use a genetic algorithm that produces a solution that best matches a given (partial) structure.

This paper aims at both generalizing and improving the original approach of Cleuziou and Dias [28]. To this end, four main contributions are presented:

1. First, we generalize the problem to any application case. The proposed generic method learns a propagation concept instead of focusing on the construction of a structure (like a lexical taxonomy for instance).

2. Second, the pseudo-closure operator is modeled in a logical (rather than numerical) formalism thus providing a larger solution space and, more importantly, facilitating the subsequent learning process.

3. Third, the original stochastic learning strategy is abandoned in favor of an efficient learning approach, resulting in a multiple instance algorithm based on a greedy learning strategy.

4. Finally, the proposed new methodology is capitalized on the AI task of Lexical Taxonomy reconstruction, showing the effectiveness of both the pretopological theory and the proposed learning strategy in such a context.

The article is organized as follows: the basics of pretopology, useful for the understanding of the studied problem, are given in the following section (Section 2). The new modeling of the pseudo-closure operator as a logical combination (in *positive*[1] disjunctive normal form (DNF)) of neighborhoods is then detailed in Section 2.3. The pseudo-closure operator is acquired by a supervised multiple instance learning process [30, 76]. Multiple instance learning is particularly suitable when an observation can be described by multiple instances.

[1]By *positive*, we mean that the formula contains no negated literals.

This methodology (presented in Section 3) is necessary because we learn a pretopological space from its known *elementary closed sets*. In pretopology theory, a single closed set (considered as an observation in the learning problem) can be obtained by several propagation models (multiple instances). In fact, the number of valid propagation models, or pseudo-closure operators, is exponential in the size of the elementary closed sets considered. This leads us to a learning problem with a set of input examples whose size is exponential. We propose in Section 4 a method to provide a (low) estimate of the number of positive bags covered (true positives) and the exact number of negative bags covered (false positives) without the need to explicitly generate the dataset.

Section 5 is devoted to quantitative and qualitative comparisons of the newly proposed Learn Pretopological Space Multiple Instance (LPSMI) approach with that presented in [28] as well as a simpler greedy variation. We simulate a forest fire in a rectangular grid and then compare the performance of each algorithm by evaluating their ability to retrieve the underlying propagation model. The results confirm that the new approach performs best due to its more refined underlying quality measure, while maintaining good performance in terms of execution time.

Finally, the LPSMI methodology provides an elegant framework to efficiently contribute to the task of learning lexical taxonomies. Given a target lexical taxonomy structuring a set E of terms and a collection of *semantic* neighborhoods, a model capturing the hypernymy semantic relation is learned and then reused to build larger lexical taxonomies. This section (Section 6) presents both the framework for combining any state-of-the-art LT acquisition methods and the very promising results obtained experimentally on three real datasets.

2. Basics of Pretopology

The pretopology theory was born in the early 70's with the aim of relaxing topological axiomatics [12, 19]. Pretopology allows to study the relations between the elements of the power set $\mathcal{P}(E)$. A wide variety of works have shown the usefulness of this theory in many research areas: image analysis [54, 14], classification [36, 1], similarity or proximity detections tasks [71], and recognition of complex propagation (or expansion) phenomena [7, 2]. We place our work in the latter area.

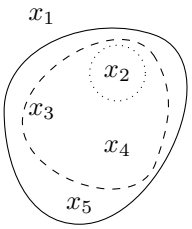
2.1. Pretopological spaces

A pretopological space is defined by a pair (E, a) where E is a non empty finite set of elements and $a(.)$ a pseudo-closure operator. The pseudo-closure operator $a(.)$ is a mapping from any set in $\mathcal{P}(E)$ to a larger set in $\mathcal{P}(E)$.

Definition 1 (Pseudo-closure operator)**.** Let E be a set and $a(.)$ an application of $\mathcal{P}(E)$ to $\mathcal{P}(E)$. Then $a(.)$ is a pseudo-closure operator if:

i $a(\emptyset) = \emptyset$

ii $\forall A \in \mathcal{P}(E)$, $A \subseteq a(A)$



$$\begin{aligned} A &= \{x_2\} \\ a(A) &= \{x_2, x_3, x_4\} \\ a^2(A) &= \{x_2, x_3, x_4, x_5\} \\ &= F(A) \end{aligned}$$

Figure 1: The pseudo-closure expansion process starting from the set $A = \{x_2\}$ and extending to the closed set $F(A) = \{x_2, x_3, x_4, x_5\}$.

The couple (E, a) is called a pretopological space.

The pseudo-closure operator is not idempotent, which means that, contrary to the closure operator of the topological theory, $a(A)$ is not necessarily equal to $a(a(A))$, where $A \in \mathcal{P}(E)$. The pretopological pseudo-closure operator is an iterated function such that $\forall A \in \mathcal{P}(E)$, $a(A) \subseteq a(a(A))$ (*cf.* Item ii). Let k be a positive integer and $A \in \mathcal{P}(E)$, we note $a^k(A)$ the k successive applications of $a(.)$ on A. It is formally defined as follows

$$\forall A \in \mathcal{P}(E),\ \forall k \in \mathbb{N},\ a^0(A) = A \text{ and } a^k(A) = (a \circ a^{k-1})(A)$$

Let k be a positive integer such that $a^k(A) = a^{k+1}(A)$, then $a^k(A)$ is called the closure of A, denoted $F(A)$. Figure 1 shows the multiple steps necessary to compute a closed set.

Definition 2 (Closed set)**.** Let (E, a) be a pretopological space. Then $K \in \mathcal{P}(E)$ is a closed set if and only if $a(K) = K$. The closure of $A \in \mathcal{P}(E)$ is a closed set noted $F(A)$. $F(A)$ is obtained by applying the pseudo-closure operator until it converges to a fixed point.

$$\forall A \in \mathcal{P}(E),\ \exists k \in \mathbb{N},\ F(A) = a^k(A) \text{ where } a^k(A) = a^{k+1}(A)$$

If A is a singleton, then $F(A)$ is an *elementary closed set.*

2.2. Pretopological spaces of type V

The pretopological spaces are classified according to the behavior of their pseudo-closure operator. *Pretopological spaces of type V* are probably the most studied.

Definition 3 (Pretopological space of type V)**.** A pretopological space (E, a) is of type V if and only if its pseudo-closure operator $a(.)$ respects the isotonic property defined as follows:

iii $\forall A, B \in \mathcal{P}(E),\ A \subseteq B \Rightarrow a(A) \subseteq a(B)$

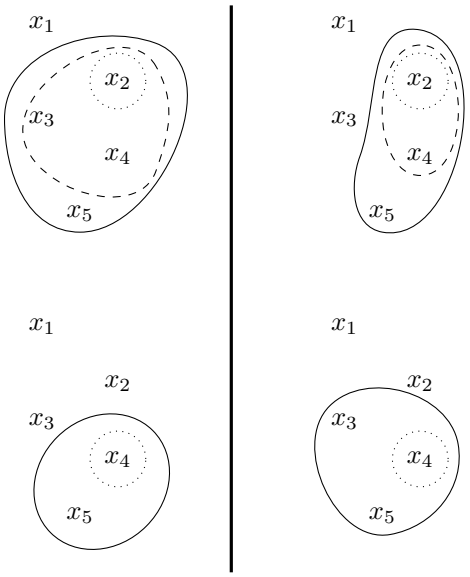


Figure 2: Computation of the elementary closed sets of two pretopological spaces. A dotted line shows the singleton from which the closure is computed, a dashed line shows an intermediate level of propagation and a solid line shows an elementary closed set. The proposed propagation steps on the left satisfy the requirements of a V-type pretopological space, while those on the right do not: $x_3 \notin a(\{x_2, x_4\})$ while $x_3 \in a(\{x_4\})$.

Figure 2 shows two pretopological spaces: the one on the left is a V-type pretopological space while the one on the right is not. V-type pretopological spaces have nice structuring properties, for example, the intersection of closed sets is a closed set. Largeron and Bonnevay [48] rely on these properties to define an algorithm that structures the elements of a V-type pretopological space into a directed acyclic graph (DAG). This algorithm is the cornerstone of the work of Cleuziou and Dias [28] in considering a lexical taxonomy as a DAG structuring the terms of natural language. Consider the set of elementary closed sets in Table 1, Figure 3 shows the idea behind this structuring process: first, all elementary closed sets are computed and then a DAG is deduced based on the inclusion scheme of the elementary closed sets.

This algorithm is equivalent to computing the transitive reduction of the graph defined by the adjacency matrix describing the elementary closed sets of the pretopological space. Aho et al. [4] show that the transitive reduction of any DAG can be computed from its transitive closure. Let M_{ecs} be an adjacency matrix encoding a set of elementary closed sets, its transitive reduction M_{red}, can be calculated as follows:

$$M_{red} = M_{ecs} \wedge \neg(M_{ecs} \cdot M_{ecs})$$

where $\wedge$ and $\neg$ are pairwise logical AND and logical NOT and $\cdot$ is the Boolean matrix product (i.e. addition is replaced by a logical OR and multiplication

$x \in E$	$F(\{x\})$
x_1	$\{x_1, x_2, x_3, x_4, x_5\}$
x_2	$\{x_2, x_3, x_4, x_5\}$
x_3	$\{x_3, x_5\}$
x_4	$\{x_4, x_5\}$
x_5	$\{x_5\}$

Table 1: A set of elementary closed sets of a pretopological space (E, a) of type V.

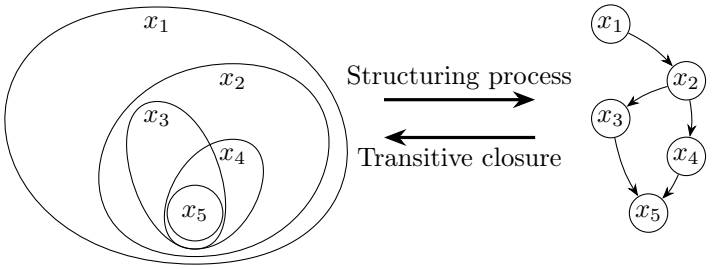


Figure 3: The elementary closed sets of a pretopological space of type V can be organized according to their inclusion scheme. A directed acyclic graph is then deductible from this organization and *vice versa*.

by a logical AND). Although this does not necessarily improve the theoretical complexity of the algorithm proposed by Largeron and Bonnevay [48], it could easily take advantage of modern hardware instructions to speed up, in practice, the computation of the DAG.

Such a DAG is a perfect encoding of the set of elementary closed sets of a pretopological space of type V, since they can be retrieved by means of the transitive closure. However, it is absolutely not sufficient to retrieve the pseudo-closure operator (and thus the pretopological space) since the non-elementary closed sets are unknown. Moreover, only the final state of the expansion of a singleton is known but the intermediate steps are necessary to deduce a pseudo-closure operator.

The isotonic property will also be of crucial importance for the pseudo-closure learning framework we propose in the remainder. This property allows for efficient learning of a pretopological space: as we will see in Section 4, it allows for an accurate estimate of the number of positive and negative bags covered by a solution under construction. Consequently, only V-type pretopological spaces will be considered in the rest of this paper.

2.3. Pseudo-closure operator and type V

The objective of this subsection is to describe several ways to define a pseudo-closure operator suitable for the construction of a V-type pretopological space. The advantages and (especially) disadvantages of each method will be discussed, leading to a new proposal.

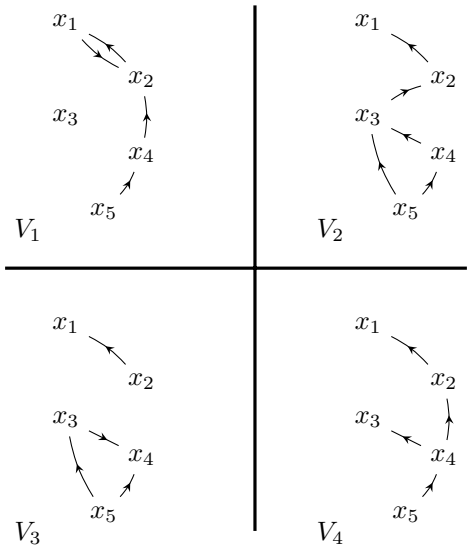


Figure 4: Four neighborhood relations (V_1, V_2, V_3, V_4) on the set $E = \{x_1, x_2, x_3, x_4, x_5\}$. An arrow going from one element $x \in E$ to another $y \in E$ indicates that $y \in V(x)$. Reflexive relations are ignored for simplicity.

A pseudo-closure operator is naturally defined by a neighborhood function on a given set E. A neighborhood function is an application of E to its power set $\mathcal{P}(E)$ such that any element $x \in E$ belongs to its neighborhood $V(x)$. It is formally defined as a function $V : E \to \mathcal{P}(E)$ such that $\forall x \in E$, $x \in V(x)$. A neighborhood function is equivalent to a binary relation R where an element $x \in E$ is in relation with another element $y \in E$ (denoted xRy) if and only if $y \in V(x)$. A neighborhood function can then be visualized as a directed graph where the vertices are elements of the set E and an edge from $x \in E$ to $y \in E$ indicates xRy and thus $y \in V(x)$. This representation is used in Figure 4 to illustrate four neighborhood functions.

Given such a neighborhood function, we can derive a pseudo-closure operator defined for any set $A \in \mathcal{P}(E)$ by $a(A) = \{x \in E \mid V(x) \cap A \neq \emptyset\}$. This operator extends a set $A \in \mathcal{P}(E)$ to any element $x \in E$ whose neighborhood intersects A. The resulting pretopological space (E, a) is known to be of type V [12, 19]. As an example, consider $a_{V_2}(.)$ as the pseudo-closure operator defined by the neighborhood relation V_2 described in Figure 4. The computation of the pseudo-closure of a set can be done visually by looking back along the edges: x_2 is in the pseudo-closure of $\{x_1\}$ since there is an edge from x_2 to x_1. The closure of the singleton $\{x_1\}$ is then computed as follows:

$$\{x_1\} \xrightarrow{a_{V_2}} \{x_1, x_2\} \xrightarrow{a_{V_2}} \{x_1, x_2, x_3\} \xrightarrow{a_{V_2}} \{x_1, x_2, x_3, x_4, x_5\}$$

Although this is a fairly common way of defining a pseudo-closure operator, it is not the most interesting way of taking advantage of the theory of pretopol-

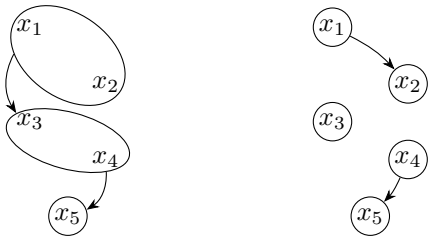


Figure 5: Two directed acyclic graphs describing the elementary closed sets of weak and strong pretopologies.

ogy, since the closure operator is basically equivalent to the transitive closure operator on the graph defined by the neighborhoods function V. The pretopology theory indeed reveals its power in the multiple-criteria context, i.e. when the pseudo-closure operator is defined by a combination of multiple neighborhoods.

Let us consider a collection of k neighborhood functions $\mathcal{V} = \{V_1, \ldots, V_k\}$ on E. Bui et al. [21] define the two concepts of *strong* and *weak* pseudo-closure operators. These two pseudo-closure operators, respectively denoted $a_{strong}(.)$ and $a_{weak}(.)$, give rise to two pretopological spaces (E, a_{strong}) and (E, a_{weak}) of type V.

$$\forall A \in \mathcal{P}(E),\ a_{strong}(A) = \{x \in E \mid \forall V \in \mathcal{V},\ V(x) \cap A \neq \emptyset\}$$
$$\forall A \in \mathcal{P}(E),\ a_{weak}(A) = \{x \in E \mid \exists V \in \mathcal{V},\ V(x) \cap A \neq \emptyset\}$$

The strong pseudo-closure operator $a_{strong}(.)$ extends a set $A \in \mathcal{P}(E)$ to any element $x \in E$ such that all the neighborhoods $V_i(x)$ intersect A; $a_{weak}(.)$ extends the set A to any element $x \in E$ such that at least one of the k neighborhoods $V_i(x)$ intersects A. Figure 5 shows the structures arising from the weak and the strong pretopological spaces defined by the four neighborhoods in Figure 4.

A strong assumption is made in both cases: the strong pretopology assumes that every neighborhood is equally *necessary* to trigger an expansion, and weak pretopology assumes that every neighborhood is *sufficient* to trigger an expansion. These assumptions seem to be wrong most of the time. Thus, research has been conducted on a methodology to construct a multiple-neighborhood based pseudo-closure operator designed such that useful neighborhoods are selected and combined while erroneous ones are discarded. Cleuziou and Dias [28] were the first to tackle this problem by proposing a pseudo-closure operator defined by a set of weighted neighborhoods.

2.3.1. Weighted modeling of pseudo-closure

This subsection first recalls the definition of (weighted) pseudo-closure proposed by Cleuziou and Dias [28] and then highlights the limitations of such a numerical and linear modeling.

Cleuziou and Dias [28] introduce the notion of *parametrized pseudo-closure*. A parametrized pseudo-closure operator is defined by a set of weighted neighborhood functions plus a bias weight. Let E be a set, $\mathcal{V} = \{V_1, \ldots, V_k\}$ a set of k neighborhoods on E, $\boldsymbol{w} = \{w_1, \ldots, w_k\}$ a set of k weights, and w_0 a bias weight. This operator is designed in such a way that a propagation is triggered if and only if a sufficient part of the neighborhoods agree on this propagation. We say that a neighborhood $V \in \mathcal{V}$ is *agree* on a propagation from a set $A \in \mathcal{P}(E)$ to an element $x \in E$ when $V(x) \cap A \neq \emptyset$. If the sum of the weights associated to these neighborhoods is greater than or equal to the bias weight w_0, then the set A propagates to the element x.

This agreement process is defined by a linear threshold function $f_{\boldsymbol{w}}()$ given by:

$$\forall A \in \mathcal{P}(E),\, \forall x \in E,\; f(A,x) = \begin{cases} 1 & \text{if } \left(\sum_{\{0<i\leq k | V_i(x)\cap A\neq\emptyset\}} w_i\right) \geq w_0 \\ 0 & \text{otherwise} \end{cases}$$

The parameterized pseudo-closure operator $a_{\boldsymbol{w}}(.)$ is then expressed by $\forall A \in \mathcal{P}(E)$, $a_{\boldsymbol{w}}(A) = \{x \in E \mid f(A,x) = 1\}$ and the pretopological space $(E, a_{\boldsymbol{w}})$ is of type V [28]. This definition of the pseudo-closure operator generalizes existing combination-based operators. For example, there exists weight assignments retrieving either *weak* or *strong* pseudo-closure operators from Bui et al. [21]. The weak pseudo-closure operator is equivalent to any parameterized pseudo-closure operator such that each weight w_i is greater than or equal to w_0; the strong pseudo-closure operator is equivalent to the parameterized pseudo-closure operator where each weight w_i is equal to $\frac{w_0}{k}$.

Although this parameterized-pseudo closure operator allows to define much more flexible pretopological spaces than previous definitions of the pseudo-closure operator, two major limitations remain. First, the **expressiveness** of the linear modeling is still quite limited, as not all propagation phenomena can be expressed in the form of a linear model. For example, a simple rule such as "if V_1 and V_4 agree, or if V_2 and V_3 agree, then a propagation must be triggered" cannot be expressed by this approach. Second, given a set E and a collection of k neighborhoods on E, the number of weight vectors is unbounded while the number of pretopological spaces is bounded. This implies a large number of **redundant** weight vectors and makes the exploration of the weight vector spaces tedious, especially for the purpose of learning the pseudo-closure operator as we will see in the next section.

As an example, consider the collection $\mathcal{V} = \{V_1, V_2, V_3, V_4\}$ of four neighborhoods illustrated in Figure 4 and the two sets of elementary closed sets conforming to type V S_1^* and S_2^* illustrated in Figure 6 and Table 2. The problem is to find two sets of weights $\boldsymbol{w_1}$ and $\boldsymbol{w_2}$ such that the pseudo-closure operators $a_{\boldsymbol{w_1}}(.)$ and $a_{\boldsymbol{w_2}}(.)$ retrieve, respectively, the sets of elementary closed sets S_1^* and S_2^*.

In order to illustrate the two limitations stated above (expressiveness and redundancy), we show that S_1^* cannot be retrieved by the linear weighting approach (lack of expressiveness) while it exposes an infinite number of equivalent

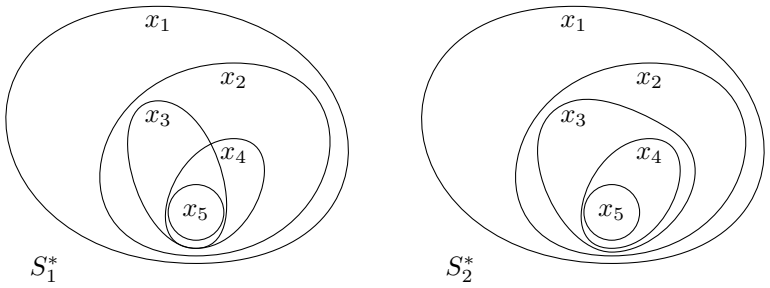


Figure 6: Two sets of elementary closed sets.

$x \in E$	$F_1^*(x)$	$F_2^*(x)$
x_1	$\{x_1, x_2, x_3, x_4, x_5\}$	$\{x_1, x_2, x_3, x_4, x_5\}$
x_2	$\{x_2, x_3, x_4, x_5\}$	$\{x_2, x_3, x_4, x_5\}$
x_3	$\{x_3, x_5\}$	$\{x_3, x_4, x_5\}$
x_4	$\{x_4, x_5\}$	$\{x_4, x_5\}$
x_5	$\{x_5\}$	$\{x_5\}$

Table 2: Two closure operators $F_1^*(.)$ and $F_2^*(.)$ leading, respectively, to elementary closed sets in S_1^* and S_2^*

solutions to retrieve S_2^* (redundancy).

Expressiveness. The target set S_1^* of elementary closed sets is retrievable by any weight assignment such that a propagation is triggered if and only if either V_1 and V_4 match, or V_2 and V_3 match. Such an assignment can be found by solving the system with the following constraints:

$$\begin{aligned} w_1 + w_4 &\geq w_0 & w_0 &> 0 \\ w_2 + w_3 &\geq w_0 & \max\{w_1, w_4\} + \max\{w_2, w_3\} &< w_0 \end{aligned}$$

It can be shown that this system is unsolvable even if the solution we are looking for can be formulated as simply as $(V_1 \wedge V_4) \vee (V_2 \wedge V_3)$. It is well known that such a Boolean function is not a threshold function and therefore cannot be expressed with a weighting model.

Redundancy. A solution allowing to perfectly retrieve the target set S_2^* of elementary closed sets is the vector $\boldsymbol{w} = (0.0, 0.5, 0.5, 1.0)$ with the bias weight $w_0 = 1.0$. It is easier (and equivalent) to represent $\boldsymbol{w}$ by the logical formula $(V_2 \wedge V_3) \vee V_4$ meaning that $x \in a_{\boldsymbol{w}}(A)$ if and only if at least one of the following two conditions is satisfied:

1. both V_2 and V_3 allow it (i.e. both $V_2(x)$ and $V_3(x)$ intersect A, not necessarily on the same element(s)),
2. V_4 allows it (i.e. $V_4(x)$ intersects A).

The vector $\boldsymbol{w}$ is then used to define the pseudo-closure operator $a_{\boldsymbol{w}}(.)$, which behaves as follows:

$$\begin{aligned}
&\{x_1\} \xrightarrow[\text{and } V_4]{V_2 \wedge V_3} \{x_1, x_2\} \xrightarrow{V_4} \{x_1, x_2, x_4\} \xrightarrow[\text{and } V_4]{V_2 \wedge V_3} \{x_1, x_2, x_3, x_4, x_5\} && = F_{\boldsymbol{w}}(\{x_1\}) \\
&\{x_2\} \xrightarrow{V_4} \{x_2, x_4\} \xrightarrow[\text{and } V_4]{V_2 \wedge V_3} \{x_2, x_3, x_4, x_5\} && = F_{\boldsymbol{w}}(\{x_2\}) \\
&\{x_3\} \xrightarrow[\text{and } V_4]{V_2 \wedge V_3} \{x_3, x_4, x_5\} && = F_{\boldsymbol{w}}(\{x_3\}) \\
&\{x_4\} \xrightarrow[\text{and } V_4]{V_2 \wedge V_3} \{x_4, x_5\} && = F_{\boldsymbol{w}}(\{x_4\}) \\
&\{x_5\} \xrightarrow{\emptyset} \{x_5\} && = F_{\boldsymbol{w}}(\{x_5\})
\end{aligned}$$

Each arrow describes a single step of propagation from an element $x \in E$ to its elementary closure $F_{\boldsymbol{w}}(\{x\})$, and the clauses above and below each arrow are the ones participating in the propagation. Let us observe that it is easy to find another vector $\boldsymbol{w'} \neq \boldsymbol{w}$ defining an operator $a_{\boldsymbol{w'}}$ similar to $a_{\boldsymbol{w}}$ (e.g. $\boldsymbol{w'} = (0.0, 0.6, 0.4, 1.0)$ and $w_0 = 1.0$). In fact, the operator $a_{\boldsymbol{w}}(.)$ is equivalent to any other operator $a_{\boldsymbol{w'}}(.)$ where $\boldsymbol{w'} = (w_1, w_2, w_3, w_4)$ and the bias w_0 satisfy the system of constraints below, which admits an infinite number of solutions.

$$\begin{aligned}
&w_0 > 0 && \\
&w_1 < w_0 && w_2 + w_3 \geq w_0 \\
&w_2 < w_0 && w_4 \geq w_0 \\
&w_3 < w_0 && w_1 < w_0 - \max\{w_2, w_3\}
\end{aligned}$$

It seems worth mentioning that while both $V_2 \wedge V_3$ and V_4 are involved in the expansion from $\{x_2, x_4\}$ to $\{x_2, x_3, x_4, x_5\}$, only $V_2 \wedge V_3$ is responsible for the inclusion of x_3 in $a_{\boldsymbol{w}}(\{x_2, x_4\})$. Moreover, the reasons of this propagation are not trivial; indeed, V_2 is at the origin of the propagation of $\{x_2, x_4\}$ to x_3 because x_2 is in the neighborhood of x_3; whereas V_3 is at the origin because x_4 is in the neighborhood of x_3.

2.3.2. A new logical approach

To overcome the two main limitations of weighted modeling discussed above (lack of expressiveness and redundancy), we propose to define a combination of neighborhoods as a disjunctive normal form (DNF) logical formula. We restrict our work to positive DNFs, i.e. DNFs without negative literals.

Such modeling requires translating the information provided by a neighborhood into a logical concept. This can be done by transforming any neighborhood function V_i in the collection of neighborhoods $\mathcal{V}$ into a logical predicate $q_i : \mathcal{P}(E) \times E \rightarrow \{0, 1\}$ determining whether the neighborhood $V(x)$ of an element $x \in E$ intersects a set $A \in \mathcal{P}(E)$.

$$\forall V_i \in \mathcal{V},\ A \in \mathcal{P}(E),\ x \in E,\ q_i(A, x) = \begin{cases} 1 & \text{if } V_i(x) \cap A \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Property 1. *Any predicate $q : \mathcal{P}(E) \times E \to \{0, 1\}$ derived from a neighborhood function $V : E \to \mathcal{P}(E)$ as defined in Equation* (1) *respects the isotonic property:*

$$\forall A, B \in \mathcal{P}(E),\ \forall x \in E,\ A \subseteq B \Rightarrow q(A, x) \leq q(B, x)$$

Proof. Let V be a neighborhood function defined on a set E and q the predicate derived from V.

$$\begin{aligned}
&\forall X \in \mathcal{P}(E),\ \forall x \in E, \qquad q(X, x) = 1 \Leftrightarrow V(x) \cap X \neq \emptyset \\
&\forall A, B \in \mathcal{P}(E),\ \forall x \in E, \qquad A \subseteq B \Rightarrow [(V(x) \cap A \neq \emptyset) \Rightarrow (V(x) \cap B \neq \emptyset)] \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \Rightarrow q(A, x) \leq q(B, x)
\end{aligned}$$

Thus, any predicate derived from a neighborhood function as defined in Equation (1) respects the isotonic property. □

Let $\mathcal{Q} = \{q_i\}_{\forall V_i \in \mathcal{V}}$ be a set of predicates. Then, a *logical pseudo-closure operator* $a_Q(.)$ can be derived from any positive DNF Q defined on the language $(\mathcal{Q}^*)^*$, namely, the infinite Cartesian power of the infinite Cartesian power of $\mathcal{Q}$: $\mathcal{Q}^*$ is the set of all conjunctive clauses, and $(\mathcal{Q}^*)^*$ is the set of all the positive DNFs.

$$\forall A \in \mathcal{P}(E)\,, a_Q(A) = \{x \in E \mid Q(A, x)\}$$

Theorem 1. *Let Q be a positive DNF composed of predicates respecting the isotonic property, then the pretopological space (E, a_Q) derived from Q is of type V.*

Proof. The pretopological space (E, a_Q) is of type V if and only if $\forall A, B \in \mathcal{P}(E)$, $A \subseteq B \Rightarrow a_Q(A) \subseteq a_Q(B)$ (isotonic property). Let c_i be the i-th conjunctive clause of Q and q_{ij} the j-th literal in c_i.

$$\begin{aligned}
\forall x \in E,\ x \in a_Q(A) &\Leftrightarrow Q(A, x) = 1 \\
&\Leftrightarrow \exists i, c_i(A, x) = 1 \\
&\Leftrightarrow \exists i \forall j, q_{ij}(A, x) = 1
\end{aligned}$$

Since all predicates $q_{ij}(A, x)$ are of type V, the conjunctive clause $c_i(A, x)$ is obviously of type V as well, and so is the DNF Q. So if Q is a positive DNF composed of predicates of type V, then (E, a_Q) is a pretopological space of type V. □

In what follows, we consider only well-formed (or simplified) DNFs. That is, DNFs that satisfy the following properties:

- A clause is present only once
- A clause is not subsumed by another (more general)

For example, $Q_1 = q_1 \vee q_1$ and $Q_2 = (q_1 \wedge q_2) \vee q_1$ are both ill-formed DNFs. Q_1 is ill-formed because the clause q_1 appears twice and Q_2 is also ill-formed because $q_1 \wedge q_2 \Rightarrow q_1$, so $q_1 \wedge q_2$ is subsumed by q_1. $Q_3 = q_1$ is the only well-formed DNF equivalent to both Q_1 and Q_2. This simplification reduces the space of valid DNFs since it would, like the weight vector model, be infinite otherwise.

We have introduced a new logical model to represent a combination of neighborhoods on a given finite non-empty set E. This combination is a logical formula in positive disjunctive normal form used to define a decision function $Q : \mathcal{P}(E) \times E \rightarrow \{0, 1\}$. This decision function allows us to construct a pretopological space (E, a_Q) of type V by defining the result of the pseudo-closure operator applied on a subset $A \in \mathcal{P}(E)$ as the set of all elements $x \in E$ satisfying the decision function $Q(A, x)$. Let us now show that the new proposed model is more expressive than the previous weighted model.

Property 2. *Any weighted pseudo-closure operator $a_{\boldsymbol{w}}(.)$ can be reformulated as a logical pseudo-closure $a_Q(.)$.*

Proof. Consider a set E, a collection $\mathcal{V}$ of k neighborhoods, a weight vector $\boldsymbol{w} = (w_1, \ldots, w_k)$, a bias weight w_0 and a positive DNF Q constructed by making a conjunctive clause of each combination of weights whose sum is greater than or equal to w_0. Then the threshold function f is equivalent to Q.

$f(A, x) = 1$ means that there is a combination of neighborhoods $\mathcal{V}' = \{V \in \mathcal{V} \mid V(x) \cap A \neq \emptyset\}$ such that the sum of their associated weights is greater than w_0. By construction of Q, the logical conjunction $\bigwedge_{V_i \in \mathcal{V}'} q_i(A, x)$ belongs to Q and is true (by definition of $\mathcal{V}'$).

On the contrary, $f(A, x) = 0$ means that there is no combination of neighborhoods $\mathcal{V}' = \{V \in \mathcal{V} \mid V(x) \cap A \neq \emptyset\}$ such that the sum of their associated weights is greater than w_0. By construction of Q, there does not exist either a conjunction c such that $c(A, x) = 1$. □

As an example, let us consider the vector $\boldsymbol{w} = (0.0, 0.5, 0.5, 1.0)$ and the bias weight $w_0 = 1.0$ defined previously to retrieve the set S_2^* of elementary closed sets. If we omit w_1, which is zero, there are five distinct combinations of weights whose sum is greater than or equal to w_0: w_4, (w_2, w_3), (w_2, w_4), (w_3, w_4), and (w_2, w_3, w_4). An equivalent DNF is $Q = (q_4) \vee (q_2 \wedge q_3) \vee (q_2 \wedge q_4) \vee (q_3 \wedge q_4) \vee (q_2 \wedge q_3 \wedge q_4)$. This DNF is obviously ill-formed, but it expresses exactly the same intent as $\boldsymbol{w}$. This formula can be simplified into the well-formed (and more readable) DNF $Q' = (q_2 \wedge q_3) \vee q_4$.

Property 2 establishes that the logical pseudo-closure model is at least as expressive as the weighted model, but our new formalization was motivated by improving expressiveness over the previous model. Indeed, as mentioned before, some DNF cannot be expressed as a weight vector: we could not find a linear model able to retrieve the set S_1^* of elementary closed sets. We have already proposed a DNF that meets our needs: $Q_2 = (q_1 \wedge q_2) \vee (q_3 \wedge q_4)$, which cannot be modeled by a weight set.

To conclude on this section, we have introduced a simple and versatile model for the pseudo-closure operator which is based on a logical formalism. Our model

is versatile because it is able to express a wider variety of V-type pretopological spaces than existing models. It is simple because the space of all DNFs is finite and easier to explore unlike the (infinite) space of weighted vectors. It is also simple because a DNF is much more human readable than a weight vector, and thus easier to interpret.

3. Learning a pretopological space of type V

In this section, we address the problem of learning a V-type pretopological space. Cleuziou and Dias [28] introduced the framework *Learn Pretopological Space* (LPS). They propose to learn a pretopological space in a semi-supervised or, as they claim, a *self-supervised* manner. Their work is presented as an approach to automatically extract a lexical taxonomy. This is in fact a reductive view of the outcomes of learning pretopological (V-type) spaces. As we have shown in the previous section, a hierarchical structure is hidden behind any V-type pretopological space. The goal of the LPS framework is to learn a *structuring model* and it can be used to find a solution to any task whose output is a directed acyclic graph (DAG), lexical taxonomy extraction is an example of such tasks.

The original LPS algorithm uses a genetic algorithm to learn a structuring model. The learning algorithm is guided by an optimization function that measures the correspondence between the learned structure and a target structure; we call this function the *extrinsic quality measure*. In this section we outline our approach to learning a positive DNF using the same genetic algorithm, but in a supervised setting, instead of a semi-supervised setting as in the original work. Since this type of algorithm suffers from high complexity, we then introduce a greedy learning approach that reduces this complexity issue.

3.1. Extrinsic quality measure

Given a finite set E of elements, a set $S^* = \{F^*(\{x\})\}_{x\in E}$ of target elementary closed sets and a collection $\mathcal{V}$ of neighborhoods, the LPS task is to learn a pretopological space (E, a) whose underlying structuring $S = \{F(\{x\})\}_{x\in E}$ (i.e. its set of elementary closed sets) best matches S^*. A quality measure, quantifying the degree of correspondence, is thus necessary to guide the learning process.

Cleuziou and Dias [28] used the F-measure which computes a trade-off between accuracy and completeness when comparing two sets of closed sets. Accuracy is defined as the precision of the solution, and completeness by the recall of the solution:

$$\text{Precision}(S, S^*) = \frac{\sum_{x\in E} |F(\{x\}) \cap F^*(\{x\})|}{\sum_{x\in E} |F(\{x\})|}$$

$$\text{Recall}(S, S^*) = \frac{\sum_{x\in E} |F(\{x\}) \cap F^*(\{x\})|}{\sum_{x\in E} |F^*(\{x\})|}$$

$x \in E$	$F(\{x\})$	$F^*(\{x\})$	$\lvert F(\{x\}) \cap F^*(\{x\})\rvert$
x_1	$\{x_1, x_2, x_3, x_4, x_5\}$	$\{x_1, x_2, x_3, x_4, x_5\}$	5
x_2	$\{x_2, x_3, x_4, x_5\}$	$\{x_2, x_3, x_4, x_5\}$	4
x_3	$\{x_3, x_4, x_5\}$	$\{x_3, x_5\}$	2
x_4	$\{x_4, x_5\}$	$\{x_4, x_5\}$	2
x_5	$\{x_5\}$	$\{x_5\}$	1
		Precision:	$\frac{14}{15} \approx 0.93$
		Recall:	$\frac{14}{14} = 1$
		F-measure:	$2 \cdot \frac{0.93 \cdot 1}{0.93+1} \approx 0.96$

Table 3: Example of computing the F-measure between two sets of elementary closed sets.

Precision (resp. recall) is defined as the ratio of the number of correctly retrieved elements in the learned elementary closed sets (S) to the total number of elements in the learned elementary closed sets S (resp. in the target elementary closed sets S^*). The F-measure is then defined as the harmonic mean between precision and recall.

$$\text{F-measure}(S, S^*) = 2 \cdot \frac{\text{Precision}(S, S^*) \cdot \text{Recall}(S, S^*)}{\text{Precision}(S, S^*) + \text{Recall}(S, S^*)}$$

Table 3 illustrates the computation of the F-measure by comparing a *closure function*[2] $F(.)$ to a target structuring, $F^*(.)$, on a set E.

In the specific context of the LPS problem, we call such a quality measure an *extrinsic quality measure* since it evaluates only the final elementary closed sets revealed by the structuring process, rather than the (internal or *intrinsic*) propagation process itself.

3.2. Genetic LPS

For clarity, we present two variants of LPS: *Numerical Genetic LPS* [28] and *Logical Genetic LPS* whose outputs are pretopological spaces defined by a DNF. The term *Genetic LPS framework* refers to these two algorithms.

The Genetic LPS framework aims at learning a pretopological space by means of a genetic algorithm. Given a set S^* of target elementary closed sets and a collection $\mathcal{V}$ of neighborhoods, its output is a combination of neighborhoods (a weight vector or a DNF, depending on the algorithm) defining a pseudo-closure operator leading to the highest extrinsic measure on the resulting set S of elementary closed sets. Algorithm 1 provides a pseudo-code of the generic Genetic LPS framework.

The algorithm starts by generating a set of random candidate solutions (Line 7). A pretopological space is then constructed from each candidate and

[2]The term *closure function* refers to the function which associates to each element x its (elementary) closed set $F(x)$.

its set of elementary closed sets is computed and evaluated against the target S^* (Lines 12 and 13). Finally, a new set of candidates is computed by crossing and mutating the best candidates of the previous population (Line 28). The algorithm stops when the score of the best candidate remains stable for a given number of steps (Line 19).

In order to compute the extrinsic quality of a solution Q, the set S_Q of elementary closed sets of the pretopological space (E, a_Q) must be computed. This structuring step is the most expensive operation we rely on, so we choose to define the complexity of LPS algorithms in terms of the number of structuring steps required.

Genetic algorithms tend to need many iterations to converge to a solution, which makes their execution a bit slow. The quality of their results is often related to the size of their initial population, which is itself related to the time needed to converge to an acceptable solution. For example, Algorithm 1 has a complexity (in terms of number of structurations) of $O(initial\_pop \cdot max\_iter)$ with $initial\_pop$ generally high. Moreover, due to the stochastic nature of these algorithms, multiple runs may lead to different outputs. For these reasons, we introduce in the next subsection a less expensive (greedy) learning approach.

3.3. Greedy LPS

The Greedy LPS is a greedy variant of the Logical Genetic LPS: given a set S^* of target elementary closed sets and a collection $\mathcal{V}$ of neighborhoods, its output is a combination of the neighborhoods in $\mathcal{V}$ as a positive DNF Q. It uses a greedy heuristic designed to speed up the learning process.

The pseudo-code of the Greedy LPS is presented in Algorithm 2. The algorithm starts with an empty DNF $\tilde{Q}$ (Line 6); conjunctive clauses are then successively added to $\tilde{Q}$. A beam search is performed to find the clause c such that the set of elementary closed sets S_Q computed from the DNF $Q = \tilde{Q} \vee c$ maximizes the extrinsic measure defined above (Line 9). If the quality of Q is higher than the quality of the previous DNF $\tilde{Q}$, then c is added to $\tilde{Q}$ and the next iteration starts, otherwise the algorithm stops.

Algorithm 3 is the engine of the Greedy LPS. Given a DNF Q, its output is a conjunctive clause c maximizing the extrinsic quality measure of the DNF $Q \vee c$. The search of the clause c is done by specializing an empty clause until it can no longer be specialized (Line 2). It is called the first time in Line 9 of Algorithm 2. The function starts by evaluating all well-formed conjunctive clauses a little more specialized than the parameter $base\_clause$, that is to say $base\_clause$ plus a predicate in $Preds$ and not in $base\_clause$ (see the loop starting Line 9). Then, the set of evaluated conjunctive clauses is sorted and filtered to keep only the bests $beam$. The second loop of the function consists in calling itself recursively with the parameter $base\_clause$ fixed to one of the previously filtered clauses (Line 23), in order to evaluate larger clauses.

The Greedy LPS has a complexity of $O(max\_iter \cdot |\mathcal{V}| \cdot beam)$ that is in practice much lower than that exhibited by the Genetic LPS since $|\mathcal{V}| \cdot beam$ is typically much smaller than the initial population size of the Genetic LPS.

Algorithm 1: Genetic LPS

```
1  Function GeneticLPS (E, S*, 𝒱, max_iter, initial_pop,
     required_iter_convergence)
2      iter ← 0
3      iter_conv ← 0
4      score ← 0
5      stop ← false
6      Q̃ ← ∅
7      population ← initial_pop random solutions
8      while iter < max_iter and ¬stop do
9          iter ← iter + 1
10         scores ← [ ]                          /* an empty array */
11         foreach Q ∈ population do
12             S_Q ← structuring(E, 𝒱, Q)
13             Append extrinsic_measure(S_Q, S*) to scores
14         end
15         best_score ← greatest value in scores
16         best_individual ← the individual exposing the highest score
17         if best_score = score then
18             iter_conv ← iter_conv + 1
19             if iter_conv = required_iter_convergence then
                   /* If best_score remains the same for
                      required_iter_convergence steps, then stop the
                      learning                                      */
20                 stop ← true
21             end
22         end
23         else if best_score > score then
24             iter_conv ← 0   /* best_score is modified, so reset to
                   0 */
25             score ← best_score
26             Q̃ ← best_individual
27         end
28         population ← cross and mutate best individuals
29     end
30     return Q̃
31 end
```

Algorithm 2: Greedy LPS

Function *GreedyLPS (E, S^*, $\mathcal{V}$, max_iter, beam)*
  $iter \leftarrow 0$
  $score \leftarrow 0$
  $stop \leftarrow false$
  $Preds \leftarrow$ predicates derived from $\mathcal{V}$
  $\tilde{Q} \leftarrow \emptyset$
  while $iter < max\_iter$ *and* $\neg stop$ **do**
    $iter \leftarrow iter + 1$
    $c \leftarrow FindBestClause(E, S^*, Preds, \tilde{Q}, \emptyset, beam)$
    $Q \leftarrow \tilde{Q} \vee c$
    $S_Q \leftarrow structuring(E, Q)$
    if $clause = \emptyset$ *or* $extrinsic\_measure(S_Q, S^*) \leq score$ **then**
      $stop \leftarrow true$
    end
    else
      $score \leftarrow extrinsic\_measure(S_Q, S^*)$
      $\tilde{Q} \leftarrow Q$
    end
  end
  return $\tilde{Q}$
end

Algorithm 3: Building the *best* clause

```
1  Function FindBestClause (E, S*, Preds, Q, base_clause, beam)
2      if |base_clause| ≥ |Preds| then
           /* base_clause contains all the predicates in Preds  */
3          return base_clause
4      end
5      best_clause ← ∅
6      best_score ← 0
7      clauses ← [ ]
8      scores ← [ ]
9      foreach q ∈ Preds \ base_clause do
10         c ← base_clause ∧ q
11         Q' ← Q ∨ c
12         S_Q' ← structuring(E, Q')
13         score ← extrinsic_measure(S_Q', S*)
14         clauses ← append c to clauses
15         scores ← append score to scores
16         if score > best_score then
17             best_score ← score
18             best_clause ← c
19         end
20     end
21     clauses ← sort clauses according to scores in decreasing order
        /* loop over the beam best clauses                    */
22     foreach c ∈ clauses[1 ... beam] do
23         beam_c ← FindBestClause(E, S*, Preds, Q, c, beam)
24         Q' ← Q ∨ beam_c
25         S_Q' ← structuring(E, Q')
26         score ← extrinsic_measure(S_Q', S*)
27         if score > best_score then
28             best_score ← score
29             best_clause ← c
30         end
31     end
32     return best_clause
33 end
```

The Greedy LPS is much simpler than the Genetic LPS framework and results in much shorter run times. In addition, further experimental comparisons (*cf.* Section 5) reveal that, despite the lack of completeness in exploring the solution space and due to the conciseness of the pseudo-closures produced, the greedy learning strategy significantly outperforms the stochastic approaches in the LPS task.

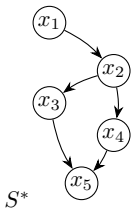
The premise of the following contribution is that the extrinsic measure used in the previous learning methodologies is not suitable for incremental learning strategies. Greedy LPS is expected to retrieve even better models by using an objective criterion that considers not only the quality of structuring (elementary closed sets) but also the potential of pseudo-closure (elementary and non-elementary closed sets).

For example, suppose one is learning the pretopological space presented in Figure 7. Given two candidate DNFs Q and Q', one needs to know which of the two closure operators F_Q and $F_{Q'}$ best retrieves the set S^* of target elementary closed sets. Both Q and Q' share the same (extrinsic) F-measure score, therefore the Greedy LPS is unable to determine which DNF is best. Choosing the wrong clause in the iteration i could have a huge impact on the clause chosen in the iteration $i+1$ (and further). Therefore, it is fundamental to be able to determine which of Q or Q' is more useful.

Recall that only pretopological spaces of type V are considered, so $\forall A, B \in \mathcal{P}(E)$, $A \subseteq B \Rightarrow a(A) \subseteq a(B)$. Thus, the fact that $F_Q(\{x_2\}) = \{x_2, x_3\}$ can be used to deduce information about the closures of the super-sets of $\{x_2\}$: any set $A \in \mathcal{P}(E)$ such that $x_2 \in A$ will propagate to at least x_3. Of course, the same applies to Q': $\{x_4\}$ and its super-sets will propagate to x_5. Although the information provided by Q and Q' are quantitatively equivalent, Q gives more useful information. Indeed, Q' tells us that, because $x_5 \in F_{Q'}(\{x_4\})$, then $x_5 \in F_{Q'}(\{x_3, x_4\})$. But the propagation of the set $\{x_3, x_4\}$ does not provide any useful information since it is not reachable from any singleton[3] (according to S^*). On the contrary, Q tells us that all the super-sets of $\{x_1\}$ propagate to x_3. Since $F^*(\{x_1\}) = E$, all super-sets of x_1 are reachable (at least from $\{x_1\}$), so this information is valuable for learning the target pretopological space. Thus, Q gives more qualitative information than Q'.

A different quality measure that takes this *intrinsic information* into account is needed because such a method would be able to determine that, although Q and Q' are equivalent in terms of F-measure (i.e., the extrinsic quality measure), Q has more *potential* with respect to subsequent iterations. The term *potential* is appropriate because, although Q informs us that $x_3 \in F_Q(\{x_1, x_2\})$, this information is not reflected by the resulting elementary closed sets. However, it can be, and will be, important when learning additional clauses since it will influence the algorithm to learn a clause completing the (elementary) closure of $\{x_1\}$. On the contrary, the extrinsic quality measure cannot influence the results

[3]Even if $\{x_3, x_4\} \subset F^*(\{x_1\})$, $x_1 \notin \{x_3, x_4\}$, it is thus not reachable from $\{x_1\}$



$x \in E$	$F^*(\{x\})$	$F_Q(\{x\})$	$F_{Q'}(\{x\})$
x_1	$\{x_1, x_2, x_3, x_4, x_5\}$	$\{x_1, x_3\}$	$\{x_1\}$
x_2	$\{x_2, x_3, x_4, x_5\}$	$\{x_2, x_3\}$	$\{x_2, x_3\}$
x_3	$\{x_3, x_5\}$	$\{x_3\}$	$\{x_3\}$
x_4	$\{x_4, x_5\}$	$\{x_4\}$	$\{x_4, x_5\}$
x_5	$\{x_5\}$	$\{x_5\}$	$\{x_5\}$

Figure 7: A target set of elementary closed sets S^* and two candidate closure operators.

of the following iterations in this way because it considers that any expansion is equally important, which is false as we have just shown.

Moreover, the pretopological closure operator is defined in such a way that a single collection of elementary closed sets can be obtained by many pseudo-closure operators. For example, there are three ways to obtain an elementary closure $F(\{x\}) = \{x, y, z\}$, as illustrated below. This means that if we try to learn a pretopological space of type V from its elementary closed sets, we have to consider all the pseudo-closure operators leading to the same elementary closed sets.

1. $a(\{x\}) = \{x, y, z\}$
2. $a(\{x\}) = \{x, y\}$ and $a(\{x, y\}) = \{x, y, z\}$
3. $a(\{x\}) = \{x, z\}$ and $a(\{x, z\}) = \{x, y, z\}$

Given a set S^* of elementary closed sets, we propose to learn a pseudo-closure operator able to retrieve S^*. Such an operator seems not to be unique. This means that the target function can take several valid *forms*, each one being described by a different set of features (i.e. different combinations of neighborhoods). This is the formulation of the multiple instance problem [30] on which the rest of the contribution is based.

4. A multiple instance approach

The first objective of this section concerns the formulation of the LPS framework in the Multiple Instance (MI) formalism, which offers the possibility of introducing a new (intrinsic) fitness measure to overcome the limitations of the existing (extrinsic) measure, especially when combined with a greedy learning strategy. However, we will see that such an MI formulation will reveal a combinatorial problem in pseudo-closure learning. The second and crucial contribution of this section is to show how to overcome this combinatorial problem, first by using the properties of V-type pretopological spaces and finally by approximations.

The multiple instance (MI) problem [30] arises when an observation can be described by several feature vectors, or instances. The set of vectors/instances

Bag	Shape	Size	Instance labels	Bag labels
1	Squared	Big	1	1
	Squared	Small	0	
	Squared	Medium	1	
2	Squared	Medium	1	1
	Triangular	Small	0	
3	Rectangular	Big	0	0
	Squared	Small	0	
	Triangular	Small	0	

Table 4: Example of a multiple instance dataset for the simple jailer problem.

describing an observation is called a *bag* of instances and is labeled either positively or negatively: positively if **at least one** instance is positive, negatively if **all** instances are negative. Bag labels may be assigned differently depending on the target task [30], but this simple bag labelling scheme, as proposed by Dietterich et al. [30], is well suited for the LPS task.

The MI task consists in finding a function predicting the label of a new instance knowing only the labels of the bags. A famous example of MI task is the *simple jailer problem* [24]: given a locked door and a bunch of key rings, the task is to find which key opens it. But the only information we have is whether a key ring is useful or not, i.e., whether it contains at least one key capable of unlocking the door. An example of a simple jailer dataset is provided in Table 4. Recall that in real cases, the column "Instance labels" is unknown and is presented here in order to understand why bags 1 and 2 are positive (because they have at least one positive instance). On the contrary, bag 3 is labeled negatively because it contains no positive instances.

We propose to model the task of learning a pretopological space (E, a) based on its expected set of elementary closed sets by an MI learning task. An instance represents a propagation from a set $A \in \mathcal{P}(E)$ to an element $x \in E$ and is labeled positively if $x \in F^*(A)$ according to S^*, otherwise the instance is labeled negatively. As we saw briefly before, multiple propagations of a set A conform to S^*. Thus, a bag corresponds to the set of instances modeling these propagations.

4.1. Building a MI dataset for the LPS task

The LPS task consists in learning a pretopological space (E, a_Q) of type V such that its set S_Q of elementary closed sets corresponds to a target set S^* of elementary closed sets. Let x be an element of E and its target elementary closure be $F^*(\{x\}) = \{x, y, z\}$. In general, several steps are necessary to compute an elementary closed set. For example, the valid steps for propagation from $\{x\}$ to $\{x, y, z\}$ are $\{x, y\}$, $\{x, z\}$ and $\{x, y, z\}$. If the pseudo-closure operator $a_Q(.)$ produces something different during the computation of $F_Q(\{x\})$, then the target elementary closed set $F^*(\{x\})$ cannot be retrieved.

The set of valid propagation steps for an element x are given by $\mathcal{X} = \{X \in \mathcal{P}(E) \mid x \in X, X \subseteq F^*(\{x\})\}$. The set of all propagation steps valid for all elements in E can thus be projected on the Boolean lattice of subsets of E. We denote this lattice $\mathcal{L}$ and define two *sub-lattice* notations for all sets A and B such that $A \subseteq B \subseteq E$.

$$\mathcal{L}\,[A, B] = \mathcal{F}(A) \cap \mathcal{P}(B)$$
$$\mathcal{L}\,[A, B[= \mathcal{L}\,[A, B] \setminus B$$

$\mathcal{L}\,[A, B]$ is defined as the intersection between the super-sets of A (i.e. the filter $\mathcal{F}(A)$) and the subsets of B (i.e. the power-set $\mathcal{P}(B)$), i.e. the sets between A and B in the lattice $\mathcal{L}$. $\mathcal{L}\,[A, B[$ is equal to $\mathcal{L}\,[A, B]$ deprived of its greatest element.

The LPS task can then be reformulated more precisely as the task of learning a pretopological space (E, a_Q) such that for any $x \in E$:

- Any set $A \in \mathcal{L}\,[\{x\}, F^*(\{x\})]$ propagates to **at least one element** of $F^*(\{x\}) \setminus A$.
- **No element** of $\overline{F^*(\{x\})}$ is reachable from a set $A \in \mathcal{L}\,[\{x\}, F^*(\{x\})]$.

This definitely sounds like the definition of positive and negative bags in an MI learning task! Indeed, it seems natural to formulate the LPS task as an MI learning problem. Each positive bag represents all allowed propagations of a given set $A \in \mathcal{P}(E)$, where A is a valid propagation step for an element $x \in E$. Therefore, positive bags are identified by a pair $(x, A) \in E \times \mathcal{P}(E)$ and mean "the element x must eventually reach all elements in A and at least one propagation described in this bag must occur". Each element of the sub-lattice $\mathcal{L}\,[\{x\}, F^*(\{x\})[$ gives rise to a positive bag engendered by x. The positive bag $(x, F^*(\{x\}))$ does not exist since, by definition of a closed set, $F^*(\{x\})$ does not propagate anymore.

A negative bag is identified by a couple $(x, y) \in E \times E$ and means "the element x must never propagate to y and none of the propagations described in this bag should occur". Any element not being part of the closed set of x gives rise to a negative bag engendered by x.

The set of all the positive and negative bags engendered by an element $x \in E$ are respectively noted $\text{bags}^+(x)$ and $\text{bags}^-(x)$.

$$\forall x \in E,\ \text{bags}^+(x) = \{(x, A) \mid \forall A \in \mathcal{L}\,[\{x\}, F^*(\{x\})[\}$$
$$\forall x \in E,\ \text{bags}^-(x) = \{(x, y) \mid \forall y \in E \setminus F^*(\{x\})\}$$

The number of positive and negative bags engendered by an element x is therefore given by:

$$\forall x \in E,\ \left|\text{bags}^+(x)\right| = 2^{|F^*(\{x\})|-1} - 1$$
$$\forall x \in E,\ |\text{bags}^-(x)| = \left|\overline{F^*(\{x\})}\right|$$

It is worth mentioning that in some cases where two (or more) elements $x, y \in E$ share the same elementary closed set $F^*_{xy} = F^*(\{x\}) = F^*(\{y\})$, some positive bags are engendered by both x and y. In fact, any positive bag (x, A) where $\{x, y\} \subseteq A \subset F^*_{xy}$ is engendered by both x and y. Such a positive bag can be designated as either by (x, A), (y, A) or $(\{x, y\}, A)$ to insist on the plurality of its origins. This is not a trivial property since it poses some problems of evaluation of a potential solution, as explained in the Appendix A.

Instances of bags represent a unique propagation from a subset $A \in \mathcal{P}(E)$ to an element $y \in E$. An instance is identified by the bag to which it belongs and a couple $(A, y) \in \mathcal{P}(E) \times E$. The notation (A, y) is rather confusing because it looks like the notation used to identify positive bags. To clarify this notation, we use (x, A) to identify positive bags and $A \to y$ to identify instances. Moreover, it clearly expresses the semantics of an instance, which is "does the set A propagates to the element y?". As we shall see, this can be answered either by "yes", "no" or "maybe".

A positive bag (x, A) contains all the instances modeling an allowed propagation from the set A to an element y. Such a propagation is allowed if and only if $y \in F^*(\{x\})$. Trivial propagations where $y \in A$ are ignored because they provide no useful information. Thus, any element $y \in F^*(\{x\}) \setminus A$ gives rise to an instance of the positive bag (x, A).

On the contrary, a negative bag (x, y) contains all instances modeling a forbidden propagation from any valid propagation step of x to the element y. Thus, any subset $A \in \mathcal{L}[\{x\}, F^*(\{x\})]$ gives rise to a negative instance of the bag (x, y).

Remark. *In the case of LPS, there are no negative instance of a positive bag, as in traditional MI learning tasks. Indeed, the instances of a positive bag are always positive too, in the sense that a model covering all these instances will perfectly retrieve the set S^* of target elementary closed sets. However, such a solution is only one solution among others perfectly retrieving S^*, since any solution covering at least one instance of each positive bag (and rejecting every negative bags) will also be a perfect solution. Our MI formulation allows the learning of any perfect solution by ignoring the siblings (instances of the same bag) of the covered positive instances.*

Instances of positive bags are always positive, but they do not need to be covered by a solution. However, all positive bags must be covered.

Let us illustrate this MI learning task using the set S^* of target elementary closed sets represented in Figure 8. The target closed set of the singleton $\{x_2\}$ is $F^*(\{x_2\}) = \{x_2, x_3, x_4, x_5\}$. As shown in Table 5, x_2 engenders one positive bag per element in $\mathcal{L}[\{x_2\}, F^*(\{x_2\})[$ and one negative bag per element not in $F^*(\{x_2\})$. The set $F^*(\{x_2\})$ is a closed set and must not propagate anymore, that is why no positive bag describes its propagation. The definition of a positive bag states that at least one of its instances is positive, which is why every instance of every positive bag is labeled as "maybe"; the only exception is when a bag contains only one instance, then its instance is obviously positive.

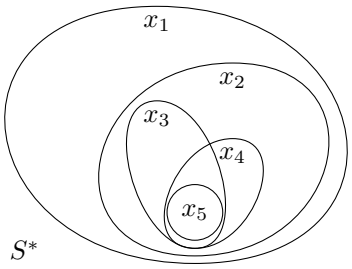


Figure 8: A set S^* of closed sets of a pretopological space of type V.

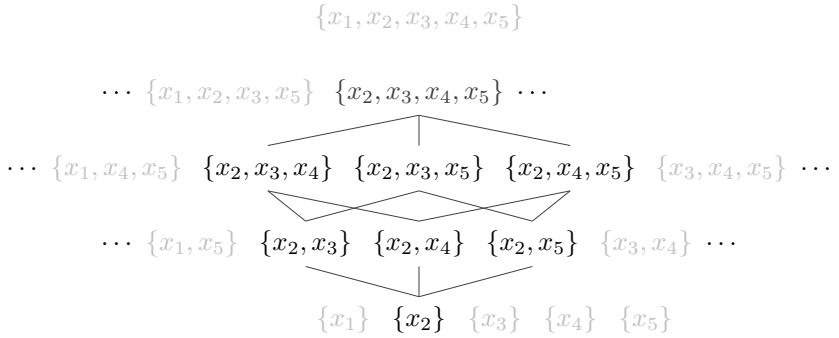


Figure 9: The sub-lattice $\mathcal{L}[\{x_2\}, F^*(\{x_2\})[$ embedded in the Boolean lattice on $E = \{x_1, x_2, x_3, x_4, x_5\}$. There is an exact correspondence between this sub-lattice and the positive bags engendered by x_2.

Figure 9 shows the sub-lattice $\mathcal{L}[\{x_2\}, F^*(\{x_2\})[$ which reflects the set of positive bags engendered by x_2. A line from a subset to another expresses that the source subset must propagate to a higher set in the lattice. Such lines are *transitive*, meaning that the result of the pseudo-closure operator applied on the subset $\{x_2, x_3\}$ must be equal to one of $\{x_2, x_3, x_4\}$, $\{x_2, x_3, x_5\}$ or $\{x_2, x_3, x_4, x_5\}$. This is exactly the definition of the positive bag $(x_2, \{x_2, x_3\})$, which states that $\{x_2, x_3\}$ must propagate to x_4, x_5 or both. Moreover, the absence of a line indicates that a propagation is forbidden, so, there is no line from a subset of $\mathcal{L}[\{x_2\}, F^*(\{x_2\})]$ to another one which contains x_1. This is the definition of the negative bag (x_2, x_1). Some propagations are impossible, for example $\{x_2, x_3\}$ cannot reach $\{x_3, x_4, x_5\}$ since the first is not included in the second. In this case, it is useless to insist on the absence of propagation. That is why it is not reported in the set of negative bags.

We have just explained how to construct an MI dataset for the LPS task, given a set E of elements and a set S^* of target elementary closed sets. But it is not yet possible to learn anything about this dataset due to the missing feature columns. The features are given by a collection of neighborhoods $\mathcal{V}$, more precisely by the predicates derived from these neighborhoods. Figure 10 shows a collection $\mathcal{V} = \{V_1, V_2, V_3, V_4\}$ of four neighborhoods. The set

Bag ID	Instance ID	Instance label	Bag label
$(x_2, \{x_2\})$	$\{x_2\} \to x_3$	maybe	1
	$\{x_2\} \to x_4$	maybe	
	$\{x_2\} \to x_5$	maybe	
$(x_2, \{x_2, x_3\})$	$\{x_2, x_3\} \to x_4$	maybe	1
	$\{x_2, x_3\} \to x_5$	maybe	
$(x_2, \{x_2, x_4\})$	$\{x_2, x_4\} \to x_3$	maybe	1
	$\{x_2, x_4\} \to x_5$	maybe	
$(x_2, \{x_2, x_5\})$	$\{x_2, x_5\} \to x_3$	maybe	1
	$\{x_2, x_5\} \to x_4$	maybe	
$(x_2, \{x_2, x_3, x_4\})$	$\{x_2, x_3, x_4\} \to x_5$	yes	1
$(x_2, \{x_2, x_3, x_5\})$	$\{x_2, x_3, x_5\} \to x_4$	yes	1
$(x_2, \{x_2, x_4, x_5\})$	$\{x_2, x_4, x_5\} \to x_3$	yes	1
(x_2, x_1)	$\{x_2\} \to x_1$	no	0
	$\{x_2, x_3\} \to x_1$	no	
	$\{x_2, x_4\} \to x_1$	no	
	$\{x_2, x_5\} \to x_1$	no	
	$\{x_2, x_3, x_4\} \to x_1$	no	
	$\{x_2, x_3, x_5\} \to x_1$	no	
	$\{x_2, x_4, x_5\} \to x_1$	no	
	$\{x_2, x_3, x_4, x_5\} \to x_1$	no	

Table 5: Positive and negative bags engendered by the element x_2 and its elementary closed set $F^*(\{x_2\}) = \{x_2, x_3, x_4, x_5\}$.

$\mathcal{Q} = \{q_1, q_2, q_3, q_4\}$ of four predicates can then be constructed by following the instruction of Equation (1) in Section 2.3.2. The Boolean feature vector $(q_1(A,x), q_2(A,x), q_3(A,x), q_4(A,x))$ is then constructed for each instance $A \to x$ of the dataset. Table 6 shows the bags engendered by x_2 as well as the feature vector associated to each instance.

Suppose the DNF $Q = (q_1 \wedge q_4) \vee (q_2 \wedge q_3)$ is learned by an unspecified MI learning process. The resulting pretopological space (E, a_Q) of type V would give the set $\{x_2, x_3, x_4, x_5\}$ as the elementary closure of x_2. It is obvious that $\{x_2\} \subseteq \{x_2, x_3, x_4, x_5\}$, the isotonic property of the pseudo-closure operator can be exploited to deduce that $a_Q(\{x_2\}) \subseteq a_Q(\{x_2, x_3, x_4, x_5\}) = \{x_2, x_3, x_4, x_5\}$. This means that the first step of the propagation of x_2 contains at least x_3, x_4 or x_5. It is essentially what the positive bag $(x_2, \{x_2\})$ expresses, so it is covered by Q. The same reasoning can be applied to explain why the positive bag $(x_2, \{x_2, x_3\})$ is covered: $\{x_2\} \subseteq \{x_2, x_3\}$ and $F_Q(\{x_2\}) = \{x_2, x_3, x_4, x_5\}$, so the isotonic property ensures that $\{x_2, x_3, x_4, x_5\} \subseteq F_Q(\{x_2, x_3\})$. Therefore, $a_Q(\{x_2, x_3\})$ contains either x_4 or x_5 (or even both). The positive bag $(x_2, \{x_2, x_3\})$ is thus covered. Of course, the same is true for the other positive bags.

The negative bag (x_2, x_1) is rejected by the solution Q because the set $\{x_2, x_3, x_4, x_5\}$ is a closed set in the pretopological space (E, a_Q) of type V. As a consequence of the isotonic property, any subset of $\{x_2, x_3, x_4, x_5\}$ cannot propagate to an element that is not in $\{x_2, x_3, x_4, x_5\}$. In particular, the subsets in the lattice $\mathcal{L}[\{x_2\}, F^*(\{x_2\})]$ do not propagate to x_1. Therefore, the negative bag (x_2, x_1) describing all the conceivable propagations from a subset in $\mathcal{L}[\{x_2\}, F^*(\{x_2\})]$ to x_1 is rejected by Q.

But what if the DNF $Q' = (q_1 \wedge q_4)$ is learned instead of Q? First, the elementary closed set of x_2 would be $\{x_2, x_4, x_5\}$. Can we ensure that the negative bag (x_2, x_1) remains rejected? This bag stipulates that any subset in $\mathcal{L}[\{x_2\}, F^*(\{x_2\})]$ must not propagate to x_1. For example, the set $\{x_3, x_2\}$ should not propagate to x_1. But x_3 is not reached by the propagation from x_2 to its elementary closed set. Therefore, the pseudo-closure of $\{x_3, x_2\}$, or a super-set, is never computed, which therefore makes it impossible to determine whether the negative bag (x_2, x_1) is badly covered or not.

One solution would be to compute the pseudo-closure of all subsets in the sub-lattice $\mathcal{L}[\{x_2\}, \{F_{Q'}(\{x_2\})\}]$ to check whether they propagate to x_1, but, as we explain in the following, it is not reasonable. We argue that, although this is indeed a flaw in our proposal, it is not dramatic in practice. Indeed, the ultimate goal of the LPS framework is to produce a DAG based on the **set of elementary closed sets** of the learned pretopological space. Thus, even if an erroneous propagation exists *beyond* an elementary closed set, it will not be present in the final structure. Therefore, it is safe to *assume* that such bags are rejected, as they should be.

4.2. The issue of the exponential MI dataset

As illustrated above, the Multiple Instance framework provides an elegant and accurate modeling for the non-trivial problem of learning a pretopological

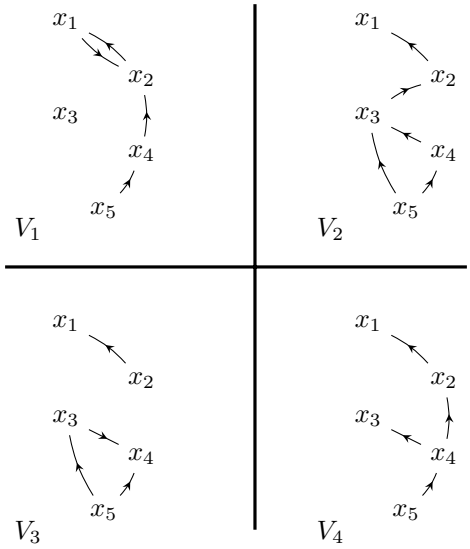


Figure 10: A collection of four neighborhoods.

Bag ID	Instance ID	q_1	q_2	q_3	q_4	Bag label
$(x_2, \{x_2\})$	$\{x_2\} \to x_3$	0	1	0	0	1
	$\{x_2\} \to x_4$	1	0	0	1	
	$\{x_2\} \to x_5$	0	0	0	0	
$(x_2, \{x_2, x_3\})$	$\{x_2, x_3\} \to x_4$	1	1	0	1	1
	$\{x_2, x_3\} \to x_5$	0	1	1	0	
$(x_2, \{x_2, x_4\})$	$\{x_2, x_4\} \to x_3$	0	1	0	0	1
	$\{x_2, x_4\} \to x_5$	1	1	1	1	
$(x_2, \{x_2, x_5\})$	$\{x_2, x_5\} \to x_3$	0	1	0	0	1
	$\{x_2, x_5\} \to x_4$	1	0	0	1	
$(x_2, \{x_2, x_3, x_4\})$	$\{x_2, x_3, x_4\} \to x_5$	1	1	1	1	1
$(x_2, \{x_2, x_3, x_5\})$	$\{x_2, x_3, x_5\} \to x_4$	1	1	0	1	1
$(x_2, \{x_2, x_4, x_5\})$	$\{x_2, x_4, x_5\} \to x_3$	0	1	1	0	1
(x_2, x_1)	$\{x_2\} \to x_1$	1	0	0	0	0
	$\{x_2, x_3\} \to x_1$	1	0	0	0	
	$\{x_2, x_4\} \to x_1$	1	0	0	0	
	$\{x_2, x_5\} \to x_1$	1	0	0	0	
	$\{x_2, x_3, x_4\} \to x_1$	1	0	0	0	
	$\{x_2, x_3, x_5\} \to x_1$	1	0	0	0	
	$\{x_2, x_4, x_5\} \to x_1$	1	0	0	0	
	$\{x_2, x_3, x_4, x_5\} \to x_1$	1	0	0	0	

Table 6: Bags engendered by the element x_2 and the features associated with each instance. The labels of the instance are hidden because they are not provided in a learning context.

space. However, since pretopology refers to the power-set $\mathcal{P}(E)$, the number of positive bags engendered by an element $x \in E$ is exponential (proportional to the size of its elementary closure $F^*(\{x\})$) thus revealing a major issue for the learning process.

We have just shown how to construct a dataset based on a target set of elementary closed sets. With such a dataset, one can apply standard MI algorithms to learn a solution from it. A standard MI approach would require enumerating all instances to count the number of bags covered, which will not be a problem with the example given above. But in real-world problems, the number of positive bags generated is overwhelming. Indeed, we explained that $\forall x \in E$, all sets belonging to the sub-lattice $\mathcal{L}[\{x\}, F^*(\{x\})[$ engender a positive bag. Such a number of bags cannot be handled efficiently by a standard learning algorithm, because it is exponential in the size of $F^*(\{x\})$.

A standard greedy MI algorithm would, for each learning step, collect the set of covered positive bags and remove them so that the next step would focus only on the remaining bags, and so on until all the positive bags are covered (or some other endpoint is reached). We note that such an algorithm does not really care whether know whether a given positive bag is covered or not. Rather, it cares about how many of them are covered, and whether the end criterion is met.

We therefore propose a method to count (or estimate) the number of bags covered by a solution. This method relies on a trick based on the properties of V-type pretopological spaces. Counting the number of negative bags covered is not a big problem, so we will not treat it in depth. On the other hand, counting the number of covered positive bags is a problem that requires much more effort. We show that while we are able to count the number of positive bags engendered, it is inefficient to accurately count the number of positive bags covered by a solution. Thus, we estimate this number by subtracting a (high) estimation of the positive bags rejected (i.e. not covered) by the solution from the number of positive bags in the dataset.

4.3. Estimation of true/false positives

In what follows, we propose a method for estimating the number of positive and negative bags covered by a solution. We begin by describing how to calculate the total number of positive bags in a dataset from the LPS task. This will serve as an introduction to estimating the number of positive bags covered by a solution (i.e. the number of true positives), since the counting mechanism is roughly similar. For completeness, we also show how to compute the total number of negative bags engendered. We then show how to compute the number of negative bags covered by a solution (i.e. the number of false positives).

4.3.1. Computation of the total number of positive/negative bags

The number of positive bags engendered by an element $x \in E$ corresponds to the number of elements in the sub-lattice $\mathcal{L}[\{x\}, F^*(\{x\})[$.

$$\forall x \in E,\ \left|\text{bags}^+(x)\right| = 2^{|F^*(\{x\})|-1} - 1$$

But the sum of the number of positive bags engendered by each $x \in E$ does not always correspond to the total number of positive bags. Indeed, when several elements share the same elementary closed set, some bags are engendered by several elements. For example, if $F^*(\{x\}) = F^*(\{y\}) = \{x, y, z\}$, then the positive bag identified by $(x, \{x, y\})$ is engendered by both elements x and y; we notice that this bag is also identified by $(y, \{x, y\})$.

Considering the set $\{F_1^*, \ldots, F_K^*\}$ of the K distinct elementary closed sets of E such that for any $x \in E$ there exists a $k \in \{1 \ldots K\}$ where $F^*(\{x\}) = F_k^*$; E can be partitioned into the set $\mathcal{A} = \{A_1, \ldots A_K\}$ of K equivalence classes where each class A_k is composed of the elements whose closure is F_k^* (i.e. $\forall x \in A_k$, $F^*(\{x\}) = F_k^*$).

For each equivalence class $A_k \in \mathcal{A}$, the number of positive bags engendered by the whole subset A_k is computed using the *inclusion-exclusion* principle.

$$\begin{aligned}\forall A_k \in \mathcal{A},\ \left|\text{bags}^+(A_k)\right| &= \sum_{i=1}^{|A_k|} (-1)^{i+1} \sum_{X \in \text{comb}(A_k, i)} \left| \bigcap_{x \in X} \mathcal{L}\left[\{x\}, F_k^*\right[\right| \\ &= \sum_{i=1}^{|A_k|} (-1)^{i+1} \binom{|A_k|}{i} (2^{|F_k^*|-i} - 1)\end{aligned}$$

where $\text{comb}(A_k, i)$ expresses the set of all i-combinations of A_k, or the power set of A_k reduced to its elements of size i. Thus, the size of the intersection between all sub-lattices $\mathcal{L}\left[X, A_k\right[$, where $|X| = i$ and $X \subseteq A_k$, is alternately added and subtracted. Since all the considered sub-lattices have a lower element of size i and share the same upper element F_k^*, they also share the same size, which is $2^{|F_k^*|-i} - 1$ thus simplifying the expression of $\left|\text{bags}^+(A_k)\right|$.

Property 3. *Let (E, a) be a pretopological space of type V, two sets $A, B \in \mathcal{P}(E)$, the sub-lattice $\mathcal{L}\left[A, F(A)\right]$ intersects $\mathcal{L}\left[B, F(B)\right]$ if and only if A and B share the same closure.*

$$\forall A \in \mathcal{P}(E),\ \forall B \in \mathcal{P}(E),\ \mathcal{L}\left[A, F(A)\right] \cap \mathcal{L}\left[B, F(B)\right] \neq \emptyset \Leftrightarrow F(A) = F(B) \quad (2)$$

Proof. Consider a pretopological space (E, a) of type V and two sets $A \in \mathcal{P}(E)$ and $B \in \mathcal{P}(E)$.

- If $F(A) = F(B) = K$, then $\mathcal{L}\left[A, F(A)\right]$ and $\mathcal{L}\left[B, F(B)\right]$ share at least their greatest element K. Therefore, $F(A) = F(B) \Rightarrow \mathcal{L}\left[A, F(A)\right] \cap \mathcal{L}\left[B, F(B)\right] \neq \emptyset$.
- If $\mathcal{L}\left[A, F(A)\right] \cap \mathcal{L}\left[B, F(B)\right] \neq \emptyset$, then $\exists C \in \mathcal{P}(E)$ such that $A \subseteq C$, $B \subseteq C$, $C \subseteq F(A)$ and $C \subseteq F(B)$. Then, by definition of a pretopological space of type V, $F(A) = F(C)$.

$$\begin{aligned}A \subseteq C \subseteq F(A) &\Rightarrow F(A) \subseteq F(C) \subseteq F(A) \\ &\Rightarrow F(A) = F(C)\end{aligned}$$

We can show that $F(B) = F(C)$ in the same way. Therefore, $\mathcal{L}[A, F(A)] \cap \mathcal{L}[B, F(B)] \neq \emptyset \Rightarrow F(A) = F(B)$.

□

Property 3 ensures that the total number $BAGS_*^+$ of positive bags can be computed by simply summing the number of positive bags engendered for each equivalence class.

$$BAGS_*^+ = \sum_{A_k \in \mathcal{A}} \left|\text{bags}^+(A_k)\right|$$

The number of negative bags is much simpler to compute since it does not require relying on the inclusion-exclusion principle. The number of negative bags engendered by an element $x \in E$ corresponds to the size of $\overline{F^*(\{x\})}$:

$$|\text{bags}^-(x)| = |E \setminus F^*(\{x\})|$$

The total number ($BAGS_*^-$) of negative bags is obtained by summing the number of negative bags engendered by each element

$$BAGS_*^- = \sum_{x \in E} |\text{bags}^-(x)|$$

4.3.2. Number of positive/negative bags covered by a solution

We have shown how to calculate the total number of positive and negative bags, without explicitly generating them. The number of positive bags covered by a solution Q, noted $BAGS_Q^+$, can be estimated by considering the number of bags covered by equivalence classes. For each equivalence class $A_k \in \mathcal{A}$, an estimate of the number $\text{bags}_\text{Q}^+(A_k)$ of positive bags covered is computed thanks again to the inclusion-exclusion principle.

$$\text{bags}_\text{Q}^+(A_k) = \sum_{i=1}^{|A_k|} (-1)^{i+1} \sum_{X \in \text{comb}(A_k, i)} \left|\text{bags}^+(X)\right| - r_Q^+(X)$$

where $\left|\text{bags}^+(X)\right|$ is the number of positive bags engendered by the elements in X, duplicates included; $r_Q^+(X)$ is the estimated number of positive bags that are (1) engendered by the elements in X and (2) rejected by Q. Then, the total number of positive bags covered can be estimated by summing $\text{bags}_\text{Q}^+(A_k)$ for each equivalence class. Estimating the number of positive bags not covered by a solution (the function $r_Q^+(.)$) is a complex task, so we prefer to detail this process in Appendix A.

$$BAGS_Q^+ = \sum_{A_k \in \mathcal{A}} \text{bags}_\text{Q}^+(A_k)$$

Counting the number of negative bags covered by a solution is a rather easy task, especially when compared to estimating the number of positive bags covered. This is mainly due to their small number: each element $x \in E$ engenders

$|E \setminus F^*(\{x\})|$ negatives bags. That is, one bag per element which does not belong to the elementary closure of $\{x\}$.

Given a DNF Q, the set $S_Q = \{F_Q(\{x\})\}_{\forall x \in E}$ is computed. The number of negative bags covered by Q for a given element $x \in E$ is equal to the number of unexpected elements in its closure, which is expressed by $|F_Q(\{x\}) \setminus F^*(\{x\})|$. The total number $BAGS_Q^-$ of negative bags covered by a solution Q can be computed by summing, for all elements, the number of negative bags covered by Q.

$$BAGS_Q^- = \sum_{x \in E} |F_Q(\{x\}) \setminus F^*(\{x\})|$$

4.4. The intrinsic quality measure

Based on the count of positive and negative bags covered by a DNF Q, we define a new quality measure taking into account both the retrieved elementary closed sets (as the extrinsic quality measure) and the internal behavior of the pretopological space (E, a_Q) of type V.

We designed our quality measure so that DNFs exhibiting a high number of positive bags covered and a low number of negative bags covered are promoted. Since the number of positive and negative bags is not of the same order of magnitude, we consider the $\log_2$ of the number of positive bags covered. Our quality measure is inspired by the *tozero score* proposed by Blockeel et al. [13], which is defined by:

$$h(Q) = \frac{\log_2(BAGS_Q^+)}{\log_2(BAGS_Q^+) + BAGS_Q^- + p}$$

where p is, as stated by Blockeel et al. [13], "a parameter that influences how strongly the measurement is pulled toward zero". Small values of k tend to favor high precision, since a small $BAGS_Q^-$ will have a larger impact, while greater values of k will be attenuated by a large $BAGS_Q^+$, thus favoring high recall.

We call it an *intrinsic quality measure* because it takes into account the result of the closure of any set included in the learned elementary closed sets (i.e. the lattices $\mathcal{L}[\{x\}, F^*(\{x\})]$) as opposed to the extrinsic quality measure which considers only the elementary closed sets.

4.5. Multiple Instance LPS

LPSMI is a multiple instance variant of Greedy LPS. It takes as input a set S^* of target elementary closed sets and a collection $\mathcal{V} = \{V_1, \ldots, V_k\}$ of k neighborhoods. LPSMI produces a positive DNF Q such that the elementary closed sets computed by the pseudo-closure operator $a_Q(.)$ best match S^*.

LPSMI builds a positive DNF by adding a new disjunctive clause to the DNF built in the previous iteration. LPSMI is implemented in the same spirit as Greedy LPS, they differ mainly in the quality measure they rely on. LPSMI finds the best clause based by relying on the intrinsic quality measure instead of

the extrinsic measure. Thus, we do not show pseudo-code of LPSMI because it would be identical to Greedy LPS (Algorithm 2) but with the extrinsic measure replaced by the intrinsic measure. Since the intrinsic quality measure is based on the number of bags covered (by a potential solution Q), it is in itself the reason why LPSMI belongs to the MI framework.

4.6. Scalability

As proposed in this paper, the main limitation of the LPS framework is scalability. Indeed, the evaluation of a pretopological space requires structuring it, which is often a very expensive operation. Since the solution space is huge, since it is the same size as the number of combinations of input neighborhoods, LPSMI has to repeat this costly structuring process many times. In addition, computing the number of positive bags covered can also be expensive, if many elements share the same elementary closed set.

The efficiency of LPSMI can therefore be massively improved either by speeding up the structuring algorithm or, better, by limiting the number of structurings. But these solutions often have a negative impact on the overall quality of the final pretopological space.

Given an already trained pretopological space (E, a) of type V, computing a single closed could be very expensive. In the worst case, all elements x of the set E share the same elementary closed set, which is the whole set E, and the computation of each elementary closed set is done one element at a time. The structure of such a space would be a ring. In this case, the computation of each elementary closed set would require applying the pseudo-closure operator $|E|$ times.

However, since we are currently considering only pretopological spaces of type V, the computation of elementary closed sets can easily be made faster by merging *compatible subsets of closed sets*. Suppose that somee parts of elementary closed sets have already been computed. For instance, the following two pseudo-closures are known:

- $a(\{x\}) = \{x, y\}$
- $a(\{y\}) = \{y, z\}$

We then know, without needing to apply it, that the second application of the pseudo-closure operator on $\{x\}$ will include $\{y, z\}$. It is therefore possible to skip a few steps of pseudo-closure, and go directly to the computation of $a(\{x, y, z\})$.

More formally, the closed set $F(A)$ of a subset $A \subseteq E$ includes the pseudo-closure of any subset $B \subseteq A$.

$$\forall A \in \mathcal{P}(E),\ \bigcup_{B \subseteq A} a(B) \subseteq F(A)$$

Actually, the following also applies:

$$\forall A \in \mathcal{P}(E),\, a\left(\bigcup_{B \subseteq A} a(B)\right) \subseteq F(A)$$

Clearly, it would not make sense to maintain the list of all subsets of $F(A)$. But maintaining the list of the k^{th} pseudo-closure step is feasible, k being the current step. The step $k+1$ can be approximated by the union of all compatible subsets in step k.

$$\forall x \in E,\, \forall k \in \mathbb{N},\, a\left(\bigcup_{\forall y,\, a^k(\{y\}) \subseteq a^k(\{x\})} a^k(\{y\})\right) \subseteq a^{k+1}(x)$$

Such an approximation has the potential to drastically reduce the number of pseudo-closure applications. For example, in the worst case, the number of computation step would go from $|E|$ to only $\log_2(|E|)$ per element.

However, the main concern comes from the learning algorithm, as we observed in practice that the propagation speed to the elementary closed sets is reasonable. The main issue comes from the fact that the solution space is very large, especially given the small size of the inputs. If minimizing the computing tine is important, a solution could be to limit the depth of the beam search strategy, which would lead to smaller conjunctive clauses. This would save time, but could also introduce many false positives. Another possible improvement to the current solution space exploration strategy could be to stop evaluating a conjunctive clause as soon as it moves too far away from the target, thus eliminating unnecessary pseudo-closure computations.

One could also sample the candidate conjunctive clauses in order to evaluate a fixed number of solution at each iteration of the training loop. But it is not clear how the sampling should be done, and it is not clear whether this can be done without eliminating too many candidates. Finding a cheap heuristic for sorting candidate clauses, as in the A^* algorithm [42], would certainly reduce the number of calls to the structuring method. But such a heuristic should not depend on elementary closed sets to be really efficient. Unfortunately, such a function is yet to be found.

As the time of writing, there is no satisfactory solution for efficiently learning a pretopological space. Laborde [47] proposes an alternative learning algorithm, but it remains rather restrictive since it assumes that there is an exact solution, which is far from realistic. Its proposal consists in first extracting the set of *forbidden conjunctions*. A forbidden conjunction is a conjunction which, if it were part of the DNF defining the pretopological space, would introduce false positives. Therefore, such clauses cannot be part of the final DNF if an exact solution is expected. The final DNF is constructed by gathering all the others conjunctive clauses into one large is disjunction. This algorithm is undoubtedly faster than LPSMI since it does not requires structuring the space at all, whereas LPSMI must do so for each candidate clause. However, it supposes that an

exact solution exists. Specifically, if a neighborhood contains a single incorrect connection, then it cannot be part of the resulting DNF. In practice, most neighborhoods contain errors, which is what motivated the combining approach of the LPS framework. Therefore, this algorithm, as presented, would surely produce an empty DNF when trained on real data. Nonetheless, this approach still holds great promise, but, while counter-intuitive, it must allow erroneous neighborhoods to be part of the final combination to be usable. One could argue that relaxing this constraint should not be difficult, since instead of prohibiting all clauses that engender at least one error, it should be possible to prohibit only clauses that engender *at least k* errors. This is true, but detecting an error is very cheap, whereas counting them requires structuring, or at least partial structuring, which would be contrary to the original intent.

5. Experiment I — Retrieval of an underlying pretopological model

This experiment consists of evaluating the performance of different algorithms on the task of retrieving a pretopological model when the data is known to be modeled by a pretopological space.

5.1. Learning an underlying pretopological model

We constructed a method for learning a propagation model based on the so-called intrinsic measure, as opposed to the extrinsic measure used in previous contributions. We assume that, because the intrinsic measure takes into account propagations on sets of elements (and not only on singletons), it better captures both the quality and the potential of a model under construction and is more suitable than the extrinsic measure. In order to validate these hypotheses, and the efficiency of the associated MI approach, we propose an experimental validation. This experiment consists in learning a propagation phenomenon, for example a forest fire. We seek to evaluate the performances of each LPS approach on the task of retrieving, or learning, an underlying pretopological space.

We consider a rectangular grid containing a set E of cells. We also consider a collection $\mathcal{V} = \{V_1, \ldots, V_8\}$ of eight neighborhoods, corresponding to the eight Moore neighborhoods represented in Figure 11. Given a known propagation model $a_{Q^*}(.)$, we construct the pretopological space (of type V) (E, a_{Q^*}). Our goal is to learn a positive DNF Q such that $a_{Q^*}(.)$ and $a_Q(.)$ produce similar closed sets.

We constructed three different models, aptly named according to our estimation of their learning difficulty:

- $Q^*_{Simple} = q_4 \vee q_6 \vee q_7$
- $Q^*_{Medium} = (q_4 \wedge q_6) \vee (q_5 \wedge q_8) \vee q_7$
- $Q^*_{Hard} = q_3 \vee q_5 \vee (q_2 \wedge q_4) \vee (q_4 \wedge q_7) \vee (q_6 \wedge q_7 \wedge q_8)$

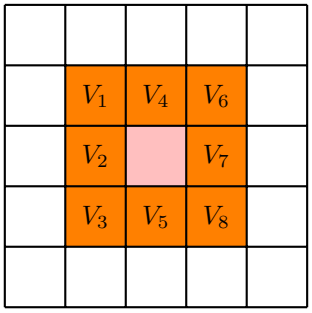


Figure 11: Moore's neighborhoods

We then derive three pseudo-closure operators (denoted $a_{Q^*_{Simple}}$, $a_{Q^*_{Medium}}$ and $a_{Q^*_{Hard}}$) to learn from these DNFs (Q^*_{Simple}, Q^*_{Medium} and Q^*_{Hard} respectively). Each pseudo-closure operator is used to model the propagation of a forest fire based on a percolation process [7]. Each DNF models a different wind-influenced propagation process. For example, Q^*_{Simple} models a forest fire under the influence of a southwest wind: a cell $x \in E$ burns when a cell located to its north, east or northeast burns, so the fire propagates to the southwest corner of the grid. Figure 12 shows the propagations obtained with Q^*_{Simple}, Q^*_{Medium} and Q^*_{Hard}, one clearly observes the fire modeled by Q^*_{Simple} propagating towards the south-west corner.

For each of these models, we constructed a set $\mathcal{G} = \{G_{10.i}\}_{i\in\{0...6\}}$ of seven 15×15 grids filled with an increasing percentage of obstructed cells. An obstructed cell is a cell $x \in E$ such that all its neighborhoods contain only itself and that the other neighborhoods do not contain this cell, i.e., $\forall V_i \in \mathcal{V}$, $V_i(x) = \{x\}$ and $\forall y \neq x, \forall V_i \in \mathcal{V}$, $x \notin V_i(y)$. Such a cell can neither be expanded nor reached by applying a pseudo-closure operator. Thus, G_0 contains 0 % of obstructed cells, G_{10} contains 10 % of obstructed cells, and so on unitl G_{60}. Let us specify that the obstructed cells in G_{i+1} are those of G_i plus additional 10 %. Then, for each grid $G_i \in \mathcal{G}$, we simulated the propagation of a fire ignited in each cell $x \in G_i$. The resulting set of burned cells was then assigned to the value of $F_{Q^*}(\{x\})$. Some of these simulations are shown in Figure 12. The green cells are combustible cells, the gray ones are obstacles (i.e. obstructed cells). The starting point of the fire is designated by the salmon-colored cell and burned areas by the red cells.

For each experiment we first picked 30 % of the inflammatory cells, and then computed their expansion (i.e., elementary closed sets) according to each proposed model to construct the set S^* of target elementary closed sets. Then, S^* and $\mathcal{V}$ were introduced as input to each of the LPS approaches — Genetic LPS (numerical and logical formalisms), Greedy LPS and LPSMI — to compare their results. Each algorithm was tested with two sets of parameters: the two Genetic LPS were tested with an initial population of 100 and 500 individuals; Greedy LPS and LPSMI were tested with a beam search of size 1 and 5. Finally, each experiment was performed ten times, so for each $Q \in \{Q^*_{simple}, Q^*_{medium}, Q^*_{hard}\}$,

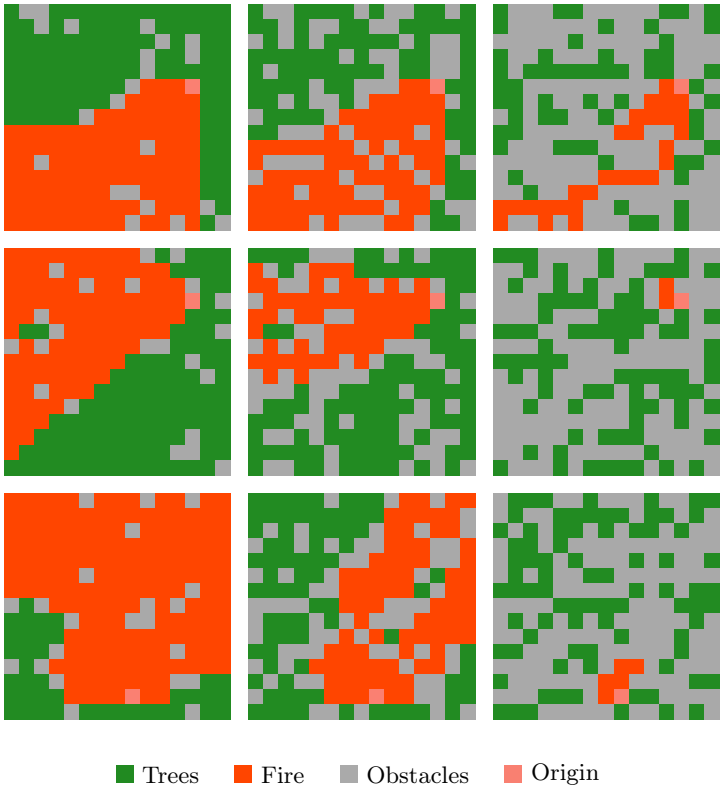


Figure 12: Result of 9 simulations: from top to bottom, fire propagation is modeled by, respectively, Q_{Simple}, Q_{Medium} and Q_{Hard}. From left to right, each grid is filled with 10%, 30% and 60% obstacles.

we built ten collections of seven grids. Figures 13 to 15 show the average results of each experiment: the F-measure obtained is indicated on the y-axis and the execution time is proportional to the size of each point (the smaller the better).

These graphs show that LPSMI gets the best score most of the time. We also notice that a larger beam size provides better results with a very slight increase in execution time: with a beam search of size 5, LPSMI is able to retrieve the entire target solution without errors (its F-measure is equal to 1)! Greedy LPS is the second best approach in terms of F-measure and offers quite similar performances. However, the scores obtained by Greedy LPS do not seem to be influenced by the size of its beam search, so unlike LPSMI, we cannot expect to obtain better results by simply expanding the beam. The quality of the result and the execution time of the Logical Genetic LPS are both very sensitive to the size of the initial population. Thus, one can expect excellent results if one's population is large enough, but at the cost of huge execution time. In some cases, it took hours to get a result while it was only a matter of seconds for

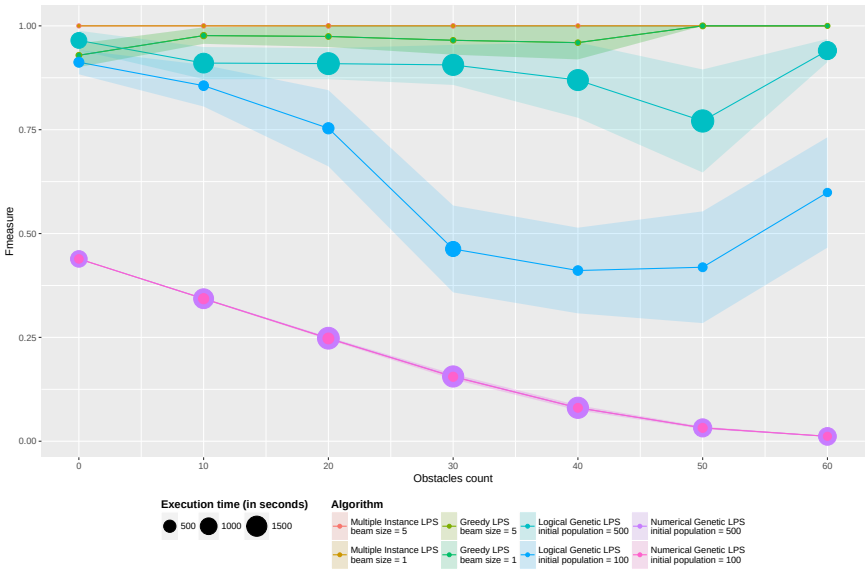


Figure 13: Performances of LPS algorithms on the learning task Q_{Simple}.

Greedy LPS and LPSMI. Finally, Numerical Genetic LPS performed the worst, both in terms of F-measure and execution time. The only difference between Numerical Genetic LPS and Logical Genetic LPS is the model they learned, the former learns a weight vector while the latter learns a positive DNF. We therefore attribute its poor performance to the inadequate and less expressive numerical and linear approach.

This experiment clearly illustrates and confirms the motivation of the present contribution: first, a positive DNF is more appropriate than a numerical vector to model an expansion process; second, it shows that while genetic algorithms have proven to be successful, their prerequisites (a huge initial population) are too large in our context; third, we compare two greedy algorithms: LPSMI and Greedy LPS which both learn a positive DNF but in different contexts. LPSMI is a multiple instance algorithm while Greedy LPS is a classical supervised algorithm; they use different quality measures: LPSMI finds the clause that maximizes our intrinsic quality measure while Greedy LPS relies on an extrinsic measure. This experiment shows that our MI approach retrieves a target model more efficiently than more classical approaches, regardless of the complexity of the model to be learned.

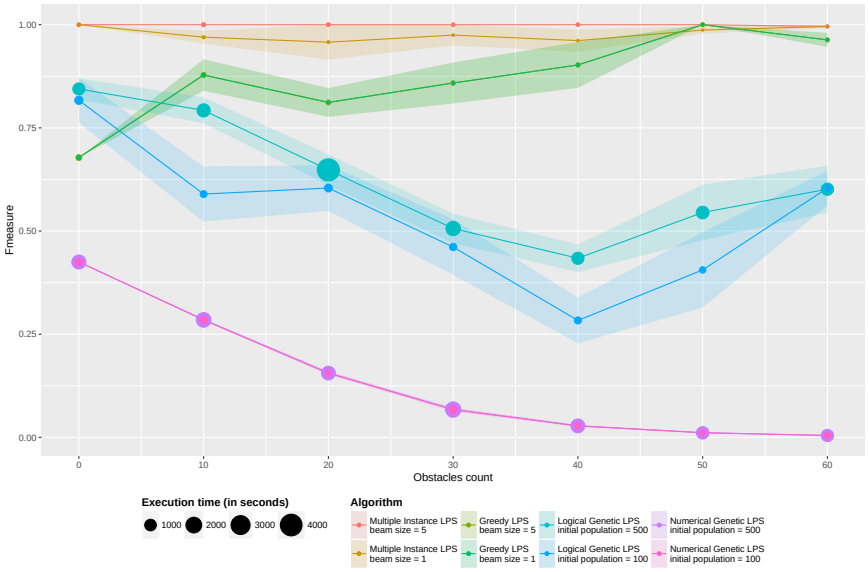


Figure 14: Performances of LPS algorithms on the learning task Q_{Medium}.

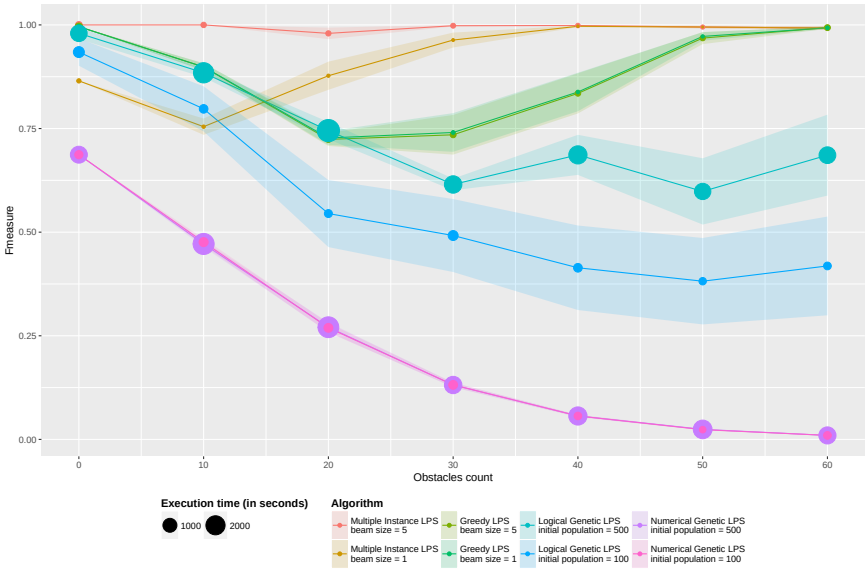


Figure 15: Performances of LPS algorithms on the learning task Q_{Hard}.

6. Experiment II — Learning lexical taxonomies with LPSMI

This experiment shows a concrete case of using the LPSMI algorithm in AI, where the goal is to learn a structuring model for the task of Lexical Taxonomy (LT) reconstruction.

6.1. Task and related works

The automatic extraction of lexical taxonomies is a widely studied research area. This task consists in structuring a set E of terms into a hierarchy of concepts according to the data found in a corpus C. Such a structure takes the form of a Directed Acyclic Graph (DAG) using hypernym-hyponym pairs $(x, y) \in E \times E$ such that x is an hypernym of y. The set E of terms can either be given as input or automatically extracted from the corpus C.

Semantic relations, such as the hypernymy relation, can be inferred quite effectively from term collocations. For example, Sanderson and Croft [67] proposed to compute statistics on words found in a corpus to infer subsumption relations between these words. Given two terms x and y, x *subsumes* y when x is highly likely to occur in a document when y is already known to occur in the same document. For example, if the term "tulip" is found in a document, then it is highly likely that the term "flower" is present as well, since one can hardly mention the concept of tulip without mentioning the broader concept of flower. The reverse is not true, since a document about flowers can refer to tulips, but also to roses or geraniums.

Hearst [43] proposed to extract hypernym-hyponym pairs by using carefully crafted patterns, such as "y is a x", where x and y are two terms. If a pair (x, y) instantiates such a pattern in the corpus, it means that x is likely to be a hypernym of y. This type of approach requires a set of input patterns, which can either be manually constructed [43, 46] or automatically extracted [39, 68].

Pattern-based approaches generally produce taxonomies with good edge accuracy, which means that an extracted hypernym-hyponym pair is usually correct. However, the edge coverage is usually low, which means that the taxonomy lacks many edges.

While these methods rely on raw text corpora to extract hypernym-hyponym pairs, other approaches take advantage of large semantic resources or networks, such as Wikipedia [59], WordNet [57] or BabelNet [58], to retrieve these pairs.

OntoLearn [72] relies on Word-Class Lattices to select candidate definitions, which are then filtered according to the domain of interest. Concepts and hypernym relations are extracted from this set of definitions. The graph of relations is then post-processed to restructure the graph into a tree (graph pruning) or even a DAG (edge recovery).

ExTaSem! [31] extracts candidate hypernyms by leveraging definitions found in BabelNet. This first step produces a set of noisy hypernym-hyponym pairs, which is passed to a second step exploiting syntactic features (from the textual definitions used earlier) to produce candidate paths from terms to their most specific hypernyms. These paths are then weighted according to the similarity

between the vectors trained with SenseEmbed [44], then filtered according to an empirical threshold. A lexical taxonomy finally emerges from the remaining paths.

MultiWiBi [34] proposes to build a *bitaxonomy* from all the pages and categories present in Wikipedia. A bitaxonomy is defined as a pair (T_P, T_C) where T_P is a taxonomy of Wikipedia pages and T_C a taxonomy of Wikipedia categories. MultiWiBi first builds a partial taxonomy of pages by extracting the hypernyms from the first sentence of each page. According to Wikipedia guidelines, the first sentence is supposed to be a short textual definition of the page title. The second phase of MultiWiBi is initialized with the pair (T_P, T_C) where T_P is the output of the first phase and T_C is a pruned version of the Wikipedia category taxonomy. Then, each taxonomy is used alternatively to enrich the second. For example, to enrich T_C, its concepts are first mapped to concepts in T_P, then the super-concepts are gathered by *climbing* the taxonomy and remapped to concepts in T_C.

The ContrastMedium algorithm, presented by Faralli et al. [32] also relies on semantic resources other than raw text corpora. This method provides a comprehensive pipeline for inducing a taxonomy by extracting structural information from a noisy semantic network linked to a reference (companion) taxonomy.

Knowledge graphs contain typed relations between entities, where an entity is a node of the graph and a relation is an edge. They are usually stored as triples (h, r, t) where h and t are two entities and r is a relation; h is called the *head* (the source node) and t is the *tail* (the destination node). A lexical taxonomy can be seen as a knowledge graph restricted to hypernymy relations (or more generally, to semantic relations). As such, work done on knowledge graphs could be applied to lexical taxonomies.

Recent work proposes to learn embeddings for both entities and relationships in a vector space [73]. These embeddings have proven useful for detecting and capturing typed relations between entities. Entity embeddings are often vectors capturing a set of latent features, while relation embeddings can be vectors or matrices. In what follows, entities are noted in normal font (x) and their embeddings with the same letter in bold face font ($\boldsymbol{x}$).

A lot of work has been done on translation-based models, such as TransE [18]. These approaches share a common principle: given a triple (h, r, t), the embedding $\boldsymbol{t}$ must be *not too far* from $\boldsymbol{h} + \boldsymbol{r}$. Therefore, these approaches often learn embeddings such that the distance between $\boldsymbol{t}$ and $\boldsymbol{h} + \boldsymbol{r}$ is minimized.

$$\sum_{(h,r,t) \in S} \|\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t}\|_2^2$$

where S is the set of training triples.

In TransE [18], both entity and relation embeddings are trained in the same space, and a triplet (h, r, t) is modeled by the linear transformation $\boldsymbol{t} \approx \boldsymbol{h} + \boldsymbol{r}$. It appears that such a simple modeling is quite powerful and performs almost as well as more expressive models, such as RESCAL [60] or SME [17], which

require much more computational power. However, TransE cannot reliably learn many-to-one, one-to-many and many-to-many relations. Given an entity h and a relation r, if they appear in at least two triples (h, r, t_1) and (h, r, t_2), then t_1 and t_2 must share the same embeddings, even though they are very different entities.

Other approaches focus on modeling semantic relations by learning word embeddings from large corpora. Word embeddings are trained, most of the time, according to the distributional hypothesis [40], which states that semantically similar terms appear in similar contexts. Therefore, term embeddings appearing in similar contexts are close to each other.

Modeling semantic relations between word embeddings can be done in the same way as modeling typed relation between entity embeddings. These approaches typically rely on pre-trained word embeddings on large-scale corpora, such as word2vec [55] or GloVe [64], which have shown great abilities to preserve semantic relations. For example, Mikolov et al. [56] and Levy and Goldberg [51] have shown that some relations can be modeled as simply as shift vectors. Pocostales [65] proposed to apply this principle to extract hypernym-hyponym pairs. His protocol is quite simple: given a set of known hypernym-hyponym pairs (x, y), a hypernym vector $\overline{\boldsymbol{h}}$ is computed as the average offset between each vector representation: $\boldsymbol{x}$ and $\boldsymbol{y}$.

Fu et al. [37] proposed a more fine-grained approach where multiple projection matrices, instead of a single hypernymy vector, are trained based on the relative positions of the two embeddings. Given a word embedding $\boldsymbol{y}$ and a projection matrix ϕ, the hypernym embedding of y should be located near $\phi(\boldsymbol{y})$, the projection of $\boldsymbol{y}$ by ϕ.

These methods are similar to the translation-based methods presented above, except that the relation embeddings are computed after entity/word embeddings are learned, and not at the same time. As such, one might expect better quality of embeddings from the translation-based approaches. However, it is difficult, if not impossible, to reuse these embeddings to discover missing relations in a knowledge graph different from the learning one. Pre-trained word embeddings are much more versatile since they are not tied to specific corpora. The meaning of words often remains the same in different corpora, so word embeddings can be reused on different corpora. On the other hand, entity embeddings are most often linked to the knowledge graph they belong to, which makes it more difficult to transfer them to other graphs.

While each approach is capable of extracting valuable pieces of information, none of them exhaustively models the complex hypernym-hyponym semantic relationship. However, each of them captures a piece of the truth. Our postulate is that a wise combination of them could allow a more accurate modeling of the hyponym-hypernym relationship between terms in a given domain. Cimiano et al. [26] has already shown that, even with a very simple combination strategy, combining multiple data sources is very profitable.

Largeron and Bonnevay [48] showed that the underlying structure of a V-type pretopological space is a DAG, thus matching with the structural expectation of

a lexical taxonomy. Next, Cleuziou and Dias [28] made use of the (multi-criteria aspect of the) pretopology theory for the LT *extraction* task. They define, successfully, a semi-supervised framework for automatically learning pretopological spaces that structure a set of terms into a DAG.

In the present experiment, we propose to apply the LPSMI algorithm for the LT task *reconstruction*. Given a set E of terms and a target (also called reference) lexical taxonomy T of terms in E, our goal is to learn a model that structures E as T. The LPSMI algorithm produces a V-type pretopological space (E, a_Q) where E is a list of terms of a given domain and $a_Q(.)$ is a logical pseudo-closure operator defined by a DNF Q and modeling the propagation of the hypernymy relation from one subset of terms to another.

The objective of this experiment is mainly to show the **potential** of the tools offered by the pretopology theory to model complex semantic relations; we focus on particular on the potential of the newly proposed LPSMI algorithm. Our goal is not to compete with existing LT extraction approaches. Instead, we propose a versatile and highly extensible framework taht allows for the combination of multiple, heterogeneous approaches, and we show that this combination outperforms each individual approach. We furthermore investigate whether a pretopological model learned on a set E of terms can be applied on a more generic domain and/or a larger set E' of terms, thus considering not only the task of LT *reconstruction* but also the more exciting task of LT *extraction*.

6.2. Outline of the experiment

Given a set E of terms in a domain and a target lexical taxonomy T to reconstruct, we propose to learn a semantic propagation model that structures E into a DAG as similar to T as possible. We consider that a given term $x \in E$ propagates its meaning to all of its descendants in T (transitivity). Thus, for each term $x \in E$, its target elementary closure, $F^*(\{x\})$, is the set of its hyponyms, i.e. x itself and the set of terms below x in T. Let $S^* = \{F^*(x)\}_{x \in E}$ the set of all target elementary closed sets, the LPSMI algorithm is designed for the task of learning a pseudo-closure operator $a_Q(.)$ modeling the propagation of the hypernymy relation.

The LPSMI algorithm offers an elegant framework for combining several state-of-the-art hyponym/hypernym extraction methods within a single pretopological process. We argue that this allows our model to more accurately capture the propagation of semantic relations between sets of terms. To accomplish this task, several approaches (algorithms) dedicated to hyponym/hypernym extraction are considered, their results are converted into neighborhood relations, and then used as inputs of the LPSMI algorithm. Given a hyponym/hypernym extraction method M, a neighborhoods relation R_M is defined such that for all terms $x, y \in E$, xR_My means that x is a direct hypernym of y according to the relation R_M. The neighborhood $V_M(y)$ of the term y is given by

$V_M(y) = \{y\} \cup \{x \in E \mid xR_My\}$. Thus, $V_M(y)$ contains y[4] and its (direct) hypernyms according to the hyponym/hypernym extraction method M.

The LPSMI algorithm is highly generic and extensible in the sense that any hyponym/hypernym extraction method (existing or future) can be easily incorporated into the LPS framework to serve the learning of the pseudo-closure operator. Our approach could be even more flexible by considering predicates that are not necessarily defined by neighborhood relations, as proposed by Caillaut et al. [22].

The experiment is conducted to validate the following three hypothesis:

1. **the benefits of the multi-criteria combination**. We believe that the combining existing methods will produce better results. Therefore, we compare the lexical taxonomies extracted by individual methods with those obtained by combining all the methods considered in the LPSMI process;

2. **the relevance of pretopology theory** for modeling lexical taxonomies. To this end, we compare the results obtained with pretopological modeling to those obtained with classical decision making methods (Support Vector Machine and Decision Tree);

3. **the model-domain dependency**. The taxonomy of a given domain E should be more easily retrievable by a model trained on a subdomain of E than another model trained on a completely disjoint domain. This would show that each domain has its own "implementation" of the same semantic relation.

6.3. The set of neighborhood relations

In the following experiment, five state-of-the-art approaches are considered that result in five neighborhood relations.

$R_{Sand.}$. A first relation, $R_{Sand.}$, is inspired by the work of Sanderson and Croft [67]. The authors propose to build a concept hierarchy based on the collocation probability of each pair of a set of terms. A term x is assumed to subsume another term y if the following property holds: $P(x \mid y) \geq 0.8 \wedge P(y \mid x) < 1$. Our relation $R_{Sand.}$ differs from the original work by two aspects:

- the set of terms is known in advance and not extracted automatically,
- we do not rely on a manually defined threshold.

The relation $R_{Sand.}$ is built on a set of terms from a common domain, and thus known to be somehow semantically related. Furthermore, we know that our target structures are trees-like: a term y usually has a unique direct hypernym

[4]The reflexivity of the neighborhood function is imposed by the pretopological formalism.

x. This means that given a set E of terms, there are approximately $|E|$ pairs (x, y) where x is a direct hypernym of y. This explains our choice to keep only the $|E|$ most probable relations. So, given a set of pairs (x, y) of terms in the set E, a relation $xR_{Sand.}y$ exists if the following two conditions are satisfied :

$$x\ R_{Sand.}\ y \Leftrightarrow \begin{cases} P(y \mid x) < 1 \text{ and} \\ |\{(x_i, y_i) \in E \times E \mid P(x_i \mid y_i) \geq P(x \mid y)\}| \leq |E| \end{cases}$$

In the present application, the collocation probabilities are computed from the english Wikipedia[5] dump dated October 1, 2020: two terms are collocated if they appear in the same Wikipedia page.

$R_{patterns}$. The second relation $R_{patterns}$ is based on syntactic pattern extraction methods [43]. Six lexical patterns, known for their accuracy [46], are considered: "y is a x", "y is an x", "y are x that", "x such as y", "x like y", "x including y". Two terms x and y are in relation $xR_{pattern}y$ if and only if they instantiate at least one of the four patterns on a given corpus. In the following experiment, we extract these patterns from the same Wikipedia corpus as for the relation $R_{Sand.}$.

$R_{strmatch}$. The relation $R_{strmatch}$ is based on the computation of a similarity on the character strings of two terms. Many terms actually look like their direct hypernym: for example "craft" is an hypernym of "aircraft". Given two terms x and y, we split each word into 3-grams and compute the ratio of tokens in x that are also in y. Two terms are in relation $xR_{strmatch}y$ if and only if 80 % of the tokens from x are included in those from y.

The following two relations are based on word embeddings. Given a term y and its corresponding embedding $\boldsymbol{y}$, we assume that a linear transformation h such that $h(\boldsymbol{y}) \approx \boldsymbol{x}$ exists, where $\boldsymbol{x}$ is the embedding of a hypernym x of y. We trained these transformations on a set of pre-trained embedding of *word2vec*[6].

R_{meanH}. The first embedding-based relation, R_{meanH}, is constructed by applying a translation on the embedding of each term, as proposed by Pocostales [65].

Given a target lexical taxonomy T, we first extract its set of direct hypernym-hyponym pairs (x_i, y_i) and their corresponding embedding pairs $(\boldsymbol{x_i}, \boldsymbol{y_i})$. We assumed that the hypernymy relation is preserved by these word embedding representations and can be retrieved by computing a shift between the hyponym embedding $\boldsymbol{y_i}$ and the hypernym embedding $\boldsymbol{x_i}$. For each pair of embedding $(\boldsymbol{x_i}, \boldsymbol{y_i})$ found in the previous step, a hypernymy vector $\boldsymbol{h_i}$ is computed: $\boldsymbol{h_i} = \boldsymbol{x_i} - \boldsymbol{y_i}$.

[5] `https://www.wikipedia.org/`

[6] `https://code.google.com/archive/p/word2vec/`

Finally, the average hypernymy vector is computed: $\overline{\boldsymbol{h}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{h_i}$ (with n the number of paris extracted from T). $\overline{\boldsymbol{h}}$ is then used to predict the direct hypernym x for each new term y. Two terms x and y are related $xR_{meanH}y$ if and only if $\boldsymbol{x} = \arg\min_{z \in E} \left\| \boldsymbol{z} - (\boldsymbol{y} + \overline{\boldsymbol{h}}) \right\|_2$; the predicted hypernym of y is the term from E whose embedding is closest to $\boldsymbol{y} + \overline{\boldsymbol{h}}$. In order to limit the false positive rate, we also introduced a maximum distance beyond which two terms are not considered as semantically linked. This threshold is set to 0.5 in our experiments.

$\boldsymbol{R_{Fu}}$. The second relation based on word embeddings, R_{Fu}, relies on the work of Fu et al. [37]. The authors assume that the hypernymy relation between two terms x and y dependends on the relative position of their respective embeddings $\boldsymbol{x}$ and $\boldsymbol{y}$.

They propose to compute a clustering of hypernym-hyponym pairs $(\boldsymbol{x}, \boldsymbol{y})$ according to the offset vector $\boldsymbol{x}-\boldsymbol{y}$. Then, for each cluster i, they train the linear projection ϕ_i projecting the embedding of an hyponym $\boldsymbol{y}$ to the embedding of its hypernym.

A term x is considered a hypernym of another term y if the embedding $\boldsymbol{x}$ is close to $\phi_i(\boldsymbol{y})$, where i is the cluster minimizing the euclidean distance between $\boldsymbol{x} - \boldsymbol{y}$ and its center. We used k-means to cluster the set of pairs into k groups. It turned out that $k = 5$ was sufficient because our datasets were small enough. We set the threshold used to determine whether $\boldsymbol{x}$ is close enough to $\phi_i(\boldsymbol{y})$ to 1. This threshold was found empirically and may be inappropriate if applied to other data. Finally, the relation R_{Fu} can be defined as below:

$$\forall x \in E,\, \forall y \in E,\, xR_{Fu}y \Leftrightarrow \|\boldsymbol{x} - \phi_{i^*}(\boldsymbol{y})\|_2 \leq 1$$

where $i^* = \arg\min_{i \in 1...k}(\|c_i - (\boldsymbol{x} - \boldsymbol{y})\|_2)$, with c_i the center of the cluster i.

6.4. Experimental Data and Results

We applied the LPSMI algorithm for the reconstruction of four domains extracted from WordNet [57]: *vehicles* (108 terms), *plants* (554 terms), *food* (1486 terms) and *animals* (688 terms). Three sub-domains per domain were also extracted, as reported in Table 7.

We fed the LPSMI algorithm with a set of five predicates defined by the neighborhood relations described in the previous sections, as explained in Section 2.3.2. The three relations $R_{Sand.}$, $R_{patterns}$ and $R_{strmatch}$ are acquired, each, by an unsupervised data extraction process from raw texts. We consider here the Wikipedia corpus. On the other hand, both R_{meanH} and R_{Fu} must be trained. Depending on the objective, they will be trained on different datasets. In the case of the task *taxonomy reconstruction* (Section 6.4.1), whose objective is to train a model to extract a given taxonomy, the train and test datasets are the same. This task is a preliminary to the more interesting *taxonomy extraction* task (Section 6.4.2), whose objective is to extract a new taxonomy by applying previously trained models.

Full domains	Sub-domains	Size	Edges	Transitive edges	Depth
vehicles	vehicles	108	109	413	8
	wagons	7	6	10	4
	crafts	35	34	103	6
	motor vehicles	30	30	57	5
plants	plants	554	553	2294	12
	bulbous plants	28	27	56	4
	aquatic plants	22	21	32	4
	grasses	23	22	40	5
food	food	1486	1527	6955	9
	candy	63	62	85	4
	bread	113	112	229	5
	snack food	24	23	45	4
animals	animals	688	689	4330	14
	reptiles	39	130	130	6
	fish	56	240	240	7
	birds	119	353	353	6

Table 7: Sizes of the different extracted sub-domains

R_{meanH} and R_{Fu} are trained on a large set of embeddings, which cover a large portion of the terms present in our three datasets (some terms are missing, especially multi-word terms). Since we use the same set of embeddings to train these relations, regardless of the target domain, a relation trained on one domain can be transferred as is to predict relations on another domain. For example, an average hypernym vector learned on *crafts* can be transferred as is to the domains *vehicles* or *plants*. This is not the case with modern embedding-based approaches, such as TransE [18], which are not designed to reuse embeddings on a different task. Therefore, embeddings trained on the domain *vehicles* could not be transferred to structure the domain *food*, mainly because the terms in *food* are not part of the domain *vehicles*. This is why such approaches are excluded from our study.

We modified the stopping criterion of the LPSMI algorithm to stop adding clauses to the logic formula when the correspondence with the target elementary closed sets starts to decrease (i.e. when the F-measure of the solution decreases).

6.4.1. Learn the hypernymy relation of a given domain

We used LPSMI to learn a pretopological model on each subdomain of Table 7 by combining the neighborhood relations described in the previous subsection. We use the recall, precision, and F-measure to evaluate the quality of each model. Recall and precision are defined in terms of the number of well retrieved relations, based on the domain reference taxonomy. A hypernym-hyponym pair (x, y) is retrieved if and only if y appears in the learned elementary closure

$F_Q(\{x\})$. Such a relation means that the model has learned that x is an hypernym of y. This relation is correct if the same relation exists in the reference taxonomy, i.e. if $y \in F^*(\{x\})$.

We made the decision to ignore reflexive relations (x is a hypernym of itself), which are required by the pretopology formalism, as this would excessively inflate the F-measure and thus not reflect the actual performance of our models. Therefore, we define the F-Measure as follows:

$$\text{Recall} = \frac{\sum_{x \in E} |F^*(\{x\}) \cap F_Q(\{x\})| - |E|}{\sum_{x \in E} |F^*(\{x\})| - |E|}$$

$$\text{Precision} = \frac{\sum_{x \in E} |F^*(\{x\}) \cap F_Q(\{x\})| - |E|}{\sum_{x \in E} |F_Q(\{x\})| - |E|}$$

$$\text{F-Measure} = 2 \cdot \frac{P + R}{P \cdot R}$$

First and foremost we show that the combination of various lexical taxonomy extraction methods within a structuring model learned by the LPSMI algorithm increases the quality of the output taxonomy. Each neighborhood relation was used to construct a lexical taxonomy for each domain and subdomain, resulting in five taxonomies per subdomain. These taxonomies are extracted by first transposing the adjacency matrices of the neighborhoods, and then computing their transitive closures. We compared the quality of these taxonomies with the taxonomies extracted by applying LPSMI. The results in Table 8 clearly show that the taxonomies extracted by a pretopological combination of neighborhood relations outperform each of the lexical taxonomies constructed from a single data source.

These results experimentally validate the first hypothesis on the benefits to be expected from a multi-criteria combination. Furthermore, in order to evaluate the relevance of the *pretopological* combination itself, we seek to compare two approaches for learning structures: on the one hand, (supervised) *learning of a structuring model* (e.g. LPS) and on the other hand, (supervised) *learning of relations* (e.g. by traditional classifiers such as Support Vector Machine (SVM) or Decision Tree (DT)).

Although recent work tends to rely on neural Language Models to learn semantic relations, the comparison with SVM and DT is actually appropriate to demonstrate the benefits of LPSMI. LPSMI and SVM indeed share some common behaviors since they can both be used to learn non-linearly separable classes. SVM does this by applying a kernel mathematical function to its input while LPSMI relies on a learning criterion similar to the information gain used in DT. However, learning a structuring model, as LPSMI does, allows to integrate prior constraints on the final induced structures; for example, the pretopological (V type) formalism on which the LPSMI method is based guarantees the generation of structures in the form of DAGs. In comparison, approaches that learn relations do not allow to integrate these constraints; the relations between elements are predicted independently of each other, and the structure derived from these relations does not offer any guarantee to satisfy the expected

	Sanderson	patterns	meanH	Fu	strmatch	LPSMI
vehicles	0.28^{4}	0.25^{5}	0.34^{3}	0.51^{2}	0.12^{6}	$\mathbf{0.62^{1}}$
wagons	0.82^{2}	0.00^{6}	0.57^{4}	0.67^{3}	0.18^{5}	$\mathbf{0.95^{1}}$
crafts	0.38^{4}	0.29^{5}	0.40^{3}	0.69^{2}	0.21^{6}	$\mathbf{0.80^{1}}$
motor vehicles	0.36^{2}	0.03^{6}	0.29^{4}	0.35^{3}	0.16^{5}	$\mathbf{0.67^{1}}$
plants	0.25^{3}	0.12^{4}	0.04^{6}	0.47^{2}	0.05^{5}	$\mathbf{0.53^{1}}$
bulbous plants	0.06^{3}	0.00^{6}	0.04^{5}	0.06^{3}	0.07^{2}	$\mathbf{0.12^{1}}$
aquatic plants	$\mathbf{0.29^{1}}$	0.06^{2}	0.05^{4}	0.00^{5}	0.00^{5}	0.06^{2}
grasses	0.14^{6}	0.44^{3}	0.35^{5}	0.70^{2}	0.37^{4}	$\mathbf{0.86^{1}}$
food	0.21^{3}	0.06^{5}	0.02^{6}	0.25^{2}	0.12^{4}	$\mathbf{0.39^{1}}$
candy	0.37^{3}	0.00^{6}	0.37^{3}	0.41^{2}	0.32^{5}	$\mathbf{0.71^{1}}$
bread	0.31^{3}	0.00^{6}	0.23^{5}	0.31^{3}	0.38^{2}	$\mathbf{0.66^{1}}$
snack food	0.22^{4}	0.00^{6}	0.27^{3}	0.30^{2}	0.19^{5}	$\mathbf{0.45^{1}}$
animals	0.14^{4}	0.19^{3}	0.08^{5}	$\mathbf{0.53^{1}}$	0.05^{6}	0.35^{2}
reptiles	0.25^{4}	0.27^{3}	0.16^{5}	0.43^{2}	0.06^{6}	$\mathbf{0.53^{1}}$
fish	0.29^{3}	0.14^{5}	0.21^{4}	0.48^{2}	0.13^{6}	$\mathbf{0.56^{1}}$
birds	0.18^{4}	0.30^{3}	0.15^{5}	0.58^{2}	0.15^{5}	$\mathbf{0.66^{1}}$
Average rank	3.31	4.62	4.38	2.38	4.81	**1.12**

Table 8: Comparison of F-measures obtained by individual state-of-the-art methods with the (combining) LPSMI approach. The scores are labeled according to their rank.

properties (e.g. absence of cycle).

The consideration of the global structure in the learning process thus distinguishes the learning of structuring models from the learning of relations. For example, SVM or DT classifiers "decide" a relation on a pair of elements using only the description of this pair. On the contrary, the LPS method allows to "contextualize" this decision rule by relying on all the elements of the closure under construction.

In the following experiment, we use Support Vector Machine (SVM) and Decision Tree (DT) the implementations provided respectively, by the R packages *e1071*[7] and *rpart*[8]. We have performed two experiments per learning algorithm: the first one aims at comparing the performances of models learned by LPSMI, SVM and DT on the same (binary) relations. The second one aims at discovering if better results can be obtained without the *thresholding* step required by LPSMI, i.e. with continuous data instead of binary neighborhoods.

We found out that performance of the two types of models (binary and continuous) were not significantly different. In fact, the SVM trained on binary data performed slightly better than its counterpart trained on continuous data, while the DT performed better with continuous data. In order to remain con-

[7] `https://CRAN.R-project.org/package=e1071`
[8] `https://CRAN.R-project.org/package=rpart`

	SVM			Decision Tree			LPSMI		
domain	R	P	F	R	P	F	R	P	F
vehicles	0.44	0.92	0.59	0.43	0.98	0.60	0.45	0.97	**0.62**
wagons	0.90	1.00	**0.95**	0.70	1.00	0.82	0.90	1.00	**0.95**
crafts	0.67	0.88	0.76	0.79	0.69	0.73	0.71	0.91	**0.80**
motor vehicles	0.64	0.72	**0.68**	0.22	1.00	0.36	0.69	0.64	0.67
plants	0.34	0.74	0.47	0.34	0.74	0.47	0.43	0.67	**0.53**
bulbous plants	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.50	**0.12**
aquatic plants	0.03	1.00	**0.06**	0.03	1.00	**0.06**	0.03	1.00	**0.06**
grasses	0.78	0.97	**0.86**	0.78	0.97	**0.86**	0.78	0.97	**0.86**
food	NA	NA	NA	NA	NA	NA	0.26	0.81	**0.39**
candy	0.58	0.92	**0.71**	0.58	0.92	**0.71**	0.58	0.92	**0.71**
bread	0.52	0.91	**0.66**	0.52	0.91	**0.66**	0.52	0.91	**0.66**
snack food	0.29	0.81	0.43	0.29	0.81	0.43	0.31	0.82	**0.45**
animals	0.53	0.70	**0.60**	0.38	0.85	0.53	0.22	0.88	0.35
reptiles	0.32	0.95	0.48	0.28	1.00	0.43	0.37	0.96	**0.53**
fish	0.47	0.68	0.55	0.37	0.68	0.48	0.48	0.68	**0.56**
birds	0.50	0.94	**0.66**	0.50	0.94	0.65	0.51	0.93	**0.66**

Table 9: Recall, precision and F-measure scores obtained by models learned by SVM, Decision Tree and our proposal, LPSMI, for the task of capturing the hypernymy relation of a given domain.

sistent with LPSMI, we present only results obtained by the models trained on binary data.

Since SVMs and DTs expect tabular data as input instead of a collection of neighborhoods, we transformed our datasets such that the instances are pairs (x, y) and their class labels are 1 if x is a hypernym (direct or not) of y, 0 otherwise. Each instance has five features corresponding to the five neighborhoods. Since, unlike a pretopological model, SVM and DT do not mimic a propagation process and therefore cannot reuse information gathered in the previous steps (since there is only one step), the feature values in the transformed datasets are obtained from the transitive closures of the neighborhoods. Thus, the SVM and DT models were trained on data *pre-propagated*. A graph whose nodes are the terms of E and whose edges are relations learned by SVM or DT can then be constructed. The output taxonomies are the transitive closures of these graphs.

Table 9 shows the performance of each algorithm for the task of capturing the hypernymy relation of a given sub-domain. The implementations used for SVM and DT fail to learn models on the domain *food* because of the size of the dataset, which contains over 2 millions instances (term pairs). The performance of LPSMI is often better, or at least similar, to that of SVM or DT.

Finally, we show the logical formulas learned by LPSMI in Table 10. We observe that some predicates appear more frequently than others, for instance the predicate q_{Fu} appears in almost all formulas. Moreover, it is often the only component of a conjunctive clause and is therefore not constrained by

any other predicate. This is consistent with the performance of the individual predicates (Table 8) as R_{Fu} is the best relation among the five considered in this study. The predicate $q_{patterns}$ appears alone half of the time and is combined with other predicates the other half. It is unsurprising that $q_{patterns}$ appears alone since pattern-based approaches are known to be very accurate. But when $q_{patterns}$ is combined with another predicate, it acts as a "barrier" preventing the hypernymy relation from spreading too far.

We also observe that the most complex formula is learned on the largest domains, *food* and *animals*. We believe that this phenomenon arises from the existence of multiple natures of hypernymy relations. For instance, one part of the structure of the taxonomy of the domain *food* may be governed by a given hypernymy relation while another part is governed by a different hypernymy relation. Larger formulas can handle this diversity more easily, by specializing a subset of the formula to model a particular nature, or shape, of the relation. Fu et al. [37], whose work inspired the predicate q_{Fu}, take this diversity into account by first clustering word pairs together, and then learning one model per cluster. We therefore believe that it is not a coincidence that q_{Fu} is the best performing predicate of this study. Although it is not sufficient on its own, as indicated by the better results obtained by LPSMI, SVM and DT, which are able to take advantage of other types of data by combining them.

We asked whether a model trained on one given domain can be efficiently reused to build the lexical taxonomy of another domain. This may be difficult because, as suggested earlier, the singularities of one domain may not match those of another domain. But, maybe, a model learned on a subdomain could be a good candidate for extracting the lexical taxonomy of its full domain. For instance, a model learned on the sub-domain *crafts* might capture the nature of the relations of the domain *vehicles* better than it does on the domain *plants*. This is the third hypothesis to be evaluated and the subject of the following subsection.

6.4.2. Applying a learned pretopological space to other domains

The purpose of our second experiment is to determine whether a single model trained on a specific domain can be reused for structuring any other domain. We believe that, given a set E of terms from a domain to be reconstructed, it is more reliable to learn a structuring model based on a set $E' \subseteq E$ of terms from a sub-domain rather than a model based on a completely disjoint set E'' of terms, such that $E'' \cap E = \emptyset$. Thus, for example, we assume that a model trained on the sub-domain *crafts* will perform better than another model trained on *grasses* for the structuring the domain *vehicles*. In order to validate this hypothesis, a structuring model (E', a) was trained by the LPSMI algorithm for each sub-domain. Then, the three domains *vehicles*, *plants* and *food* were structured with respect to the previously learned models. As a result, we obtained nine lexical taxonomies per domains. Table 11 shows the F-measure scores assigned to these taxonomies.

This experiment tends to invalidate our prior hypothesis, since only the domain *animals* seems to be better retrieved by the models trained on its sub-

Domain	Formula
vehicles	$(Fu) \vee (patterns \wedge strmatch)$ $\vee (Sand. \wedge patterns) \vee (Sand. \wedge strmatch)$
wagons	$(Sand.) \vee (meanH)$
crafts	$(Sand. \wedge strmatch) \vee (meanH \wedge strmatch)$ $\vee (Sand. \wedge patterns) \vee (Fu)$
motor vehicles	$(patterns) \vee (strmatch) \vee (Sand.)$
plants	$(Sand. \wedge patterns) \vee (strmatch) \vee (Fu)$
bulbous plants	$(strmatch) \vee (Sand. \wedge Fu)$
aquatic plants	$(patterns)$
grasses	$(strmatch) \vee (Fu)$
food	$(Sand. \wedge Fu) \vee (patterns \wedge Fu) \vee (meanH \wedge Fu)$ $\vee (Fu \wedge strmatch) \vee (patterns \wedge meanH)$ $\vee (meanH \wedge strmatch) \vee (patterns \wedge strmatch)$ $\vee (Sand. \wedge patterns) \vee (Sand. \wedge meanH)$ $\vee (Sand. \wedge strmatch)$
candy	$(Fu) \vee (Sand. \wedge meanH) \vee (strmatch)$
bread	$(strmatch) \vee (Fu)$
snack food	$(Fu) \vee (Sand. \wedge meanH) \vee (strmatch)$
animals	$(Fu \wedge strmatch) \vee (Sand. \wedge Fu) \vee (meanH \wedge Fu)$ $\vee (Sand. \wedge patterns) \vee (Sand. \wedge strmatch)$ $\vee (Sand. \wedge meanH) \vee (patterns \wedge meanH)$
reptiles	$(Fu) \vee (Sand. \wedge meanH) \vee (patterns \wedge meanH)$ $\vee (strmatch)$
fish	$(patterns) \vee (Sand. \wedge meanH) \vee (strmatch) \vee (Fu)$
birds	$(patterns) \vee (Sand. \wedge meanH) \vee (Fu) \vee (strmatch)$

Table 10: Logical formulas learned by LPSMI for each domain.

Train domain	vehicles	(mean)	plants	(mean)	food	(mean)	animals	(mean)
wagons	0.25		0.13		0.05		0.19	
crafts	0.34	**0.34**	0.12	**0.14**	0.15	**0.14**	0.06	**0.16**
motor vehicles	0.43		0.16		0.21		0.22	
bulbous plants	0.12		0.05		0.12		0.05	
aquatic plants	0.25	**0.16**	0.12	**0.08**	0.06	**0.10**	0.19	**0.10**
grasses	0.12		0.06		0.12		0.06	
candy	0.19		0.06		0.13		0.07	
bread	0.12	**0.15**	0.05	**0.05**	0.13	**0.13**	0.05	**0.06**
snack food	0.15		0.05		0.12		0.06	
reptiles	0.32		0.06		0.13		0.11	
fish	0.42	**0.38**	0.15	**0.12**	0.29	**0.24**	0.20	**0.18**
birds	0.41		0.16		0.29		0.24	

Table 11: F-measure scores obtained by pretopological models for full domain reconstruction. The models were trained on sub-domains (rows) and applied on full-domain (columns). The column "(mean)" shows the average score obtained by models learned on the three sub-domains for the reconstruction of a given complete domain.

domains. If we ignore the model trained on the very small sub-domain *wagons*, the models trained on the sub-domain *vehicles* perform as well as those trained on the sub-domain *animals*. On the other hand, the poor result obtained in the structuring of *plants* are in fact due to the sparsity of the relations extracted from the domain *plants*, which prevents the learning of a good propagation model.

The performance of each model varies greatly depending on the domain being reconstructed. This suggests that the concept of hypernym/hyponym might be different depending on the target domain, so that learning a single structuring model to govern them all is, at least, very difficult, even for common domains such as those considered in this study.

We also performed this experiment with models learned by SVM and DT on binary data (Table 12). While our previous experiment showed that LPSMI was able to more reliably retrieve the lexical taxonomy of a given domain than either SVM or DT, Table 12 suggests that LPSMI is no more suitable than SVM and DT for extracting a broader lexical taxonomy of a sub-domain. Indeed, performance of each approach on the generalization task is quite similar (except for DT which performs very poorly when trained on the sub-domain *animals*).

As a higher-level discussion, note that the learning problem considered by the SVM and DT approaches is very different in nature from the LPS learning approach. On one hand, SVM and DT aim to maximize the accuracy of decisions on a set of term pairs without any further constraints on the final structure. On the other hand, LPS approaches, such as LPSMI, learn a propagation model that provides structures that are as accurate as possible and at the same time

Train domain	vehicles	(mean)	plants	(mean)	food	(mean)	animals	(mean)
wagons	0.25		0.13		0.05		0.19	
crafts	0.30	**0.32**	0.05	**0.11**	0.12	**0.15**	0.05	**0.15**
motor vehicles	0.41		0.15		0.27		0.21	
bulbous plants	0.25		0.12		0.06		0.19	
aquatic plants	0.25	**0.21**	0.12	**0.10**	0.06	**0.08**	0.19	**0.15**
grasses	0.12		0.06		0.12		0.06	
candy	0.12		0.05		0.12		0.06	
bread	0.12	**0.12**	0.05	**0.05**	0.13	**0.12**	0.05	**0.05**
snack food	0.12		0.05		0.12		0.05	
reptiles	0.12		0.05		0.12		0.07	
fish	0.38	**0.29**	0.15	**0.12**	0.27	**0.22**	0.20	**0.17**
birds	0.38		0.15		0.27		0.23	

(a) SVM

Train domain	vehicles	(mean)	plants	(mean)	food	(mean)	animals	(mean)
wagons	0.28		0.25		0.21		0.14	
crafts	0.36	**0.26**	0.25	**0.18**	0.21	**0.18**	0.14	**0.11**
motor vehicles	0.15		0.05		0.12		0.05	
bulbous plants	0.25		0.12		0.06		0.19	
aquatic plants	0.25	**0.21**	0.12	**0.10**	0.06	**0.08**	0.19	**0.15**
grasses	0.12		0.06		0.12		0.06	
candy	0.12		0.05		0.12		0.06	
bread	0.12	**0.12**	0.05	**0.05**	0.13	**0.12**	0.05	**0.05**
snack food	0.12		0.05		0.12		0.05	
reptiles	0.00		0.00		0.00		0.02	
fish	0.00	**0.04**	0.00	**0.02**	0.00	**0.04**	0.04	**0.06**
birds	0.12		0.05		0.12		0.11	

(b) Decision Tree

Table 12: F-measure scores obtained by different models for full domain reconstruction. The models were trained on sub-domains (rows) and applied on full-domains (columns). The column "(mean)" shows the average score obtained by the models learned on three sub-domains for the reconstruction of a given full domain.

satisfies the isotonic property that constrains the final structure to be a DAG[9]. Very concretely, the isotonic property prevents a relation $y \rightarrow z$ ($z \in F(\{y\})$) from being made in cases where $x \rightarrow y$ ($y \in F(\{x\})$) exists but $x \rightarrow z$ ($z \notin F(\{x\})$) does not. Such a structuring constraint has an undeniable impact on the quantitative evaluation scores which do not summarize the whole structure obtained but simply consider the final structure as a set of independent pairs.

6.5. Propagation of the semantic relation

The aim of this last section is to detail and illustrate on an example the complex propagation process offered by the pretopological framework in the context of LT. Considering the set E of 35 terms corresponding to the sub-domain *crafts*, and the five neighborhood relations giving rise to five neighborhoods $V_{Sand.}$, $V_{patterns}$, V_{meanH}, V_{Fu} and $V_{strmatch}$ defined on $\mathcal{P}(E)$; LPSMI learns a pretopological space (E, a_Q) defined, for any $x \in E$ and for any $A \in \mathcal{P}(E)$, by the following DNF.

$$Q(A,x) = (q_{Sand.}(A,x) \wedge q_{strmatch}(A,x)) \vee q_{patterns}(A,x) \vee q_{Fu}(A,x)$$

Given a term $x \in E$, its set $F_Q(\{x\})$ of hyponyms is built by successive applications of the pseudo-closure operator $a_Q(.)$. Each application of $a_Q(.)$ leverages the information gathered from previous applications, further propagating the semantic relation. Figure 16 illustrates this process by focusing on the first two steps of expanding the singleton $\{vessels\}$ to its elementary closed set. In this example, although $a_Q(\{vessels\})$ is not able to directly retrieve the expected hyponym "tugboats", the second application of the pseudo-closure operator $a_Q(a_Q(\{vessels\}))$ leverages the information provided by $a_Q(\{vessels\})$ to determine that "tugboats" is ultimately a hyponym of "vessels".

As observed in Figure 16, the second clause ($q_{Sand.} \wedge q_{strmatch}$) of Q triggers the propagation of the subset $a_Q(\{vessels\}) = \{vessels, boats, ships\}$ to the term "tugboats". The precise propagation mechanism, detailed in Figure 17, reveals that both $q_{Sand.}$ and $q_{strmatch}$ are satisfied through different elements:

- $q_{Sand.}(a_Q(\{vessels\}), tugboats)$ is satisfied by the relation

$$ships \quad R_{Sand.} \quad tugboats$$

- $q_{strmatch}(a_Q(\{vessels\}), tugboats)$ is satisfied by the relation

$$boats \quad R_{strmatch} \quad tugboats$$

The structuring of the learned pretopological space into a lexical taxonomy is done according to its elementary closed sets, as shown in [48]. For each term $x \in F_Q(\{vessels\})$, its elementary closed set $F_Q(\{x\})$ is calculated. A lexical

[9] Considering that elements with the same closure constitute a single node in the structure.

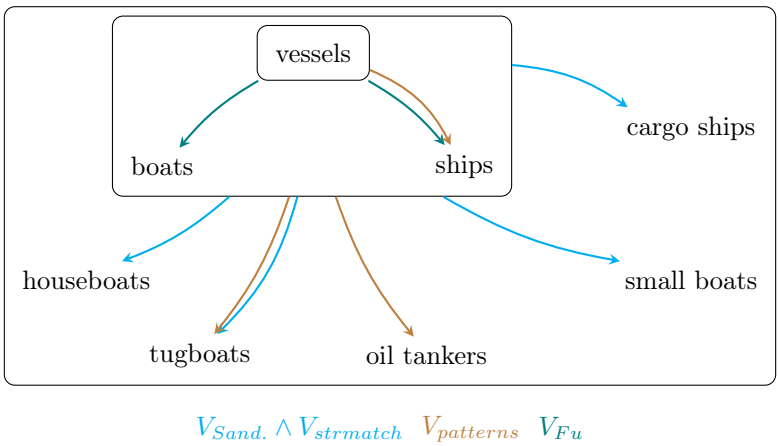


Figure 16: Step by step propagation of a semantic relation. The expansion starts from the singleton $\{vessels\}$.

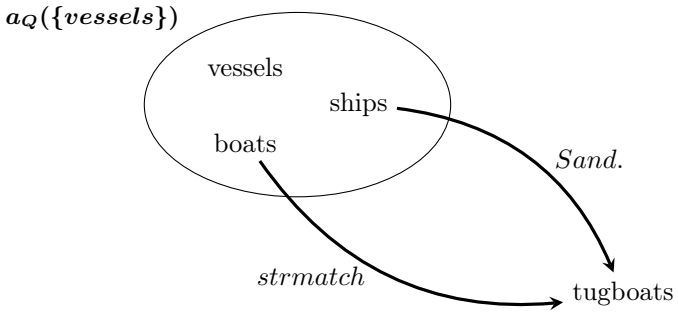


Figure 17: The set $\{vessels, boats, ship\}$ is expanded to the term "tugboats" by the conjunctive clause $q_{Sand.} \wedge q_{strmatch}$ because "ships" is a hypernym of "tugboats" according to $q_{Sand.}$ and "boats" is a hypernym of "tugboats" according to $q_{strmatch}$.

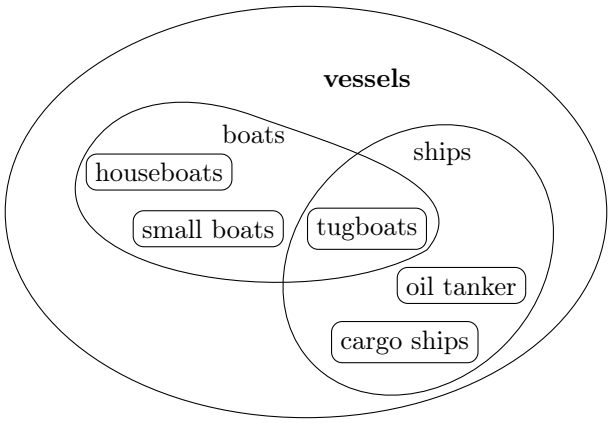


Figure 18: Inclusion relations between a collection of elementary closed sets. A squared node is an elementary closed set of size 1 and represents a leaf of the lexical taxonomy.

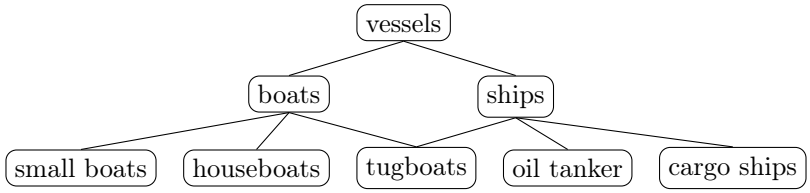


Figure 19: A learned taxonomy with *vessels* as root concept.

taxonomy is deduced from the way the closed sets include each other: Figure 18 shows the inclusion relations between the elements represented in Figure 16. For example, $F_Q(\{houseboats\}) \subset F_Q(\{boats\})$ means that the concept *boats* subsumes the concept *houseboats*, so the concept *boats* will be on top of the concept *houseboats* in the resulting lexical taxonomy. The lexical taxonomy engendered from the closed sets described in Figure 18 is presented in Figure 19.

6.6. Benchmarks

Finally, we compare the performance of LPSMI with competing systems on the Task 13 of the SemEval 2016 campaign [16]. The systems were evaluated on their performance in retrieving taxonomies from the following three domains: *food*, *science* and *environment*. As we only experimented with the domain *food*, we will not consider the last two.

The experiment is the same as the one presented in the previous section: the systems have been trained (or not if they do not need supervision) on the domains *vehicles* and *plants*, which habe been given by the organizers. Given a list of terms, the task is to produce a lexical taxonomy structuring *at least* the terms given as input, new terms can also be added to the taxonomy.

Five systems were submitted but only three produced a taxonomy for *food*. The performance of these systems, as well as the baseline proposed by the

organizers, is presented in Table 13[10]. The gold standard taxonomies used in SemEval exclusively describe direct hypernym-hyponym pairs. Thus, transitive relations are excluded from this study, unlike previously.

System	Recall	Precision	F-Measure
Baseline	0.26	0.50	0.34
JUNLP	0.34	0.15	0.20
TAXI	0.34	0.26	0.29
USAAR	0.24	0.71	0.36

Table 13: Performance of competing systems in Task 13 of SemEval 2016 on the domain *food*.

Train domain	Recall	Precision	F-Measure
vehicles	0.26	0.42	0.32
wagons	0.31	0.00	0.00
crafts	0.26	0.42	0.32
motor vehicles	0.32	0.04	0.07
plants	0.22	0.48	0.30
bulbous plants	0.25	0.35	0.29
aquatic plants	0.09	0.10	0.10
grasses	0.25	0.14	0.18

Table 14: Performances of LPSMI models, trained on different sub-domains, for the reconstruction of the taxonomy *food*.

Baseline. The baseline is based on the same principle as our relation *strmatch*. A hypernym-hyponym pair (x, y) is extracted when the hypernym x is a prefix or a suffix of the hyponym y. For example, "candy" and "corn" are hypernyms of "corn candy" along this baseline.

The performance of the baseline is quite good compared to other systems, even though it is a very naive approach. It works well since the experiment consists in structuring a set of terms of the same semantic domain. For example, half (37 out of 74) of the hyponyms in "sauce" are either prefixed or suffixed with "sauce". In addition, only a few terms (7 out of 47) beginning or ending with "sauce" are not hyponyms of "sauce". Of course, this criticism also applies to our neighborhoods *strmatch*.

JUNLP [53]. JUNLP is composed of two modules. The first queries BabelNet [58] to obtain a list of hyponyms, which is then filtered to reduce noise. BabelNet is a multilingual semantic network built from various sources, including

[10]Precision and recall are available at `alt.qcri.org/semeval2016/task13/index.php?id=evaluation`.

WordNet. Since the evaluation uses gold taxonomies extracted from WordNet, it used only the Wikipedia source of BabelNet.

The second module is based on the same assumption as the Baseline: potential hypernyms are extracted based on overlaps between words. For example, the overlap between "biochemistry" and "chemistry" is "chemistry", then "chemistry" is likely to be a hypernym of "biochemistry". It uses this same assumption to enrich the lexical taxonomy with words that are not in the input term list. For example, while "pudding" was not in the term list, it was added as a hypernym of "chocolate pudding" and "vanilla pudding".

TAXI [62]. TAXI was designed with scalability and simplicity in mind. The assumption behind TAXI is that it is better to process a lot of data with simple tools than to process little data with a sophisticated system. Thus, a lot of effort has been put into the corpora feeding TAXI. They used three general purpose corpora (Wikipedia, 59G and CommonCrawl) and built a domain specific corpus using BootCaT [11] to crawl the web.

Next, they used the same approach as the baseline to extract hypernym-hyponym pairs, as well as lexico-syntactic patterns. They train an SVM model on the trials taxonomies (*vehicles* and *plants*) to determine whether a pair (x, y) is a hypernym-hyponym pair.

USAAR [70]. The USAAR system assumes that many hypernym-hyponym pairs can be detected by their *endocentric* grammatical construction. A grammatical construction is endocentric when one of its components performs the same linguistic function as itself. For example, "aircraft" is an endocentric construction since it is a noun, as well as its component "craft". Furthermore, "craft" is indeed a hypernym of "aircraft".

The performance of LPSMI models is very heterogeneous depending on the dataset used to train them, as shown by Table 14[11]. Training on *plants*, or a sub-domain, seems difficult, mainly because our neighborhoods cannot capture the hypernymy relation of these domains. The model trained on the sub-domain *wagons* also performed poorly, but this is not surprising since *wagons* has only height terms and seven edges. On the other hand, the poor performance of the model trained on *motor vehicles* is more surprising, especially since the model trained on *crafts* performs well. The two domains have two lexical taxonomies with almost identical sizes, but the *crafts* taxonomy is a bit deeper (see Table 7). Therefore, the taxonomy *crafts* has twice as many transitive edges as the taxonomy *motor vehicles*. This could explain the performance gap between these two models.

LPSMI's performances depends heavily on the quality of its training data. First, it needs a *sufficiently large* amount of data: *wagons* has too little data, but *crafts* and *vehicles* seem to be quite large, although the size of the respective taxonomies is rather small (they can be done by hand). More importantly, the

[11] Transitive edges are ignored, which explains the differences with previous results.

Web source	Method	vehicles	plants	animals
wikipedia	OntoLearn DAG_0_99	0.19	0.28	0.27
wikipedia	OntoLearn DAG_1_3	0.17	0.26	0.25
wikipedia	Kozareva&Hovy	0.16	0.14	0.16
wikipedia	LPSMI *supervised*	0.31	0.39	0.22
wikipedia	LPSMI *semi-supervised*	0.35	0.17	0.10
Yahoo!Boss	Kozareva&Hovy	0.49	0.34	0.30

Table 15: Comparison of state-of-the-art methodologies using the cumulative Fowlkes and Mallows measure (F&M).

neighborhoods provided to LPSMI must capture some of the target relations. It seems obvious, but it is difficult to estimate in advance the quality of a given neighborhood, on a given task. Our experiments clearly show this fact: the same neighborhoods work very well on *vehicles* but not on *plants*. Therefore, the model trained on *vehicles* is very efficient, while the one trained on *plants* is not.

In order to complement this comparative study, and thus provide additional insight, we compared the taxonomies induced by LPSMI with those generated by two acquisition methods that are references in the community: the OntoLearn Realoaded approach [72] and the approach proposed by Kozareva and Hovy (K&H) [46]. This quantitative comparison was performed using data provided by the authors[12] on the following WordNet sub-domains: *Vehicles*, *Plants* and *Animals* (688 terms). Since Kozareva and Hovy initially used the Yahoo!Boss search engine, we replicated their algorithm using Wikipedia data in order to compare the methods on common bases and to understand the influence of the source information corpus (Web source) on the quality of the extracted structures.

The evaluation criterion used this time is the Cumulative Fowlkes and Mallows measure (F&M) as defined in [72], used in [15] and for which executable sources are made available[13]; this measure compares an extracted taxonomy and the gold standard by measuring the matching between the partitions (of terms) induced at each level of the structures to be compared.

Table 15 presents the results of this comparative study. The table is divided into two parts, the first five rows contain the results of the different methods (OntoLearn Reloaded, K&H and LPSMI) based on information extracted from Wikipedia (free) while the last row corresponds to the original K&H method based on Yahoo!Boss (not free). By comparing the results obtained by K&H on Wikipedia and on Yahoo!Boss, we can see how crucial the source is in this

[12] http://ontolearn.org/
[13] https://alt.qcri.org/semeval2015/task17/

process; the choice of Yahoo!Boss explains in part the good scores observed by K&H in the literature.

The upper part of the table presents results that can be reasonably compared since the associated methodologies use the same source of information (Wikipedia) but in different ways. The five methodologies compared are : OntoLearn Reloaded with two different parameters for the pruning step (DAG [1,3] and DAG [0,99]), K&H and two uses of the LPSMI algorithm :

- LPSMI for the reconstruction task: the model is learned from the complete taxonomy, then the taxonomy is reconstructed from the learned model (which naturally leads to good results).
- semi-supervised LPSMI: a set of 9 models was learned from 9 sub-domains (3 from each initial domain). Each model is then used to reconstruct a complete taxonomy. The model finally retained for a taxonomy to be reconstructed is the one that induces the most relations (without explosion[14]).

Unlike OntoLearn Reloaded and K&H, LPSMI is not a methodology dedicated to the extraction of taxonomies but a generic algorithm for learning pretopological spaces. Nevertheless, we observe that the LPSMI algorithm obtains quite competitive results on this task, outperforming even dedicated methods on the dataset *vehicles*.

7. Conclusion

We introduce a new method to learn (in a supervised context) a pretopological space based on its elementary closed sets: LPS multiple instances (LPSMI). This method is the continuation of the work done by Cleuziou and Dias [28] where a pseudo-closure was modeled by a numerical vector and learned by a genetic algorithm. Due to the lack of expressiveness of the numerical formalism, we switched to a more appropriate logical modeling: a positive DNF defining a pseudo-closure operator. This new definition of a pretopological space allows to consider many more solutions while restricting the hypotheses space, thus making its exploration both easier and faster. LPSMI is also faster than its competitors in delivering results because its learning strategy explores the search space more efficiently thanks to an appropriate (intrinsic) quality measure.

Learning a pretopological space is a wonderful improvement in how one can take advantage of pretopology theory. In previous work, the pseudo-closure operator was hand-crafted and based on a collection of neighborhoods *supposed* to accurately model a given phenomenon. Learning a pseudo-closure operator

[14] We observe that some models do not sufficiently regulate the propagation of relations, thus generating many cycles and making their evaluation impossible in practice.

allows one to construct a better combination among any collection of neighborhoods, such that only useful neighborhoods are considered in the learned pseudo-closure operator.

The learning of pretopological spaces remains an open problem, mainly due to its difficulties in scaling to large amount of data. Given an already trained pretopological space, computing its set of elementary closed sets can be costly depending on the propagation speed defined by the pretopological operator. Although, since we are currently only considering V-type pretopological spaces, the computation of the elementary closed sets can easily be speed up by merging *compatible pseudo-closures*. Let us say that we have computed a first round of pseudo-closures such that $a(\{x\}) = \{x, y\}$ and $a(\{y\}) = \{y, z\}$, then we already know that $a^2(\{x\})$ will contain z. Then, we can skip a pseudo-closure step by computing $a(a(\{x\}) \cup a(\{y\}))$ instead of $a(a(\{x\}))$. However, the major concern comes from the learning algorithm, because we observed that the propagation speed to the elementary closed sets is reasonable. Indeed, the solution space is very large, especially given the small size of the inputs. A possible improvement to the current strategy of exploring the solution space could be to compute only partial closed sets and move as soon as the candidate solution gets too far from the target. As the time of writing, there is no satisfying solution for efficiently learning a pretopological space. Laborde [47] propose an alternative learning algorithm, but it remains rather restrictive since it assumes that an exact solution exists, which is far from realistic.

By applying the LPSMI algorithm to the task of learning a lexical taxonomy, we have shown that the pretopology theory is well suited to model the semantic relations between terms in a given domain. We have also shown that a model learned with LPSMI will almost always be preferable to a solution based on a single data source.

Work needs to be done to remove the dependency on the binary neighborhood. A naive solution to this problem would be to build many binary neighborhoods from many thresholds, but this would be very expensive in terms of space and time complexity. We plan to study solutions based on fuzzy pretopology [10] or any broader theory [45]. Future works will also focus on other modeling of the pseudo-closure operator. For example, a pseudo-closure operator based on first logic order formulas could lead to even better expressiveness. We plan to learn, at the same time, a combination of continuous neighborhoods and a set of parameters modifying these neighborhoods (typically, a threshold) in the hope of reducing the false positive rate of the learned solution. We also plan to refine our estimation of the positive/negative bags covered by a solution, as a better approximation could lead to more accurate solutions.

LPSMI needs to be tested on a broader range of application cases such as the modeling of expansion behaviors in networks (social networks, biological networks, road networks etc...). We have already shown the relevance of pretopology to model complex propagation in networks [22]. Since this is the problem that motivated the learning of a pretopological space, we will continue to use our method to learn models useful for text mining, such as the task of building lexical taxonomies. We also plan to use it to learn temporal ordering models

of events, in texts, based on annotated corpora such as TimeBank-Dense [23].

References

[1] Ahat, M., Amor, S.B., Bui, M., Jhean-Larose, S., Denhière, G., 2010. Document classification with LSA and pretopology. Stud. Inform. Univ. 8, 125–144.

[2] Ahat, M., Amor, S.B., Bui, M., Lamure, M., Courel, M., 2009a. Pollution modeling and simulation with multi-agent and pretopology, in: Complex (1), Springer. pp. 225–231.

[3] Ahat, M., Amor, S.B., Bui, M., Lamure, M., Courel, M.F., 2009b. Pollution modeling and simulation with multi-agent and pretopology. Complex Sciences , 225–231.

[4] Aho, A.V., Garey, M.R., Ullman, J.D., 1972. The transitive reduction of a directed graph. SIAM Journal on Computing 1, 131–137.

[5] Amin, S.M., Wollenberg, B.F., 2005. Toward a smart grid: power delivery for the 21st century. IEEE power and energy magazine 3, 34–41.

[6] Amor, S.B., Lavallée, I., Bui, M., 2006. Percolation, pretopology and complex systems modeling. Complex Systems Modeling and Cognition Eurocontrol and EPHE Joint Research Lab 41.

[7] Amor, S.B., Levorato, V., Lavallée, I., 2007. Generalized percolation processes using pretopology theory, in: RIVF, IEEE. pp. 130–134.

[8] Athanasiadis, I.N., Mitkas, P.A., 2004. An agent-based intelligent environmental monitoring system. Management of Environmental Quality: An International Journal 15, 238–249.

[9] Auray, J., Duru, G., 1982. Fuzzy pretopological structures and formation of coalitions. IFAC Proceedings Volumes 15, 459–463.

[10] Badard, R., 1981. Fuzzy pretopological spaces and their representation. Journal of mathematical analysis and applications 81, 378–390.

[11] Baroni, M., Bernardini, S., 2004. Bootcat: Bootstrapping corpora and terms from the web., in: LREC, Citeseer. p. 1313.

[12] Belmandt, Z., 1993. Manuel de prétopologie et ses applications .

[13] Blockeel, H., Page, D., Srinivasan, A., 2005. Multi-instance tree learning, in: ICML, ACM. pp. 57–64.

[14] Bonnevay, S., 2009. Pretopological operators for gray-level image analysis. Stud. Inform. Univ. 7, 173–195.

[15] Bordea, G., Buitelaar, P., Faralli, S., Navigli, R., 2015. Semeval-2015 task 17: Taxonomy extraction evaluation (texeval), in: Proceedings of the 9th International Workshop on Semantic Evaluation, Association for Computational Linguistics.

[16] Bordea, G., Lefever, E., Buitelaar, P., 2016. Semeval-2016 task 13: Taxonomy extraction evaluation (texeval-2), in: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), pp. 1081–1091.

[17] Bordes, A., Glorot, X., Weston, J., Bengio, Y., 2014. A semantic matching energy function for learning with multi-relational data. Machine Learning 94, 233–259.

[18] Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O., 2013. Translating embeddings for modeling multi-relational data, in: Burges, C.J.C., Bottou, L., Ghahramani, Z., Weinberger, K.Q. (Eds.), Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States., pp. 2787–2795.

[19] Brissaud, M., Lamure, M., Milan, J.J., Auray, J.P., Nicoloyannis, N., Duru, G., Terrenoire, M., Tounissoux, D., Zighed, D.A., Bonnevay, S., et al., 2011. Basics of pretopology.

[20] Bui, M., Amor, S.B., Lamure, M., Basileu, C., 2014. Gesture trajectories modeling using quasipseudometrics and pre-topology for its evaluation, in: International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Springer. pp. 116–134.

[21] Bui, Q.V., Sayadi, K., Bui, M., 2015. A multi-criteria document clustering method based on topic modeling and pseudoclosure function, in: Proceedings of the Sixth International Symposium on Information and Communication Technology, ACM. pp. 38–45.

[22] Caillaut, G., Cleuziou, G., Dugué, N., 2019. Learning pretopological spaces to extract ego-centered communities, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, p. to appear.

[23] Cassidy, T., McDowell, B., Chambers, N., Bethard, S., 2014. An annotation framework for dense event ordering. Technical Report. CARNEGIE-MELLON UNIV PITTSBURGH PA.

[24] Chevaleyre, Y., Zucker, J., 2001. Solving multiple-instance and multiple-part learning problems with decision trees and rule sets. application to the mutagenesis problem, in: Canadian Conference on AI, Springer. pp. 204–214.

[25] Cimiano, P., Hotho, A., Staab, S., 2005a. Learning concept hierarchies from text corpora using formal concept analysis. J. Artif. Intell. Res.(JAIR) 24, 305–339.

[26] Cimiano, P., Pivk, A., Schmidt-Thieme, L., Staab, S., 2005b. Learning taxonomic relations from heterogeneous sources of evidence. Ontology Learning from Text: Methods, evaluation and applications 123, 59–73.

[27] Cleuziou, G., Buscaldi, D., Levorato, V., Dias, G., 2011. A pretopological framework for the automatic construction of lexical-semantic structures from texts, in: Proceedings of the 20th ACM international conference on Information and knowledge management, ACM. pp. 2453–2456.

[28] Cleuziou, G., Dias, G., 2015. Learning pretopological spaces for lexical taxonomy acquisition, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer. pp. 493–508.

[29] Dalud-Vincent, M., Brissaud, M., Lamure, M., 2009. Closed sets and closures in pretopology. International Journal of Pure and Applied Mathematics 50, 391–402.

[30] Dietterich, T.G., Lathrop, R.H., Lozano-Pérez, T., 1997. Solving the multiple instance problem with axis-parallel rectangles. Artif. Intell. 89, 31–71.

[31] Espinosa-Anke, L., Saggion, H., Ronzano, F., Navigli, R., 2016. Extasem! extending, taxonomizing and semantifying domain terminologies, in: Proceedings of the 30th conference on artificial intelligence (AAAI'16).

[32] Faralli, S., Panchenko, A., Biemann, C., Ponzetto, S.P., 2017. The contrastmedium algorithm: Taxonomy induction from noisy knowledge graphs with just a few links, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, pp. 590–600.

[33] Farmer, J.D., 2012. Economics needs to treat the economy as a complex system, in: Paper for the INET Conference 'Rethinking Economics and Politics.

[34] Flati, T., Vannella, D., Pasini, T., Navigli, R., 2016. Multiwibi: The multilingual wikipedia bitaxonomy project. Artificial Intelligence 241, 66–102.

[35] Foster, J., 2005. From simplistic to complex systems in economics. Cambridge Journal of Economics 29, 873–892.

[36] Frélicot, C., Lebourgeois, F., 1998. A pretopology-based supervised pattern classifier, in: ICPR, IEEE Computer Society. pp. 106–109.

[37] Fu, R., Guo, J., Qin, B., Che, W., Wang, H., Liu, T., 2014. Learning semantic hierarchies via word embeddings, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1199–1209.

[38] Galindo, J.F., Rubiano, G., Daza, E.E., 2014. Pretopological spaces as a classification tool for rnas represented as a succession. MATCH Commun. Math. Comput. Chem 72, 453–474.

[39] Gupta, S., MacLean, D.L., Heer, J., Manning, C.D., 2014. Induced lexico-syntactic patterns improve information extraction from online medical forums. Journal of the American Medical Informatics Association 21, 902–909.

[40] Harris, Z.S., 1954. Distributional structure. Word 10, 146–162.

[41] Hart, M.G., Ypma, R.J., Romero-Garcia, R., Price, S.J., Suckling, J., 2016. Graph theory analysis of complex brain networks: new concepts in brain mapping applied to neurosurgery. Journal of neurosurgery 124, 1665–1678.

[42] Hart, P.E., Nilsson, N.J., Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics 4, 100–107.

[43] Hearst, M.A., 1992. Automatic acquisition of hyponyms from large text corpora, in: 14th International Conference on Computational Linguistics, COLING 1992, Nantes, France, August 23-28, 1992, pp. 539–545. URL: https://www.aclweb.org/anthology/C92-2082/.

[44] Iacobacci, I., Pilehvar, M.T., Navigli, R., 2015. Sensembed: Learning sense embeddings for word and relational similarity, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 95–105.

[45] Khedr, F., Abd-Allah, M.A., Abdelgaber, E., 2019. Fuzzy soft pretopological spaces. Global Journal of Mathematics Vol 13.

[46] Kozareva, Z., Hovy, E., 2010. A semi-supervised method to learn and construct taxonomies using the web, in: Proceedings of the 2010 conference on empirical methods in natural language processing, Association for Computational Linguistics. pp. 1110–1118.

[47] Laborde, J., 2019. Pretopology, a mathematical tool for structuring complex systems: methods, algorithms and applications. Ph.D. thesis. PSL Research University.

[48] Largeron, C., Bonnevay, S., 2002. A pretopological approach for structural analysis. Inf. Sci. 144, 169–185.

[49] Levin, S.A., 1998. Ecosystems and the biosphere as complex adaptive systems. Ecosystems 1, 431–436.

[50] Levorato, V., 2014. Group measures and modeling for social networks. Journal of Complex Systems 2014.

[51] Levy, O., Goldberg, Y., 2014. Linguistic regularities in sparse and explicit word representations, in: Proceedings of the eighteenth conference on computational natural language learning, pp. 171–180.

[52] Liu, Z., 2015. Complex systems and health systems, computational challenges. Ph.D. thesis. Versailles-St Quentin en Yvelines.

[53] Maitra, P., Das, D., 2016. Junlp at semeval-2016 task 13: A language independent approach for hypernym identification, in: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), pp. 1310–1314.

[54] Meziane, A., Iftene, T., Selmaoui, N., 1997. Satellite image segmentation by mathematical pretopology and automatic classification, in: Aerospace Remote Sensing'97, International Society for Optics and Photonics. pp. 232–236.

[55] Mikolov, T., Chen, K., Corrado, G., Dean, J., 2013a. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 .

[56] Mikolov, T., Yih, W.t., Zweig, G., 2013b. Linguistic regularities in continuous space word representations, in: Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies, pp. 746–751.

[57] Miller, G.A., 1998. WordNet: An electronic lexical database. MIT press.

[58] Navigli, R., Ponzetto, S.P., 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. Artificial Intelligence 193, 217–250.

[59] Navigli, R., Velardi, P., 2010. Learning word-class lattices for definition and hypernym extraction, in: Hajic, J., Carberry, S., Clark, S. (Eds.), ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden, The Association for Computer Linguistics. pp. 1318–1327. URL: `https://www.aclweb.org/anthology/P10-1134/`.

[60] Nickel, M., Tresp, V., Kriegel, H.P., 2011. A three-way model for collective learning on multi-relational data., in: Icml, pp. 809–816.

[61] Norberg, J., 2004. Biodiversity and ecosystem functioning: a complex adaptive systems approach. Limnology and Oceanography 49, 1269–1277.

[62] Panchenko, A., Faralli, S., Ruppert, E., Remus, S., Naets, H., Fairon, C., Ponzetto, S.P., Biemann, C., 2016. Taxi at semeval-2016 task 13: a taxonomy induction method based on lexico-syntactic patterns, substrings and focused crawling, in: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), pp. 1320–1327.

[63] Pastor-Satorras, R., Castellano, C., Van Mieghem, P., Vespignani, A., 2015. Epidemic processes in complex networks. Reviews of modern physics 87, 925.

[64] Pennington, J., Socher, R., Manning, C.D., 2014. Glove: Global vectors for word representation, in: Moschitti, A., Pang, B., Daelemans, W. (Eds.), Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL, ACL. pp. 1532–1543. URL: https://doi.org/10.3115/v1/d14-1162, doi:10.3115/v1/d14-1162.

[65] Pocostales, J., 2016. Nuig-unlp at semeval-2016 task 13: A simple word embedding-based approach for taxonomy extraction, in: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), pp. 1298–1302.

[66] Resnik, P., et al., 1999. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. J. Artif. Intell. Res.(JAIR) 11, 95–130.

[67] Sanderson, M., Croft, B., 1999. Deriving concept hierarchies from text, in: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, ACM. pp. 206–213.

[68] Snow, R., Jurafsky, D., Ng, A.Y., 2005. Learning syntactic patterns for automatic hypernym discovery, in: Advances in neural information processing systems, pp. 1297–1304.

[69] Taghizadeh, N., Faili, H., 2016. Automatic wordnet development for low-resource languages using cross-lingual wsd. J. Artif. Intell. Res.(JAIR) 56, 61–87.

[70] Tan, L., Bond, F., van Genabith, J., 2016. Usaar at semeval-2016 task 13: Hyponym endocentricity, in: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), pp. 1303–1309.

[71] Van Le, T., Truong, T.N., Nguyen, H.N., Pham, T.V., 2013. An efficient pretopological approach for document clustering, in: Intelligent Networking and Collaborative Systems (INCoS), 2013 5th International Conference on, IEEE. pp. 114–120.

[72] Velardi, P., Faralli, S., Navigli, R., 2013. Ontolearn reloaded: A graph-based algorithm for taxonomy induction. Comput. Linguistics 39, 665–707. URL: https://doi.org/10.1162/COLI_a_00146, doi:10.1162/COLI_a_00146.

[73] Wang, Q., Mao, Z., Wang, B., Guo, L., 2017. Knowledge graph embedding: A survey of approaches and applications. IEEE Transactions on Knowledge and Data Engineering 29, 2724–2743.

[74] Wu, C., Yue, Y., Li, M., Adjei, O., 2004. The rough set theory and applications. Engineering Computations 21, 488–511.

[75] Zadeh, L.A., 1973. Outline of a new approach to the analysis of complex systems and decision processes. IEEE Transactions on Systems, Man, and Cybernetics SMC-3, 28–44.

[76] Zhou, Z.H., 2004. Multi-instance learning: A survey. Department of Computer Science & Technology, Nanjing University, Tech. Rep .

A. Number of positive bags covered by a solution

The number of positive bags covered by a solution Q (the number of true positives) is estimated by subtracting the estimated number of rejected (or not yet covered) positive bags from the total number of positive bags in the MI dataset. Due to the complexity of the pseudo-closure operator, it is not possible to accurately count the number of rejected positive bags in a reasonable time span. We propose a method to estimate this number.

We consider a finite set E, a set S^* of target elementary closed sets and a solution Q under construction. For any element $x \in E$, we consider the true/right subset of its learned elementary closed set, denoted $F^*_Q(\{x\}) = F_Q(\{x\}) \cap F^*(\{x\})$. The definition of the pretopological space (E, a_Q) of type V guarantees, for any element $x \in E$ and any set $A \in \mathcal{P}(E)$, that if $x \in A$ then $F^*_Q(\{x\}) \subseteq F_Q(A)$, because $F^*_Q(\{x\}) \subseteq F_Q(\{x\})$ and $F_Q(\{x\}) \subseteq F_Q(A)$ according to the isotonic property (Definition 3). This means that any set containing x will be correctly expanded, by the pseudo-closure operator$a_Q(.)$, to at least all elements of $F^*_Q(\{x\})$.

$$\forall A \in \mathcal{P}(E),\, x \in A \Rightarrow F^*_Q(\{x\}) \subseteq F_Q(A)$$

In what follows, a (strong) assumption is made that all positive bags (x, A) such that $F^*_Q(\{x\}) \subseteq A \subset F^*(\{x\})$, i.e. the bags described by the sub-lattice $\mathcal{L}\left[F^*_Q(\{x\}), F^*(\{x\})\right[$, are not covered by Q. This is the *elementary coverage hypothesis*.

Definition 4 (The elementary coverage assumption)**.** For any $x \in E$, the set of positive bags whose coverage (by Q) is *not* ensured by the elementary closed set learned from x ($F_Q(\{x\})$) is defined as follows:

$$ec(x) = \mathcal{L}\left[F^*_Q(\{x\}), F^*(\{x\})\right[$$

$ec(x)$ is the set of positive bags (1) engendered by x and (2) rejected by Q if we consider the only knowledge provided by $F^*_Q(\{x\})$. This is a high estimate because the positive bag (x, A) can be covered by Q while not being considered as such by $ec(x)$. This case occurs when A is strictly between $F^*_Q(\{x\})$ and $F^*(\{x\})$, i.e. $F^*_Q(\{x\}) \subset A \subset F^*(\{x\})$, and A propagates to an element of $F^*(\{x\})$, i.e. $a_Q(A) \cap F^*(\{x\}) \setminus A \neq \emptyset$. In such a case, the positive bag (x, A) is theoretically covered but $ec(x)$ marks it as rejected. This problem can be solved by computing the closure of each super-set of $F^*_Q(\{x\})$, which is terribly inefficient in practice. Although this is not a dramatic problem because it only affects non-elementary closed sets, which are not reflected in the final DAG structure.

A low estimation of the positive bags whose coverage (by Q) is provided by the elementary closed set of an element $x \in E$ can be computed by subtracting the size of $ec(x)$ to the total number of positive bags engendered by x.

Figure 20 illustrates how the number of covered positive bags is estimated. Let $E = \{x_1, x_2, x_3, x_4\}$, $F^*(\{x_1\}) = \{x_1, x_2, x_3, x_4\}$ and a DNF Q such that $F_Q(\{x_1\}) = \{x_1, x_2\}$. The whole set of positive bags engendered by the element x_1 is described by the sub-lattice $\mathcal{L}\left[\{x_1\}, F^*(\{x_1\})\right[$. The smallest set that is not properly expanded by $a_Q(.)$ is $F^*_Q(\{x_1\}) = \{x_1, x_2\}$. Thus, we assume that all sets above $\{x_1, x_2\}$ (i.e. $ec(x_1)$) are not properly expanded as well. This seems wrong since $F_Q(\{x_1, x_2, x_4\}) = F^*(\{x_1\})$, so the positive bag $(x_1, \{x_1, x_2, x_4\})$ is covered by Q but ignored by our estimation. The number of positive bags covered by $F_Q(\{x_1\})$ (the green area) is then calculated by subtracting the size of $ec(x_1)$ from the total number of positive bags engendered by x_1. So, there is $|\mathcal{L}\left[\{x_1\}, F^*(\{x_1\})\right[| - |ec(x_1)| = 4$ positive bags engendered by x_1 and covered by Q.

For all $x \in E$, we can estimate (by a lower bound) the number of positive bags engendered by x and covered by Q, by subtracting the number of bags *not yet* covered from the number of bags engendered. But, as with the calculation of the total number of positive bags, care must be taken when several elements share the same elementary closed set. We again use the inclusion-exclusion principle to compute a better estimation of the positive bags not covered by Q. We define the function $r^+_Q : \cup_{A_k \in \mathcal{A}} \mathcal{P}(A_k) \to \mathbb{N}$ which, given a subset B of an equivalence class A_k, produces an estimate of the number of positive bags engendered by the elements in B whose coverage by Q is not always guaranteed by all the elementary closed sets of the elements in B.

On a general purpose, we first define how to compute the size of the union of n sub-lattices $\mathcal{L}_1, \ldots, \mathcal{L}_n$ having the same upper element, noted $\top$, and different lower elements $\bot_1, \ldots, \bot_n$.

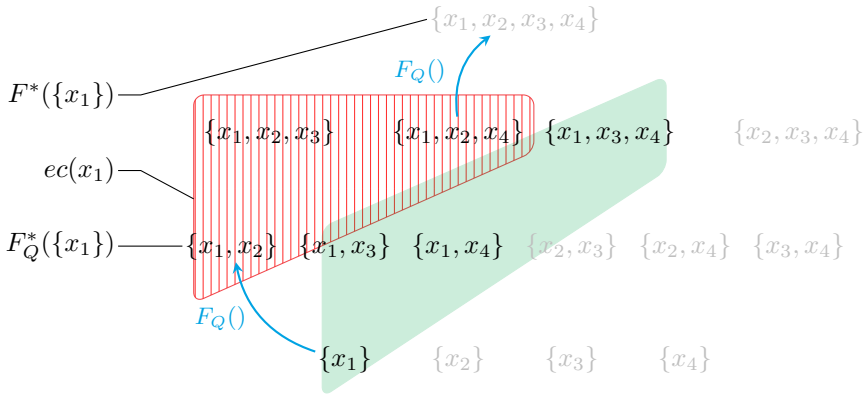


Figure 20: Estimation of the number of positive bags engendered by x_1 that are covered by a solution Q. We suppose $F^*(\{x_1\}) = \{x_1, x_2, x_3, x_4\}$ and $F_Q(\{x_1\}) = \{x_1, x_2\}$. Positive bags whose coverage is ensured by $F_Q(\{x_1\})$ are highlighted by the green area, which is $\mathcal{L}\left[\{x_1\}, F^*(\{x_1\})\right[\setminus ec(x_1)$.

$$\left|\bigcup_{i=1}^{n} \mathcal{L}_i\right| = \sum_{i=1}^{n} (-1)^{i+1} \sum_{B \in \text{comb}(\{1 \ldots n\}, i)} \left|\bigcap_{j \in B} \mathcal{L}_j\right|$$
$$= \sum_{i=1}^{n} (-1)^{i+1} \sum_{B \in \text{comb}(\{1 \ldots n\}, i)} 2^{|\top| - |\cup_{j \in B} \perp_j|}$$

$\text{comb}(\{1 \ldots n\}, i)$ expresses the power-set of $\{1 \ldots n\}$ reduced to its elements of size i. Thus, the size of the union of sub-lattices is calculated by alternatively adding and subtracting the sizes of the intersections between the sub-lattices. The intersection between several sub-lattices sharing the same top element $\top$ is the sub-lattice whose biggest element is $\top$ (obviously) and whose bottom element is the union of the bottom elements of the considered sub-lattices. Then the size of this intersection is expressed by $2^{|\top| - |\cup_{j \in B} \perp_j|}$.

Given any i-permutation B of the equivalence class A_k, we use the elementary coverage assumption to estimate the number of positive bags whose coverage, by Q, is not always ensured by all the elementary closed sets of the elements in B. That is, the number of positive bags which belong to, at least, one element of $\{ec(x)\}_{x \in B}$.

$$\forall B \in \text{comb}(A_k, i),\ r_Q^+(B) = \left|\bigcup_{x \in B} \mathcal{L}\left[F_Q^*(\{x\}) \cup B, F_k^*\right]\right| - 1$$

We have to subtract 1 in order to ignore the top element F_k^*, since a positive bag cannot be identified by a closed set.

Then, for any equivalence class $A_k \in \mathcal{A}$, the estimated number of positive bags engendered by (the elements of) A_k and covered by Q is calculated as follows:

$$\begin{aligned}\forall A_k \in \mathcal{A},\ \text{bags}^+_{\text{Q}}(A_k) &= \sum_{i=1}^{|A_k|} (-1)^{i+1} \sum_{B \in \text{comb}(A_k,i)} |\mathcal{L}\,[B, F_k^*[| - r_Q^+(B) \\ &= \sum_{i=1}^{|A_k|} (-1)^{i+1} \sum_{B \in \text{comb}(A_k,i)} 2^{|F_k^*|-i} - 1 - r_Q^+(B)\end{aligned}$$

The estimated total number of positive bags covered by Q is finally deduced, thanks to Property 3, by summing $\text{bags}^+_{\text{Q}}(A_k)$ for all $A_k \in \mathcal{A}$.

$$BAGS_Q^+ = \sum_{A_k \in \mathcal{A}} \text{bags}^+_{\text{Q}}(A_k)$$

Example. Let's consider a set $E = \{x_1, x_2, x_3, x_4\}$ such that $F^*(\{x_1\}) = F^*(\{x_3\}) = \{x_1, x_2, x_3, x_4\} = F_1^*$. Both x_1 and x_2 belong to the equivalence class $A_1 = \{x_1, x_3\}$. Suppose we learn a DNF Q such that $F_Q(\{x_1\}) = \{x_1, x_2\}$ and $F_Q(\{x_3\}) = \{x_3, x_4\}$. Figure 21 shows three lattices:

- ($\mathcal{L}_{x_1}$) the set of positive bags engendered by x_1 whose coverage is ensured by $F_Q(\{x_1\})$
- ($\mathcal{L}_{x_3}$) the set of positive bags engendered by x_3 whose coverage is ensured by $F_Q(\{x_3\})$
- ($\mathcal{L}_{x_1x_3}$) the set of positive bags engendered by both x_1 and x_3 whose coverage is ensured by both $F_Q(\{x_1\})$ and $F_Q(\{x_3\})$.

The inclusion-exclusion principle is, again, used to estimate the number of positive bags whose coverage is ensured by the elementary closed sets of elements of the equivalence class A_1. First, for all $x \in A_1$ the number of positive bags engendered by x whose coverage is ensured by $F_Q(\{x\})$, i.e. $\left|\text{bags}^+(x)\right| - |ec(x)|$, are summed. Therefore, the positive bag $(A_1, \{x_1, x_3\})$ is counted twice. This error is removed by subtracting the number of positive bags whose coverage is ensured by both $F_Q(\{x_1\})$ and $F_Q(\{x_3\})$, i.e. $|\mathcal{L}\,[\{x_1, x_3\}, F_1^*[| - r_Q^+(\{x_1, x_3\})$. The calculation is expressed below.

$$\begin{aligned}\text{bags}^+_{\text{Q}}(A_1) &= \Big(|\mathcal{L}\,[\{x_1\}, F_1^*[| - r_Q^+(\{x_1\})\Big) + \Big(|\mathcal{L}\,[\{x_3\}, F_1^*[| - r_Q^+(\{x_3\})\Big) \\ &\quad - \Big(|\mathcal{L}\,[\{x_1, x_3\}, F_1^*[| - r_Q^+(\{x_1, x_3\})\Big) \\ &= (7-3) + (7-3) - (3-2) \\ &= 7\end{aligned}$$

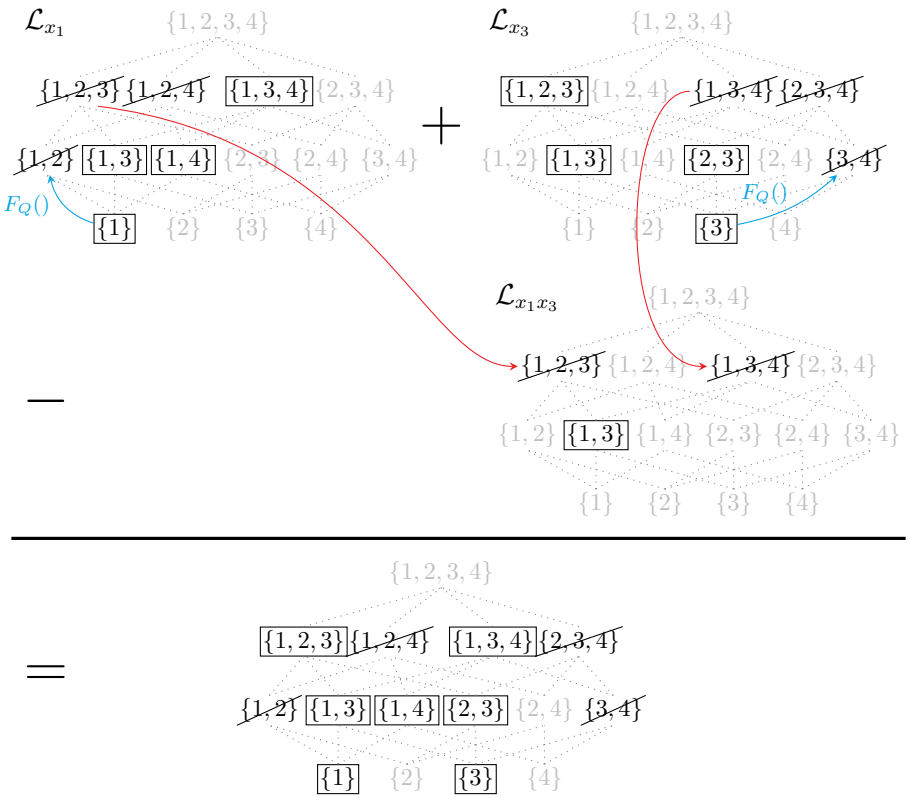


Figure 21: Calculation of the estimate of the number of positive bags engendered by $A_1 = \{x_1, x_3\}$ and covered by Q, with $E = \{x_1, x_2, x_3, x_4\}$, $F_1^* = E$, $F_Q(\{x_1\}) = \{x_1, x_2\}$ and $F_Q(\{x_3\}) = \{x_3, x_4\}$. The first line describes the first step of the inclusion-exclusion principle and the second line describes the second step. A strike-through sets represent the rejected positive bags. The "x" parts are deleted for readability, so the set $\{1, 2\}$ corresponds to the set $\{x_1, x_2\}$.