An Approach to Predicate Invention Based on Statistical Relational Model

Stefano Ferilli[1,2(✉)] and Giuseppe Fatiguso[1]

[1] Dipartimento di Informatica, Università di Bari, Bari, Italy
stefano.ferilli@uniba.it
[2] Centro Interdipartimentale per la Logica e sue Applicazioni, Università di Bari, Bari, Italy

Abstract. Predicate Invention is the branch of symbolic Machine Learning aimed at discovering new emerging concepts in the available knowledge. The outcome of this task may have important consequences on the efficiency and effectiveness of many kinds of exploitation of the available knowledge. Two fundamental problems in Predicate Invention are how to handle the combinatorial explosion of candidate concepts to be invented, and how to select only those that are really relevant. Due to the huge number of candidates, there is a need for automatic techniques to assign a degree of relevance to the various candidates and select the best ones. Purely logical approaches may be too rigid for this purpose, while statistical solutions may provide the required flexibility.

This paper proposes a new Statistical Relational Learning approach to Predicate Invention. The candidate predicates are identified in a logic theory, rather than in the background knowledge, and are used to restructure the theory itself. Specifically, the proposed approach exploits the Markov Logic Networks framework to assess the relevance of candidate predicate definitions. It was implemented and tested on a traditional problem in Inductive Logic Programming, yielding interesting results.

1 Introduction

In the context of symbolic Machine Learning, Predicate Invention (PI) deals with the problem of discovering new emerging concepts in the available knowledge. Invention of suitable predicates is so relevant not just because it allows to identify hidden concepts in the available knowledge, but also because it may have important consequences on the efficiency and effectiveness of many kinds of tasks involved in the exploitation of such a knowledge. E.g., it allows theory restructuring in order to have more compact and comprehensible theories; it may help in using and linking multi-domain knowledge; it may suggest how to shift the language towards higher-level representations; it may enrich the available knowledge and improve the comprehension of a domain by human experts; etc. Associated to inductive learning, in particular, PI would allow to obtain a compression of the learned theory or to catch exceptions in the learned clauses.

M. Gavanelli et al. (Eds.): AI*IA 2015, LNAI 9336, pp. 274–287, 2015.
DOI: 10.1007/978-3-319-24309-2_21

Despite the problem has been investigated for many years, limited results have been obtained due to its intrinsic complexity.

Two fundamental problems in PI are how to handle the combinatorial explosion of candidate concepts to be invented, and how to select only those that are really relevant. Indeed, a huge number of possible concepts might be defined with the given knowledge, most of which are just casual or otherwise irrelevant aggregations of features. So, proper filtering must be carried out in order to keep the most significant and relevant concepts only. Due to the large number of candidates, manual selection by domain experts is infeasible, and automatic techniques are needed to assign a degree of relevance to the various candidates and select the best ones.

PI can be carried out on two different kinds of sources: on a background knowledge made up of facts, or on a general theory made up of rules. In a Machine Learning setting, both are available: the theory is the model obtained as a result of running some learning system on the facts that describe the observations and the examples. In some sense, such a model has already performed a kind of selection of relevant information in the background knowledge (in this case, information that may characterize a set of classes and/or discriminate them from other classes). So, working on the theory should make the predicate invention problem somehow easier while still accounting also for the background knowledge.

Inductive Logic Programming (ILP) is the branch of Machine Learning interested in dealing with logic-based (especially first-order) representation formalisms. However, using a purely logical approach, ILP techniques are sometimes too rigid to deal with noisy data, where they may be affected by overfitting. Statistical approaches may provide the flexibility that is required to overcome these problems. Statistical Relational Learning (SRL) is the branch of ILP aimed at extending its classical, purely logical approach to handle probabilities. Using SRL to approach the PI problem resulted in the Statistical Predicate Invention (SPI) area of research, interested in the discovery of new concepts from relational data by means of statistical approaches.

This paper proposes *Weighted Predicate Invention* (WPI), a SPI approach that focuses on the discovery of tacit relations inside the theories learned by a traditional ILP system.

Albeit working in the SPI setting, it aims at preserving the ILP perspective on the PI problem, but combining it with statistical learning as a guidance to avoid the problem of noisy concepts. Thus, the proposed approach should fit both the specific perspectives of ILP and SRL. As regards the former, it invents new concepts based on the analysis of the structure of a first-order theory in order to find commonalities between the formulas in the theory and then inventing predicates by inter-construction. As to the latter, the role SRL plays in WPI is crucial but quite different from previous SPI proposals. Markov Logic Networks (MLNs) are used to estimate whether a candidate concept to be invented is actually useful with respect to the logic theory in which it was found, by comparing the weight of first-order formulas in MLNs with and without the subsumption of

an invented predicate we can estimate if an that predicate should be introduced in the initial theory or not. After a predicate is invented it can be incorporated into the theory, and the process can be carried out again, until no new predicate can be invented. This results in a restructuring of the initial first-order theory which now contains new possible concepts and has a more compact form.

After discussing related works in the next section, we recall some preliminaries in Section 3. Then, we describe our approach to the problem of Statistical Predicate Invention in Section 4, and provide experimental results on a standard ILP dataset in Section 5. Finally, Section 6 concludes the paper and proposes future work.

2 Related Works

Historically, the problem of PI originated in the ILP field, where typical approaches have been based on the analysis of the structure of clauses. In the classical vision of PI in ILP, predicates are invented by using different techniques based on the analysis of the structure of clauses. The goal is to obtain a compression of a first-order theory, or to improve the performance of inductive learning tasks, or to catch exceptions in learned clauses [9]. The problem has been explored analyzing first-order formulas, trying to apply restructuring operators such as inter-construction, looking for commonalities among clauses, as in [9,22], or intra-construction, looking for differences among clauses, as in [9,13]. Many of these works introduced PI in the context of learning systems such as Golem and Cigol [9]. Some results in this field also came from the work of Pazzani [19], in which the search for a predicate to invent is on a second-order instantiation of the logic theory. Other approaches to PI rely on the idea of finding tacit concepts that are useful to catch irregularities in the patterns obtained for a given induction problem [12,20].

Another stream of research that can be considered as related to PI was carried out in the pure SRL setting. Specifically, the proposal in [4] works in statistical learning. It consists of the search for hidden variables in a Bayesian network, by looking for structural patterns, and if a subsequent development using a clustering method to group observed variables and find hidden ones for each analyzed group. A task in SRL that is very similar to the task of PI in ILP is known as Hidden Variables Discovery. The approaches to PI proposed in this stream focus on the generation of concepts starting from facts. This makes them more relevant to the field of Hidden Variable Discovery, which is pure SRL, than they are to the original ILP concept of PI. So, the research ended up with proposing solutions that are more related to standard Parameter Learning, and specifically Structure Learning [8], than they are to PI. This caused the current cost/benefit balance of these works not clearly positive, although they are SPI are promising.

Recently some works better merged together the ideas of PI with those coming from Hidden Variable Discovery. In [8] attention is moved to the problem of SPI in a context of MLNs using a multiple clustering approach to group both

relations and constants in order to improve Structure Learning techniques for Markov Logic. Another approach [2] invents predicates in a statistical context and then exploits the invented predicate in the FOIL learning system. In [16] a particular version k-means is used to cluster separately constants and relations. Another proposal [15] involves the application of a number of techniques to cluster multi-relational data.

Only in the last few years there have been some developments in the field of SPI. Some of the proposed approaches use some version of relational clustering to aggregate concepts and relations into new ones [7]. E.g., [6] aims at avoiding the limitations of the purely ILP approach (that is prone to overfitting on noisy data, which causes the generation of useless predicates) using a statistical bottom-up only approach to invent predicates only in a statistical-relational domain.

3 Preliminaries

Our approach to SPI works in the first-order logic setting. Specifically, it uses Datalog (a function-free fragment of Prolog) as a representation language. The basic elements in this setting are atoms, i.e. claims to which a truth value can be assigned. An *atom* takes the form of a predicate applied to its arguments. The number of arguments required by a predicate is called its *arity*. The arguments of a predicate must be *terms*, that in Datalog can be only variables or constants. A *literal* is either an atom (positive literal) or a negation of an atom (negative literal). A *clause* is a disjunction of literals; if such a disjunction involves at most one positive literal then it is called a Horn clause, its meaning being that, if all atoms corresponding to negative literals (called the *body* of the clause) are true, then the positive literal (called the *head* of the clause) must be true. A clause may be represented as a set of literals. An atom, a literal or a clause is called *ground* if its terms are all constants. More information on this setting can be found in [1,10]

A clause made up of just the head is a *fact*, a clause made up of just the body is a *goal*, while a clause made up of a head and a non-empty body is a *rule*. A literal in a rule is linked if at least one of its terms is present also in other literals; a rule is linked if all of its literals are linked. We adopt the Object Identity (OI) assumption, by which terms (even variables) that are denoted by different symbols in a clause must refer to distinct objects[1]. So, a variable in a clause cannot be associated to another variable or constant in the same clause, nor can two variables in the same clause be associated to the same term.

So, for instance, $p(X, a, b, Y)$ is an atom (and also a positive literal), and $\neg p(X, a, b, Y)$ is a (negative) literal, where p is the predicate, the arity of p is n, and X, a, b, Y are terms (specifically, X, Y are variables and a, b are constants). $\{p(t_1)\}$ and $\{p(t_1), \neg q(t_1, t_2), \neg r(t_2)\}$ are (Horn) clauses. The former is a fact,

[1] While not causing a loss in expressive power [18], OI allows to define a search space for first-order logic machine learning systems that fulfills desirable properties ensuring efficiency and effectiveness. The ILP system we will use in the experiments works under OI.

and the latter is a (linked) rule, with head $p(t_1)$ and body $\{q(t_1, t_2), r(t_2)\}$. Under OI, in a clause $\{p(X), \neg q(X, Y)\}$ it is implicitly assumed that $X \neq Y$.

The vocabulary on which we build our representations is a triple $\mathcal{L} = \langle P, C, V \rangle$ where P is a set of predicate symbols, C is a set of constant symbols and V is a set of variable symbols, all possibly infinite. So, $C \cup V$ is the set of terms in our vocabulary. In the following, predicates and constants will be denoted by lowercase letters, and variables by uppercase letters.

A Markov Logic Network [17] consists of a set of first-order formulas associated to weights that represent their relative strength. Given a finite[2] set of constants C, the set of weighted first-order formulas can be used as a template for constructing a Markov Random Field (MRF) by grounding the formulas by the constants in all possible ways. The result is a graph in which nodes represent ground atoms, and maximal cliques are groundings of first-order formulas, also called 'features' in the MRF. The joint probability for atoms is given by:

$$P(X = x) = \frac{1}{Z} exp\left(\sum_{i=1}^{|F|} w_i n_i(x)\right) \tag{1}$$

where Z is a normalization constant, $|F|$ is the number of first order formulas in the set, $n_i(x)$ is the number of true groundings of formula F_i given the set of constants C, and w_i is the weight associated to formula F_i. In the Markov Logic framework two types of parameters learning can be distinguished: Structure Learning and Weight Learning. The former consists in inducing first-order formulas given a set of ground atoms as evidence and a query predicate, by maximizing the joint probability on the MRF based on the evidence. The latter consists in estimating the weight of each first-order formula given a first-order theory and the facts in the evidence, by maximizing the likelihood of a relational database using formula (1). The weight of a formula captures the importance of the formula itself, i.e. its inclination to be true in all possible worlds defined by the set of constants observed in the evidence. A complete description of inference and parameters learning techniques are discussed in [3].

4 Weighted Predicate Invention

As already pointed out, PI aims at generating of new symbols that define latent concepts in relational data or in a logic theory, defined in terms of the previously available symbols. Approaches to PI can be bottom-up (starting from ground atoms) or top-down (starting from a first-order theory). We propose a top-down approach to SPI based on using Markov Logic Networks, called *Weighted Predicate Invention (WPI)*. WPI is motivated by the observation that concepts that are latent in a logic theory may be useful in accomplishing tasks related to the

[2] This is a requirement set by MLNs. In our setting, the step that uses MLNs works on a given theory in which the set of constants used is finite, even if they were drawn from a possibly infinite set.

given problem, but not all such concepts are really significant. While this intuition is not novel [9], it is one of the major problems that PI has encountered so far. Our approach distinguishes significant concepts to be invented by their weight, obtained by considering all possible worlds according to evidence data.

In this context we chose the MLN framework, which is a well-known approach to combining FOL and probability. In MLNs, each rule is assigned a weight, that can be learned exploiting a convex optimization problem by the use of gradient descent techniques as in [11]. Discriminative Weight Learning aims at finding the weights that maximize the product of the rules' prior probabilities and data likelihood; so, weights determine the best characterization of a MLN in the worlds defined by the training set. Since the structure of first order rules plays a central role in the computation of weights, weights are particularly suited as a metric to validate the effectiveness of invented concepts.

The basic idea underlying WPI is to analyze an inductively learned theory consisting of a set of first-order rules under OI. It finds common patterns between the rules' bodies working in different steps. First a bipartite graph is created, whose nodes are distinguished between predicates and rules appearing in the theory. Then, a pattern is searched in the graph and checked for matching with subsets of literals in the body of rules. Specifically, th pattern consists of predicate nodes that may be involved in the definition of the predicate to be invented. This problem is complex due to the indeterminacy that characterizes the first-order logic setting. Indeed, it may not have a solution at all. If a matching is found, a candidate rule to be invented is built, and checked for validation before definitely including it in the existing first-order theory. The validation process is based on the use of Weight Learning in the MLN framework to assign a weight to rules in first-order theory. We produce two MLN, weights in the latter must be non-decreasing with respect the former to consider the new rule validated. In this case we can add the invented rule into the first-order theory.

WPI can be applied iteratively, inventing several predicates that allow to obtain more compact versions of the theory. Every invented predicate subsumes the body of rules involved in its invention.

4.1 Searching for a Pattern

Call R the set of first-order rules in the theory. Given a rule $r_i \in R$, let us denote by b_i the set of literals in the body of r_i, and by c_i a subset of b_i. Also, call P the set of all predicates in bodies of rules in the theory, where a literal l is represented as a pair $l = (p, V)$ with p its predicate and V the list of its arguments. We define a bipartite graph $G = \langle R \cup P, E \rangle$ where $E \subseteq R \times P$ is the set of edges such that $\{r, p\} \in E$ iff $\exists l = (p, V) \in R$. We call *upper nodes* the elements of R and *lower nodes* the elements of P. So, every rule is connected to all the predicate symbols appearing in its body.

To find commonalities in the bodies of clauses we consider all possible pairs $I = (\pi, \rho)$, where $\pi \subseteq P$ is a set of lower nodes (made up of at least two elements) that are connected to the same upper-node and $\rho \subseteq R$ is the set of rules in the theory that include π. Among all possible such pairs, we look for

one that maximizes (wrt set inclusion) π. Predicates appearing in such I's will be used to form a candidate pattern to define a predicate to be invented.

Consider a theory made up of three rules, $R = \{r_1, r_2, r_3\}$, where:

$$\begin{aligned} r_1 &: q(X) :- \quad a(X), b(Y), b(W), c(X,Y), d(Y,W). \\ r_2 &: q(X) :- a(X), b(W), c(X,Y), c(Y,W), g(X), h(Z,Y). \\ r_3 &: q(X) :- \quad a(X), f(Z,Y), h(X,Y). \end{aligned}$$

The following predicates are available in each rule:

$$\begin{aligned} r_1 &\rightarrow \quad \{a/1, b/1, c/2, d/2\} \\ r_2 &\rightarrow \{a/1, b/1, c/2, g/1, h/2\} \\ r_3 &\rightarrow \quad \{a/1, f/2, h/2\} \end{aligned}$$

After the building the bipartite graph, the maximal intersection of lower-nodes is $I = (\pi, \rho)$ where $\pi = \{a/1, b/1, c/2\}$ and $\rho = \{r_1, r_2\}$.

4.2 Candidate Selection

Given a pattern, we aim at finding a subset of literals in the theory rules that matches it. Clearly, only the rules in ρ are involved in this operation. For each rule $r_i \in \rho$, consider the subset of its literals that are relevant to the pattern, $c_i = \{l \in r_i | l = (p, V), p \in \pi\}$. Note that c_i may not be linked. For each predicate p_j in the pattern, call n_j the minimum number of occurrences across the c_i's. We try to invent predicates defined by n_j occurrences of predicate p_j, for $j = 1, \ldots, |\pi|$. The underlying rationale is that this should ensure higher chance to find that set of literals in the rules of the theory. In the previous example the minimum number of literals for all predicates in the pattern $\{a/1, b/1, c/2\}$ is one, thus the best subset of literals to match is $\{a(\cdot), b(\cdot), c(\cdot, \cdot)\}$.

Now, for each c_i, consider the set Γ_i of all of its subsets that include exactly n_j occurrences of each predicate p_j in π. We call each element of a Γ_i a *configuration*, and look for a configuration that is present in all Γ_i's (modulo variable renaming). More formally, we look for a $\overline{\gamma}$ s.t. $\forall i = 1, \ldots, |\rho| : \exists j_i \in \{1, \ldots, |\Gamma_i|\}$ s.t. $\overline{\gamma} \equiv \gamma_{ij_i} \in \Gamma_i$. In the example we have $\Gamma_1 = \{\gamma_{11}, \gamma_{12}\}$ and $\Gamma_2 = \{\gamma_{21}, \gamma_{22}\}$ where:

$$\gamma_{11} = \{a(X), b(Y), c(X,Y)\}, \gamma_{12} = \{a(X), b(W), c(X,Y)\}$$

$$\gamma_{21} = \{a(X), b(W), c(X,Y)\}, \gamma_{22} = \{a(X), b(W), c(Y,Z)\}$$

The existence of a solution is not guaranteed. In case it does not exist, we proceed by removing one occurrence of a predicate and trying again, until subsets two literals are tried (inventing a predicate defined by a single literal would be nonsense, since it would just be a synonym). For the proposed example the best configuration is $\gamma_{12} \equiv \gamma_{21}$.

The selected configuration $\overline{\gamma}$ becomes the body of the rule $\overline{r}$ that defines the predicate to be invented. If i is the name of the invented predicate, then the head

Algorithm 1. Finding matching clause

```
1: function CANDIDATE_SELECTION(π, ρ)
2:     bestConfig ← emptyList();
3:     maxIntersection ← 0;
4:     maxLiteralsInConfig ← 0;
5:     for all pred ∈ π do
6:         maxLiteralsInConfig ← maxLiteralsInConfig + minUsed(pred);
7:     Γ ← ⟨⟩;
8:     for all rule ∈ ρ do                                  ▷ A list of configurations
9:         Γ ← ⟨getAllConfigs(rule, π))|Γ⟩;
                          ▷ candidateConfig is a list of configurations from every rule.
10:    for all candidateConfig in cartesianProduct(Γ) do
11:        intersection ← setIntersection(candidateConfig);
12:        if |intersection| > maxIntersection then
13:            maxIntersection ← |intersection|;
14:            bestConfig ← candidateConfig;
15:        if maxIntersection = maxLiteralsInConfig then
16:            break;
17:    return bestConfig;
```

of such a rule is obtained by applying i to a list of arguments corresponding to the different variables in $\overline{\gamma}$. In the example the rule would be

$$i(X, Y, W) \ :- \ a(X), b(W), c(X, Y).$$

After adding this rule to the theory, for each rule $r_i \in \rho$ the configuration $\overline{\gamma_i} \in \Gamma_i$ equivalent to $\overline{\gamma}$ can in principle be removed and replaced by the corresponding instance of predicate i. The whole procedure is sketched in Algorithm 1.

4.3 Candidate Validation

Let us call the new rule r_0, that defines the invented predicate, the *invented rule*. To prevent the invention of useless predicates, a validation step must be run that determines whether the invented predicate is actually relevant. Such a validation should be based on an estimator of the relevance of a rule in the context defined by the given theory and the set of evidence facts in the background knowledge. Given such an estimator, the idea is that the introduction of the invented rule in the original theory must not decrease the relevance of the existing rules.

We propose to use the weights learned by the weight learning functionality of the MLN framework as an estimator, and proceed by building two MLNs. For each MLN we apply Discriminative Weight Learning to maximize the likelihood of the training dataset using a learning process based on Diagonal Newton as in [11]. The former simply adds the invented rule to the initial theory. The latter also applies the invented rule to the existing rules, replacing the subset of literals in each rule, that match the invented rule body, with the head of the invented rule properly instantiated. In the previous example, one would get:

$$\begin{aligned} r_0 : i(X,Y,W) :- & \quad a(X), b(W), c(X,Y). \\ r_1 : q(X) :- & \quad b(Y), d(Y,W), i(X,Y,W). \\ r_2 : q(X) :- & \quad c(Y,W), g(X), h(Z,Y), i(X,Y,W). \\ r_3 : q(X) :- & \quad a(X), f(Z,Y), h(X,Y). \end{aligned}$$

In the former, the invented rule is disjoint from the rest of the graph, because the invented predicate is not present in the other rules. This causes the weights of the other rules not to change. In the latter, the body of some rules in the original theory has changed so that the invented rule is no more disjoint in the graph. So, we expect a variation of the rule weights in the two cases. Comparing the two weights, we consider the invented predicate as relevant if the weight in the latter template is greater than the weight in the former.

We run Discriminative Weight Learning on both templates, obtaining two sets of weighted first-order rules with respect to the evidence facts and the query predicate for the problem defined by theory. Call $w'_0, w'_1, \ldots, w'_k$ the weights of rules in the former MLN, and $w''_0, w''_1, \ldots, w''_k$ the weights of rules in the latter MLN. Then, we pairwise compare the weights of rules, and specifically the weight of the invented rule and the weights of the rules in ρ involved in the predicate invention process. The invented rule is considered as validated if no weight after the application of the invented predicate is less than it was before:

$$\forall i = 0, \ldots, k : w'_i \leq w''_i$$

Otherwise, if the introduction of the invented rule in the initial theory causes a decrease in the relative importance of any rule, then the invented rule is not added to the theory. Given the new theory, WPI can be run again on it in order to invent, if possible, further predicates that define implicit concepts. Iterating this procedure yields a wider theory restructuring.

4.4 Discussion

A typical problem of PI approaches is the risk of combinatorial explosion for the search space of the groups of literals that define the invented predicate. Instead of analyzing this problem from a theoretical or structural viewpoint, in this paper we propose an operational model that avoids the invention of trivial or useless concepts. Nevertheless, the computational complexity forces us to consider the advantages taken by the use of WPI in a context of *learning for restructuring* for possible applications that are discussed in the final section.

The main cause of the possible combinatorial explosion is in the variable number of literals per predicate for each rule in ρ. The higher the number of literals per predicate, the higher the number of possible configurations γ_i. Another limitation is in the cost of evaluating Discriminative Weight Learning twice for every predicate we can invent, in fact this kind of parameters learning consists in numerical optimization problem on the MRF instantiated by the correspondent MLN [3].

5 Experiments

The WPI approach was implemented to test its effectiveness, both as regards predicate invention and as regards theory restructuring. To manage MLNs and Discriminative Weight Learning, WPI relies on Tuffy [14], an implementation of the MLN framework based on a Relational DBMS to scale up learning. Compared to Alchemy [21], Tuffy exploits a simplified syntax for MLNs and provides a Discriminant Weight Learning technique to learn weights starting from a template, the data and a query predicate. Then, an experiment was devised. To simulate a totally automatic setting, the theories for the experiment were learned using InTheLEx [5], a fully incremental, non-monotonic, multi-strategy learner of First-Order Logic (Datalog) theories from positive and negative examples, based on the OI assumption.

A toy dataset was exploited in the experiment, for a twofold reason. First, because of the computational complexity of the SPI approach. Second, because it allows an easier insight into the results. Specifically, we considered the classical *Train Problem*, well-known in ILP, in its extended version given by Muggleton. It includes 20 examples of Eastbound or Westbound trains, with the goal to predict Eastbound ones. Due to the small size of this dataset we applied a

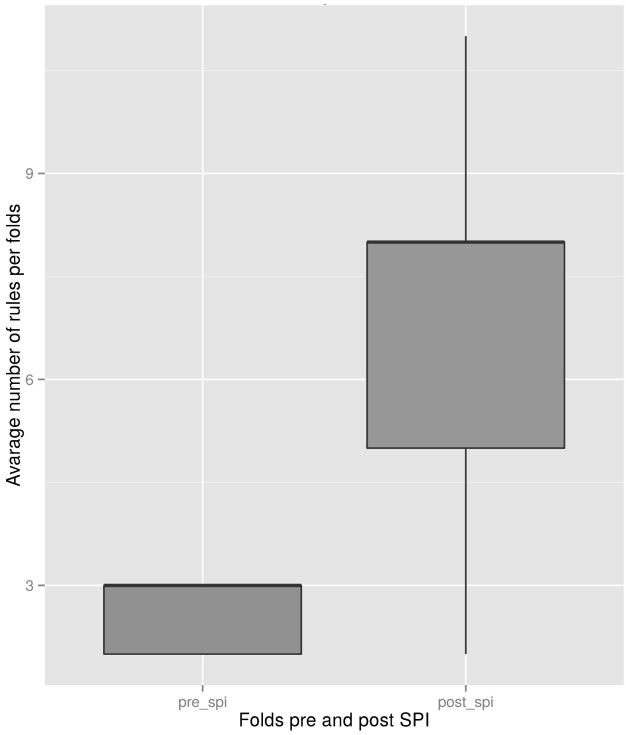

Fig. 1. Number of rules per fold in the theory before and after using WPI

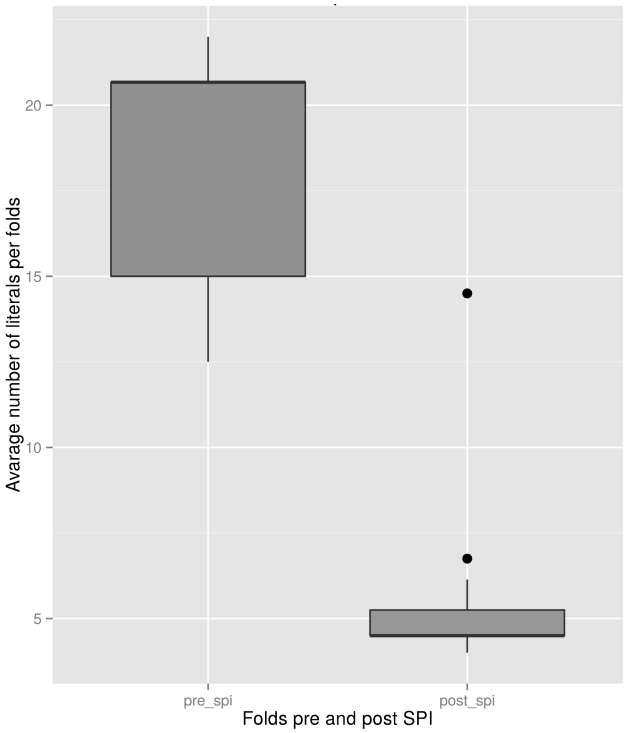

Fig. 2. Number of literals per rule in folds before and after using WPI

Leave-One-Out Cross-Validation technique to avoid the problem of overfitting. This resulted in a total of 20 folds, each using one example for testing purposes and the remaining 19 for training. For each fold, InTheLEx was run on the training set to inductively learn a theory, which was to be tested on the remaining example. Of course, each fold resulted in a different theory, which in turn allowed different possibilities to predicate invention. So, we applied WPI to each learned theory and compared the resulting restructured theory with the one originally provided by InTheLEx. Experimental results confirmed that different theories were learned in the different folds, yielding different invented predicates and, thus, different restructured theories.

The quantitative comparison focused on the number of new (invented) concepts, the number of rules in the theories before and after restructuring, and the number of literals per rule in these theories. A statistical analysis of the outcomes is reported in Figures 1 and 2. The boxplot reported in Figure 1 clearly shows that the number of rules in the theories significantly increases on average after invention and restructuring. However, the number of invented predicates/rules shows some variability in the different folds, including folds in which no predicate could be invented at all. Applying the invented predicates to the initial theory should result in a compression of the theory itself, because the definition

of the invented predicate is removed from the other rules and replaced by a single literal that is an instance of the invented predicate. The significant compression obtained in the experiment can be appreciated from the boxplot in Figure 2, concerning the number of literals per rule. So, overall, as expected, the number of rules increases on average, but their size decreases on average. Specifically, 4.25 new concepts were invented on average in each of the 20 folds, which more than doubled the size of the theories on average. However, the average number of literals per rule dropped from 18.41 to 5.30 on average, which (also considering the increase in number of rules) yields an average compression factor of 28.79%.

From a qualitative point of view, we looked for possible interesting new concepts emerging from the inductively learned theories. An interesting result is, for example, the invention in many folds of the concept that any railway car in the train is somehow connected to the locomotive: $\{car(Car), has\_car(Train, Car)\}$. This concept catches a general intuition about trains, and factorizing it makes the theory more understandable.

6 Conclusions and Future Works

Predicate Invention is the branch of symbolic Machine Learning aimed at discovering new emerging concepts in the available knowledge. The outcome of this task may have important consequences on the efficiency and effectiveness of many kinds of exploitation of the available knowledge. Two fundamental problems in Predicate Invention are how to handle the combinatorial explosion of candidate concepts to be invented, and how to select only those that are really relevant. Due to the huge number of candidates, there is a need for automatic techniques to assign a degree of relevance to the various candidates and select the best ones. Purely logical approaches may be too rigid for this purpose, while statistical solutions may provide the required flexibility. This paper proposed a new Statistical Relational Learning approach to Predicate Invention. The candidate predicates are identified in a logic theory, rather than in the background knowledge, and are used to restructure the theory itself. Specifically, the proposed approach exploits the Markov Logic Networks framework to assess the relevance of candidate predicate definitions. It was implemented and tested on a traditional problem in Inductive Logic Programming, yielding interesting results in terms of invented predicates and consequent theory restructuring.

Future work will include: optimizing the implementation; devising more informative heuristics for selecting and choosing candidate definitions for the predicate to be invented; running more extensive experiments, also on real-world datasets. Specifically, we want to investigate the implications of using WPI on the benchmark Mutagenesis dataset, to check whether interesting implicit concepts can be found in that domain.

Acknowledgments. This work was partially funded by the Italian PON 2007-2013 project PON02_00563_3489339 'Puglia@Service'.

References

1. Ceri, S., Gottlob, G., Tanca, L.: Logic Programming and Databases. Springer-Verlag (1990)
2. Craven, M., Slattery, S.: Relational learning with statistical predicate invention: Better models for hypertext. Machine Learning **43**(1/2), 97–119 (2001)
3. Domingos, P., Kok, S., Lowd, D., Poon, H., Richardson, M., Singla, P.: Markov logic (2008)
4. Elidan, G., Friedman, N.: Learning hidden variable networks: The information bottleneck approach. Journal of Machine Learning Research **6**, 81–127 (2005)
5. Esposito, F., Semeraro, G., Fanizzi, N., Ferilli, S.: Multistrategy Theory Revision: Induction and Abduction in INTHELEX. Machine Learning **38**(1/2), 133–156 (2000)
6. Kemp, C., Tenenbaum, J.B., Griffiths, T.L., Yamada, T., Ueda, N.: Learning systems of concepts with an infinite relational model. In: AAAI, pp. 381–388. AAAI Press (2006)
7. Kok, S., Domingos, P.: Toward statistical predicate invention. In: Open Problems in Statistical Relational Learning, SRL 2006 (2006)
8. Kok, S., Domingos, P.: Statistical predicate invention. In: Ghahramani, Z. (eds.) ICML. ACM International Conference Proceeding Series, vol. 227, pp. 433–440. ACM (2007)
9. Kramer, S.: Predicate invention: A comprehensive view. Technical Report TR-95-32, Oesterreichisches Forschungsinstitut fuer Artificial Intelligence, Wien, Austria (1995)
10. Lloyd, J.W. : Foundations of Logic Programming, 2nd edn. Springer-Verlag (1987)
11. Lowd, D., Domingos, P.: Efficient weight learning for Markov logic networks. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 200–211. Springer, Heidelberg (2007)
12. Muggleton, S.: Predicate invention and utilization. J. Exp. Theor. Artif. Intell **6**(1), 121–130 (1994)
13. Muggleton, S., Buntine, W.: Machine invention of first-order predicates by inverting resolution. In: MLC 1988, pp. 339–352 (1988)
14. Niu, F., Ré, C., Doan, A., Shavlik, J.W.: Tuffy: Scaling up statistical inference in markov logic networks using an RDBMS (2011). CoRR, abs/1104.3216
15. Perlich, C., Provost, F.: Aggregation-based feature invention and relational concept classes. In: KDD 2003, pp. 167–176. ACM Press (2003)
16. Popescul, A., Ungar, L.H.: Cluster-based concept invention for statistical relational learning. In: KDD, pp. 665–670. ACM (2004)
17. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning **62**(1–2), 107–136 (2006)
18. Semeraro, G., Esposito, F., Malerba, D., Fanizzi, N., Ferilli, S.: A logic framework for the incremental inductive synthesis of datalog theories. In: Fuchs, N.E. (ed.) LOPSTR 1997. LNCS, vol. 1463, pp. 300–321. Springer, Heidelberg (1998)

19. Silverstein, G., Pazzani, M.: Relational cliches: constraining constructive induction during relational learning. In: Proceedings of the Sixth International Workshop on Machine Learning. Kaufmann, Los Altos (1989)
20. Srinivasan, A., Muggleton, S., Bain, M.: Distinguishing exceptions from noise in non monotonic learning. In: Rouveirol, C. (eds.) ECAI 1992 Workshop on Logical Approaches to Machine Learning (1992)
21. Sumner, M., Domingos, P.: The alchemy tutorial, July 26, 2013
22. Wogulis, J., Langley, P.: Improving efficiency by learning intermediate concepts. In: IJCAI 1989, pp. 657–662. Morgan Kaufmann (1989)