Affordable Discovery of Positive and Negative Rules in Knowledge-Bases

us

ABSTRACT

We present KRD, a system for the discovery of declarative rules over knowledge-bases (KBs). KRD does not limit its search space to rules that rely on "positive" relationships between entities, such as "if two persons have the same parent, they are siblings", as in traditional mining of constraints for KBs. On the contrary, it extends the search space to discover also negative rules, i.e., patters that lead to contradictions in the data, such as "if two person are married, one cannot be the child of the other". While the former class is fundamental to infer new relationships in the KB, the latter class is crucial for error detection in data cleaning, or for the creation of negative examples when bootstrapping learning algorithms.

The main technical challenges addressed in this paper consist in enlarging the expressive power of the considered rules to include comparison among constants, including disequalities, and in designing a disk-based discovery algorithm, effectively dropping the assumption that the KB has to fit in memory to have acceptable performance. To guarantee that the entire search space is explored, we formalize the mining problem as an incremental graph exploration. Our novel search strategy is coupled with a number of optimization techniques to further prune the search space and efficiently maintain the graph. Finally, in contrast with traditional ranking of rules based on a measure of support, we propose a new approach inspired by set cover to identify the subset of useful rules to be exposed to the user. We have conducted extensive experiments using both real-world and synthetic datasets to show that KRD outperform previous proposals in terms of efficiency and that it discovers more effective rules for the application at hand.

1. INTRODUCTION

2. PRELIMINARIES AND DEFINITIONS

Talk about KBs.

2.1 Language

Horn Rule, with the restriction of having each variables appearing twice. Extension of predicates with inequalities.

A Horn Rule r has the form $A_1 \wedge A_2 \wedge \cdots A_n \Rightarrow r(a,b)$, where $A_1 \wedge A_2 \wedge \cdots \wedge A_n = r_{body}$ is the *body* of the rule.

2.2 Coverage

Given a pair of entities (x,y) from the KB and a Horn Rule r, we say that r_{body} *covers* (x,y) if $(x,y) \models r_{body}$. In other words, given a Horn Rule $r = r_{body} \Rightarrow r(a,b)$, r_{body} covers a pair of entities (x,y) iff r_{body} can be instantiated over the KB by substituting a with x and b with y. Given a set of pair of entities $E = \{(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)\}$ and a rule r, we denote by $C_r(E)$ the *coverage* of r_{body} over E as the set of elements in E covered by r, $C_r(E) = \{(x,y) \in E | (x,y) \models r_{body}\}$.

Given the body r_{body} of a Horn Rule r, we denote by r^*_{body} the *unbounded body* of r. The unbounded body of a rule is obtained by substituting each atom in r that contains either variable a or b with a new atom where the other variable that is not a or b is substituted with another unique variable. As an example, given $r_{body} = rel_1(a,b)$, $r^*_{body} = rel_1(a,v_1) \wedge rel_1(v_2,b)$. Paolo: I suggest to have $rel_3(a,b)$ to avoid the confusion raised by cartesian product Stefano: Better now? Given a set of pair of entities $E = \{(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)\}$ and a rule r, we denote by $U_r(E)$ the *unbounded coverage* of r^*_{body} over E as the set of elements in E covered by r^*_{body}, $U_r(E) = \{(x,y) \in E | (x,y) \models r^*_{body}\}$.

Example 1. *Given the rule* $r =$ `hasChild`$(a,v_0) \wedge$`hasChild`(b,v_0) *and a KB* K*, we denote by* E *the set of all possible pairs of entities in* K*. The coverage of* r *over* E $(C_r(E))$ *is the set of all pairs of entities* (x,y) *where both* x *and* y *are in relation* `hasChild` *with the same entity* v_0*, while the unbounded coverage of* r *over* E $(U_r(E))$ *is the set of all pairs of entities* (x,y) *where* x *is in relation* `hasChild` *with an entity* v_1 *and* y *is in relation* `hasChild` *with an entity* v_2*, and not necessarily* $v_1 = v_2$*.*

Stefano: Explain why unbounded coverage is important

Similarly, the coverage and the unbounded coverage for a set of rules $R = \{r_1, r_2, \cdots, r_n\}$ is the union of individual coverages:

$$C_R(E) = \bigcup_{r \in R} C_r(E) \qquad U_R(E) = \bigcup_{r \in R} U_r(E)$$

We can now formalize the *exact discovery problem*. Given a KB K, a set of pair of entities G, a set of pair of entities V,

and a universe of rules R, a solution for the *exact discovery problem* is a subset R' of R such that:

$$R_{opt} = \underset{|R'|}{\text{argmin}}(R' | (C_{R'}(G) = G) \wedge (C_{R'}(V) \cap V = \varnothing))$$

G is the *generation set*, which contains good examples for the rule that we are trying to discover (G contains examples of married couples if we are discovering rules for a spouse relation). V is the validation set, which contains counter examples for the target rule (pairs of people that are not in a married relation). The ideal solution is a set of rules that covers all examples in G, and none of the examples in V. Note that given a pair of entities (x, y), we can always generate a Horn Rule whose body covers only (x, y) by assigning variable a to x and variable b to y.

Unfortunately, since the solution is not allowed to cover any element in V, in the worst case the exact solution may be a set of rules s.t. each rule covers only one example in G, making such set of rules difficult to use.

2.3 Scoring Function

In order to allow flexibility and errors in both G and V, we drop the strict requirement of not covering any element of V. However, since covering elements in V is an indication of potential errors, we want to limit the coverage over V to the minimum possible. We therefore define a *weight* to be associated with a rule.

Given a KB K, two sets of pair of entities G and V from K where $G \cap V = \varnothing$, and a Horn Rule r, the weight of r is defined as follow:

$$w(r) = \alpha \cdot (1 - \frac{|\, C_r(G)\,|}{|\, G\,|}) + \beta \cdot (\frac{|\, C_r(V)\,|}{|\, U_r(V)\,|}) + \gamma \cdot (1 - \frac{|\, U_r(V)\,|}{|\, V\,|})$$

Paolo: if we want to minimize it, it should be defined as a cost function Stefano: weight ok? Paolo: here goes the description of the three intuition behind the three components, why they are needed, etc. Show with example introduce in intro

Similarly, the weight for a set of rules R is defined as:

$$w(R) = \alpha \cdot (1 - \frac{|\, C_R(G)\,|}{|\, G\,|}) + \beta \cdot (\frac{|\, C_R(V)\,|}{|\, U_R(V)\,|}) + \gamma \cdot (1 - \frac{|\, U_R(V)\,|}{|\, V\,|})$$

2.4 Problem Definition

We can now state the approximate version of the problem.

Given a KB K, two sets of pair of entities G and V from K where $G \cap V = \varnothing$, a universe of rules R, and a w weight function for R, a solution for the *approximate discovery problem* is a subset R' of R such that:

$$R_{opt} = \underset{w(R')}{\text{argmin}}(R' | R'(G) = G)$$

We can map this problem to the well-known weighted set cover problem, which is proven to be an NP-Complete problem [1], where the universe is G and the sets are all the possible rules defined in R.

Stefano: Discuss what the approximate version of the problem is trying to accomplish: cover all elements in G, and as few as possible in V. Discuss that in the worst case there will not exists good rules that cover more than one element in V and few elements in G. In those cases for those elements in V that cannot be covered, there will be a single rule for each element.

Section 4 will describe a greedy polynomial algorithm to find a good solution for our problem.

3. RULES DISCOVERY

Talk about translation from Horn Rules to paths on the graph.

3.1 Literals and Constants

Generation of artificial edges to include inequalities. Substitions of variables with constants if same value appears for each example.

3.2 Input Examples Generation

Define how we compute generation and validation set: how we generate positive and negative examples.

4. A* GREEDY ALGORITHM

Since the universe of all possible rules R is too big to enumerate, we solve the online variant of the above problem. Paolo: if offline problem is NP, online is at least NP; to be verified Stefano: Shouldn't we just say that the problem is NP therefore we go for a greedy algorithm that allows also to avoid the enumeration of all rules?

4.1 Optimality

Define property on why the A* algorithm produces the greedy solution. Maybe study when the greedy solution become optimal? (If all rules identify disjoint set of input example, then greedy solution is optimal)

5. EXPERIMENTS

5.1 Negative Rules Evaluation

Evaluation of negative rules.

5.2 Comparison Evaluation

Comparison against AMIE and evaluation of positive rules.

5.3 Machine Learning Application

DeepDive.

6. RELATED WORK

7. CONCLUSION

8. REFERENCES

[1] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233–235, 1979.