Essence of Factual Knowledge

Ruoyu Wang[1,2,3] Daniel Sun[2,3,4] Guoqiang Li[1] Raymond Wong[2] Shiping Chen[4]
[1]Shanghai Jiao Tong University, Shanghai, China
[2]University of New South Wales, Sydney, Australia
[3]Enhitech LLC., Shanghai, China
[4]Data61, CSIRO, Australia
{wang.ruoyu, li.g}@sjtu.edu.cn, {daniel.sun, shiping.chen}@data61.csiro.au, wong@cse.unsw.edu.au

I. Introduction

Knowledge bases are collections of domain-specific and commonsense facts. Recently, the sizes of KBs are rocketing due to automatic extraction for knowledge and facts. For example, the number of facts in WikiData is up to 974 million! According to our observation, current KBs, especially domain KBs, show strong relevance in relations according to some topics[1], [2]. These patterns can be used to conclude and infer for part of facts in the KBs. Therefore, the original KBs can be minimzed by extracting patterns and essential facts.

In this paper, we introduce a framework for extracting knowledge essence and reducing overall volume of KBs by mining semantic patterns in relations. Facts are formalized as first-order predicates and patterns are induced as Horn rules.

Table I and Rule (1), (2) show an example of such extraction. By extracting the rules from listed facts, both table Ib and Ic can be inferred from other tables and then be removed.

The remaining is organized as follows: Section II analysed properties of rules as equivalence classes. Essence extraction problem is formally defined in Section III. And Section IV introduces the basic framework for essence extraction. Finally Section V concludes the paper.

II. Properties of Horn Rules

A. Semantic Length and Fingerprint of a Rule

First-order Horn rules are adopted in our technique to describe semantic patterns in relations. They can further be decomposed into equivalence classes. Elements in each of the classes are arguments that are assigned to the same variables, and if some argument is assigned to constants, then the corresponding equivalence class only consists of the argument and the constant. For example, Rule (1) is decomposed to the following equivalent classes (number in the brackets denotes the argument index of certain relation, starting from 0):

$$
\begin{aligned}
X :& \{father[0], parent[0], male[0]\} \qquad && father[0] \\
Y :& \{father[1], parent[1]\} && father[1]
\end{aligned}
$$

The length of a rule is defined by the follwoing equation:

$$|r| = \sum_i (|C_i| - 1) \tag{3}$$

where C_i is one of these equivalence classes.

Fingerprints of rules are based on the equivalence classes with labels of arguments in the head that it applies to. For example, the last column of the above example shows the label of head arguments.

Lemma 1. *Two rules are semantically equivalent if and only if their fingerprints are identical.*

Proof. (Necessity)If two rules are semantically equivalent, they can be written in syntactically identical form. Thus equivalence classes of corresponding variables or constants are identical.

(Sufficiency)Each equivalence class tells position of one variable. Therefore, equivalence of all classes ensures that the set of predicates in both rules are identical. The labels of head arguments further determine that the head predicates are the same. Thus, the two rules are identical. □

B. Search Space for Rules

Let Ω be the search space for first-order Horn rules. Some elements in Ω make no sense and should be excluded. If some predicate in the body is identical to the head, then the predicate in the body is redundant. These rules are trivial rules. If some subset of the body does not share any variable with the remaining part (include the head), then the rule is either redundant nor unsatisfiable. The subset is called independent fragment. The new search space excluding these two types of rules is written as Ω_m.

C. Extension on Rules

Definition 2 (Limited Variable, Unlimited Variable, Generative Variable)**.** *A variable is unlimited in some Horn rule r if there is only one argument in r that is assigned to it. A variable is limited in r if there exist at least two arguments in r that are assigned to it. A variable is generative if there exist arguments in both the head and body of r that are assigned to it.*

Searching for rules starts from most general forms, i.e. rules only with head predicate and arguments in the predicate are all unique unlimited variables. To construct new rules, new equivalence conditions are added to the equivalence classes. Syntactically, these operations fell in five extension operations, which is noted by $ext(r)$:

Case 1: Assign an existing limited variable to some argument.

Case 2: Add a new predicate with unlimited variables to the rule and then assign an existing limited variable to one of these arguments.

TABLE I: Partial data on server configurations and status

(a) parent/2

parent	child
james	harry
lily	harry
harry	sirius
harry	albus
ginny	sirius
ginny	albus

(b) father/2

father	child
james	harry
harry	sirius
harry	albus

(c) mother/2

mother	child
lily	harry
ginny	sirius
ginny	albus

(d) male/1

person
james
harry
albus
sirius

(e) female/1

person
lily
ginny

$$father(X,Y) \leftarrow parent(X,Y), male(X) \tag{1}$$
$$mother(X,Y) \leftarrow parent(X,Y), female(X) \tag{2}$$

Case 3: Assign a new limited variable to a pair of arguments.
Case 4: Add a new predicate with unlimited variables to the rule and then assign a new limited variable to a pair of arguments. In this case, the two arguments are not both selected from the newly added predicate.
Case 5: Assign a constant to some argument.

According to the rule extension, $\forall r, r_e \in \Omega_m$, if $r_e \in ext(r)$, then r_e is the extension of r, and r is the origin of r_e (denoted as $r \in ext^{-1}(r_e)$ since one may have multiple origins). Neighbours of a rule in Ω_m consist of all its extensions and origins. The above extension operations can be used to search on Ω_m. Let $S = \{r|r$ has only a head predicate p and all arguments of p are unlimited variables$\}$, every element in Ω_m can be searched from some $r_0 \in S$. To prove this we define a property *link* between predicates in a certain rule: If two predicates p and q in a rule r share a limited variable X, then p and q are linked by X in r, written as $p \diamond_X q$, or in short $p \diamond q$. Moreover, if there is a sequence of predicates $p \diamond w_0 \diamond \cdots \diamond w_k \diamond q$, then there is a *linked path* between p and q, written as: $p \leftrightarrow^{\diamond} q$. With this property, we can prove the search completeness as follows:

Lemma 3. *$\forall r \in \Omega_m$, every predicate in r has a linked path with the head of r.*

Proof. Suppose a predicate p in rule r has no linked path with the head. Then p is not itself the head. Let $P = \{q|p \leftrightarrow^{\diamond} q\}$, every predicate in P has no linked path with the head. Then the fragment noted by P does not share any variables with remaining predicates. Namely, P denotes an independent fragment in rule r. According to the definition of Ω_m, we have $r \notin \Omega_m$, which contradicts with $r \in \Omega_m$. ☐

Lemma 4. *(Search Completeness)Let $S = \{r|r$ has only a head predicate p and all arguments of p are unlimited variables$\}$, $\forall r \in \Omega_m, \exists r_0, r_1, \ldots, r_n \in \Omega_m$, such that $r_0 \in S, r_1 \in ext(r_0), \ldots, r \in ext(r_n)$.*

Proof. Suppose $p \diamond_X q$ in r. During the searching process of r, when p is already in a intermediate status r', an extension of r' can be constructed by adding a new predicate q and turning corresponding variables to X. Thus, predicate q is introduced into r'. Therefore, if $w \leftrightarrow^{\diamond} q$ and w is already in a intermediate status, then q can be introduced into r'. According to Lemma 3, all predicates in r has linked path with its head. Each predicate can be introduced into the rule iteratively starting from the head predicate where arguments are all different unlimited variables. Other limited variables and constants can be added to the rule by other extension operations to finally construct r. ☐

Rules with independent fragments will not be constructed starting from S, as the extension operations do not introduce new predicates without any shared variables with other predicates.

III. Problem Definition

Definition 5 (Essential Knowledge Extraction)**.** *Let B be the original KB, which is a finite set of atoms. The extraction on B is a triple (H, N, C), where H (for "Hypothesis") is the set of first-order Horn rules, N (for "Necessary") is a subset of B, and C (for "Counter Examples") is a subset of the complement of B subject to CWA. B, H, N, C satisfies ($\models$ is logical entailment):*

- $N \wedge H \models (B \setminus N) \cup C$
- $\forall e \notin B \cup C, N \wedge H \not\models e$
- $|N| + |C| + |H|$ *is minimal*

where $|N|$ is the number of predicates in N, and so be $|C|$. $|H|$ is defined as the sum of lengths of all rules in it.

Definition 6 (Minimum Vertex Cover Problem)**.** *Let $\mathcal{G}_{vc} = \langle V_{vc}, E_{vc} \rangle$ be an undirected graph. A minimum vertex cover V_c of $\mathcal{G}_{vc}$ is a minimum subset of V_{vc} such that $(u, v) \in E_{vc} \implies u \in V_c \vee v \in V_c$.*

Complexity of essence extraction can be proved by reducing minimum vertex cover problem to relational compression. Let $\mathcal{G}_{vc} = \langle V_{vc}, E_{vc} \rangle$ be the graph in the vertex cover problem. By the following settings we create a relational knowledge base aligning with $\mathcal{G}_{vc}$: Let v be a unary predicate in B for each $v \in V_{vc}$; let *edge* be a unary predicate in B for edges; add two constants e_{ij} and e'_{ij} to C and six predicates $edge(e_{ij})$, $edge(e'_{ij})$, $v_i(e_{ij})$, $v_i(e'_{ij})$, $v_j(e_{ij})$, $v_j(e'_{ij})$ to B for each $(v_i, v_j) \in E_{vc}$; add the following predicates to B:

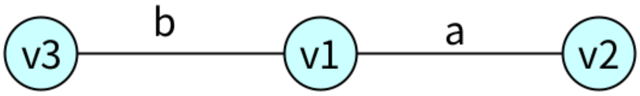

Fig. 1: Vertex Cover Example

$edge(c_1), edge(c_2), \ldots, edge(c_{2\cdot|E_{vc}|+1})$; and add the following constants to C: $d_1, d_2, \ldots, d_{4\cdot|E_{vc}|+1}$.

For example, Figure 1 shows a graph with three vertices and two edges. The corresponding setting of relational compression is as follows:

- $C = \{a, a', b, b', c_1, \ldots, c_5, d_1, \ldots, d_9\}$
- $B = \{v_1(a)$, $v_1(a')$, $v_2(a)$, $v_2(a')$, $v_1(b)$, $v_1(b')$, $v_3(b)$, $v_3(b')$, $edge(a)$, $edge(a')$, $edge(b)$, $edge(b')$, $edge(c_1)$, $\ldots$, $edge(c_5)\}$

By reducibility from minimum vertex cover problem to relational compression we can prove the latter is NP-hard. The details are as follows:

Lemma 7. *$edge(X) \leftarrow true$ is not in H.*

Proof. Let $arg^+(p) = \{c \in C | p(c) \in B\}$, then $|arg^+(edge)| = 2n + 2n + 1 = 4n + 1$, where $n = |E_{vc}|$. Thus, the number of predicates this rule entails is $4n + 1$. Taking constants $d_1, \ldots, d_{4n+1}$ into consideration, the number of counter examples this rule entails is also $4n + 1$. The size reduced is $4n + 1 - (4n + 1) - 1 = -1$, no actual reduction. Therefore, it does not reduce the size of knowledge base. It is not in H. □

Lemma 8. *Predicates of $edge$ can only be entailed by the following rules: $edge(X) \leftarrow v_i(X)$.*

Proof. Let rule r_i be: $edge(X) \leftarrow v_i(X)$, the length of which is 1. Then the number of predicates it entails is $2k$, where k is the number of edges connected to vertex v_i. There are no counter examples entailed by this rule. Thus the size it reduces is $2k - |r_i| = 2k - 1$. If $k \geq 1$, this rule can be used to reduce the size of knowledge base.

According to Lemma 7, $edge$ cannot be entailed by axioms, and since there is no other predicate in B, $edge$ can only be entailed by some v_i. □

Lemma 9. *Let $S = \{edge(e) | edge(e) \in B\} \setminus \{edge(c) | \exists c_i = c\}$. All predicates in S are provable after compression. That is, $S \subseteq R$, where R is the set of all provable predicates.*

Proof. According to Lemma 8, proof of $edge(e) \in S$ relies only on predicates of v_i. No matter predicates of v_i is provable or not, the rules of $edge(X) \leftarrow v_i(X)$ can always be applied to prove $edge(e) \in S$. Suppose $\exists edge(e) \in S$ such that $edge(e) \notin R$. Then there is another predicate $edge(e') \in S$ and $edge(e') \notin R$, where e and e' correspond to some edge in E_{vc} and its duplicate, since these two predicates are both entailed by some rule $edge(X) \leftarrow v_i(X)$ if one of them is entailed by the rule. Then a new rule can be applied to entail these two predicates to further reduce the size of given result. However, according to definition of relational compression, output cannot be further reduced. Contradiction occurs. □

Algorithm 1 Essence Extraction

Input: Knowledge Base B
Output: Summarization on B: (H, N, C)

$H \leftarrow \varnothing$
$C \leftarrow \varnothing$
$\mathcal{G} \leftarrow \langle B \cup \{\top\}, \varnothing\rangle$
while $r \leftarrow findSingleRule(B)$ **do**
    $H \leftarrow H \cup \{r\}$
    $C \leftarrow C \cup E_r^-$
    Update graph $\mathcal{G}$ with respect to r
end while
$cc \leftarrow CoverCycle(\mathcal{G})$
$N \leftarrow \{h \in V \setminus \{\top\} | \forall b \in V, (b, h) \notin E\} \cup cc$
return (H, N, C)

Lemma 10. *Let V_c be the solution of minimum vertex cover problem. Let H_{vc} be a rule set and $H_{vc} = \{edge(X) \leftarrow v(X) | v \in V_c\}$. Let $\bar{H}_{vc}$ be a rule set and $\bar{H}_{vc} = \{edge(X) \leftarrow v(X) | v \notin V_c\}$. Then $H_{vc} \subseteq H$ and $\bar{H}_{vc} \cap H = \varnothing$.*

Proof. According to Lemma 8 and 9, all edges are provable and only provable by vertices, and this is equal to the setting that all edges are covered and only covered by vertices for minimum vertex cover problem. Thus H_{vc} entails S in a minimum cost. $H_{vc} \subseteq H$ and $\bar{H}_{vc} \cap H = \varnothing$. □

Theorem 11. *Relational compression is NP-hard.*

Proof. Let V_c be the set of minimum vertex cover of $\mathcal{G}_{vc}$. According to the lemmas above, $V_c = \{v \in V_{vc} | \exists edge(X) \leftarrow v(X) \in H\}$. All the operations involved with reducibility are with polynomial cost. Thus minimum vertex cover problem can be polynomially reduced to relational compression. Relational compression is NP-hard. □

IV. Extraction Framework

To tell whether a fact is provable by others, we employ a directed graph $\mathcal{G} = \langle V, E\rangle$ to encode dependency among facts with respect to inference. $V = B \cup \{\top\}$, where each vertex is either a fact in B or an assertion of truth under no condition. $(b, h) \in E$ if b is involved in the proof of h by some rule. $(\top, h) \in E$ if h can be inferred by some rule with empty body. The extraction for essence is given by Algorithm 1.

If the dependency graph is a DAG, then essential predicates are represented by the vertices with zero in-degree. However, if cycles appear in $\mathcal{G}$, then at least one vertex in each cycle should be included in N. This assertion is proved bellow:

Lemma 12. *If some cycle in $\mathcal{G}$ is not overlapping with other cycles, then at least one vertex should be included in N.*

Proof. A vertex in the dependency graph is guaranteed provable if it is in N or all of its in-neighbours are guaranteed provable. In the following proof, we assume that all other parts in $\mathcal{G}$ are guaranteed provable except the cycles. If none of vertices in a single cycle (not overlapping with other cycles) is included in N, then for each of these vertices, there is one in-neighbour not guaranteed provable. Thus, none of vertices in the cycle is guaranteed provable. At least one vertex should be selected in N. $\square$

Lemma 13. *If some cycles in $\mathcal{G}$ are overlapping, then at least one vertex should be included in N.*

Proof. Suppose two cycles are overlapping in $\mathcal{G}$. If none of vertices in these cycles is in N, then none of them are guaranteed provable. If one of the vertices in the non-overlapping part is in N, then from this vertex to the one before intersection, all of these vertices are guaranteed provable. The other cycle is remained equivalent to circumstances of non-overlapping cycle and at least one of these vertices should be in N. If one of the vertices in the overlapping part is in N, then both cycles are guaranteed provable. In this case, still, at least one vertex is selected in each cycle. Cases are similar for more than two over lapping cycles. $\square$

Lemma 14. *If there are cycles in the dependency graph, then at least one vertex should be included in N.*

Proof. It is clear by Lemma 12 and 13. $\square$

In the framework, two components may be implemented in different strategies according to specific domains: *findSingleRule* and *CoverCycle*. To implement *findSingleRule*, pruning techniques are needed as the search space is large and useful candidates are sparse in the space. Given that semantic correlations may be strong in domain specific KBs, cycles are predicted to be large and frequent. Therefore, efficient coverage procedure is also required in the framework.

V. Conclusion

In this paper, we introduced a framework for extracting essence from factual knowledge. Theoretical proofs are also given for key properties of the framework. To put it into practice, more concrete work is required to design and analyze in *findSingleRule* and *CoverCycle*.

References

[1] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek, "Amie: association rule mining under incomplete evidence in ontological knowledge bases," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 413–422.

[2] C. Moran and C. Lee, "What's normal? influencing women's perceptions of normal genitalia: an experiment involving exposure to modified and nonmodified images," *BJOG: An International Journal of Obstetrics & Gynaecology*, vol. 121, no. 6, pp. 761–766, 2014.