KNOWLEDGE REASONING BASED ON **LOGIC RULES**

Presented by Jinge Wu
23/06/2022

1. Knowledge reasoning based on **First-order predicate logic rules**
2. Knowledge reasoning based on **Rule**
3. Knowledge reasoning based on **Ontology**
4. Knowledge reasoning based on **Random walk algorithm**

1. Knowledge reasoning based on **First-order predicate logic rules**
2. Knowledge reasoning based on **Rule**
3. Knowledge reasoning based on **Ontology**
4. Knowledge reasoning based on **Random walk algorithm**

The process of reasoning using first-order predicate logic is

$$(\mathit{YaoMing},\ \mathit{wasBornIn},\ \mathit{Shanghai}) \wedge (\mathit{Shanghai},\ \mathit{locatedIn},\ \mathit{China}) \Rightarrow (\mathit{YaoMing},\ \mathit{nationality},\ \mathit{China})$$

Predicate: used to describe the nature and things of the individual

IsFruit(x); IsFather(x,y); IsBetween(a,b,c)

Literal: It can be defined as any predicate or negated predicate applied to any terms.

Clause – It can be defined as any disjunction of literals whose variables are universally quantified.

literal

clause

$$(A \vee \neg B \vee C) \wedge$$
$$(D \vee B \vee E) \wedge$$

First-**O**rder **I**nductive **L**earner (FOIL)

```
FOIL(Target predicate, predicates, examples)
• Pos ← positive examples
• Neg ← negative examples
• Learned rules ← {}
• while Pos, do
    //Learn a NewRule
    – NewRule ← the rule that predicts target-predicate with no preconditions

    – NewRuleNeg ← Neg
    – while NewRuleNeg, do
        Add a new literal to specialize NewRule
        1. Candidate_literals ← generate candidates for newRule based on Predicates
        2. Best_literal ←
                 argmax_{L∈Candidate literals} Foil_Gain(L,NewRule)
        3. add Best_literal to NewRule preconditions
        4. NewRuleNeg ← subset of NewRuleNeg that satisfies NewRule preconditions
    – Learned rules ← Learned rules + NewRule
    – Pos ← Pos – {members of Pos covered by NewRule}
• Return Learned rules
```

First-**O**rder **I**nductive **L**earner (FOIL)

Say we are tying to predict the **Target-predicate-** *GrandDaughter(x,y)*.

We perform the following steps:

Step 1 - NewRule = GrandDaughter(x,y)

Step 2 –

2.a - Generate the candidate_literals.

(Female(x), Female(y), Father(x,y), Father(y.x), Father(x,z), Father(z,x), Father(y,z), Father(z,y))

2.b - Generate the respective candidate literal negations.

(¬Female(x), ¬Female(y), ¬Father(x,y), ¬Father(y.x), ¬Father(x,z), ¬Father(z,x), ¬Father(y,z), ¬Father(z,y))

Step 3 - FOIL might greedily select Father(x,y) as most promising, then

NewRule = GrandDaughter(x,y) ← Father(y,z) [Greedy approach]

Step 4 - Foil now considers all the literals from the previous step as well as:

(Female(z), Father(z,w), Father(w,z), etc.) and their negations.

Step 5 - Foil might select Father(z,x), and on the next step Female(y) leading to

NewRule = GrandDaughter (x,y) ← Father(y,z) ∧ Father(z,x) ∧ Female(y)

Step 6 - If this greedy approach covers only positive examples it terminates the search for further better results.

FOIL now **removes all positive examples covered by this new rule.**

If more are left then the outer while loop continues.

Different versions of FOIL

- nFOIL and tFOIL (Landwehr, Kersting, & Raedt, 2007)
  integrate naïve Bayes and tree augmented naïve Bayes
- kFOIL (Landwehr, Passerini, De Raedt, & Frasconi, 2010)
  combines FOIL's rule learning algorithm and kernel
- AMIE (Galárraga, Teflioudi, Hose, and Suchanek, 2013)
  mine Horn rules on a knowledge graph for complementing knowledge graphs and detecting errors
- AMIE+ (Galárraga, Teflioudi, Hose, and Suchanek, 2015)
  mine larger KBs by considering type information and using joint reasoning
- RDF2Rules (Wang and Li, 2015)
  mine Frequent Predicate Cycles (FPCs) to parallelize this process

1. Knowledge reasoning based on **First-order predicate logic rules**
2. Knowledge reasoning based on **Rule**
3. Knowledge reasoning based on **Ontology**
4. Knowledge reasoning based on **Random walk algorithm**

NELLs: Never-Ending Language Learning system- Carnegie Mellon University

Continuous operation since January 2010

Inputs:

- Initial ontology
- Handful of examples of each predicate in ontology
- The web
- Occasional interaction with human trainers

The task:

- Run 24 x 7, forever
- Each day:
  1. Extract more facts from the web to populate the initial ontology
  2. Learn to read (perform #1) better than yesterday
  3. Algorithm: Semi-supervised Bootstrap Learning with EM algorithm

Reasoning over KG by applying simples rules or statistical features

- **Spass-YAGO**: expands the knowledge graph by abstracting the triples into equivalent rule classes
- **SDType and SDValidate**: exploit statistical distributions of properties and types for type completion and error detection.
  - ➢ **SDType**: uses the statistical distribution of types in the head entity and tail entity position of the property for predicting the entities' types.
  - ➢ **SDValidate**: computes the relative predicate frequency (RPF) for each statement, with a low RPF value meaning incorrect.
- **Programming with Personalized PageRank (ProPPR):** based on a personalized PageRank process over the proof constructed by SLD resolution theorem-prover.
- **TensorLog**: where inference uses a differentiable process.
- **Neural logic programming**: in which the structure and parameter learning of logical rules are combined in an end-to-end differentiable model.

Combine logic rules with probability graph models and logical network

- **Markov logic-based system** for cleaning NELL: allows knowledge bases to make use of joint probabilistic reasoning, or, applies **Markov logic network (MLN)** to a web-scale problem.

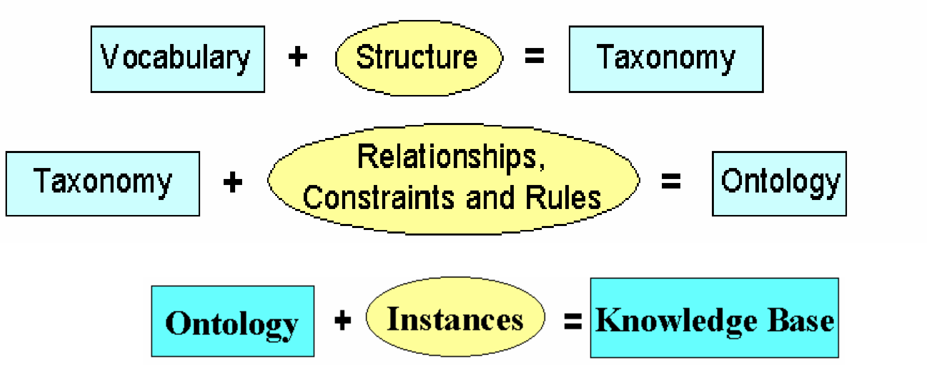
- **Probabilistic knowledge base (ProbKB):** allows an efficient SQL-based inference algorithm for knowledge completion that applies MLN inference rules in batches.

- **Hinge-Loss Markov Random Fields (HL-MRFs):** capture relaxed, probabilistic inference with Boolean logic and exact, probabilistic inference with fuzzy logic, making them useful models for both discrete and continuous data.

1. Knowledge reasoning based on **First-order predicate logic rules**
2. Knowledge reasoning based on **Rule**
3. Knowledge reasoning based on **Ontology**
4. Knowledge reasoning based on **Random walk algorithm**

Ontologies

- Ontologies are about vocabularies and their meanings, with explicit, expressive, and well-defined semantics, possibly machine-interpretable.
- *"Ontology is a formal specification of a conceptualization." Gruber, 1993*
- Main elements of an ontology:
  - Concepts
  - Relationships
    - Hierarchical
    - Logical
  - Properties
  - Instances (individuals)



Why develop ontologies?

- To share knowledge
  - E.g., using an ontology for integrating terminologies
- To reuse domain knowledge
  - E.g., geography ontology
- To make domain assumptions explicit
  - Facilitate knowledge management
  - Enable new users to learn about the domain
- To distinguish domain knowledge from operational knowledge
  - e.g., biblio metadata
- **Reasoning about knowledge from ontology language**

Reasoning about knowledge in ontology language

Class membership

- If X is an instance of class C, and C is a subclass of D, then we can infer that x is an instance of D

Equivalence of classes

- If class A is equivalent to class B, and class B is equivalent to class C, then A is equivalent to C, too

Consistency

- X instance of classes A and B, but A and B are disjoint
- This is an indication of an error in the ontology

Classification

- Certain property-value pairs are a sufficient condition for membership in a class A; if an individual x satisfies such conditions, we can conclude that x must be an instance of A

Ontology Languages

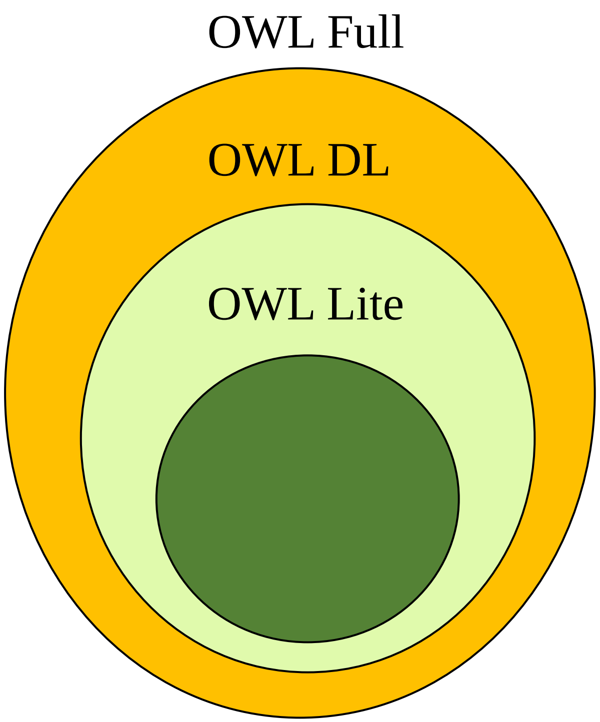
- Graphical notations
  - Semantic networks
  - Topic maps
  - UML
  - **RDF (Resource Description Framework)**
- Logic based
  - Description Logics (e.g., OIL, DAML+OIL, **OWL**)
  - Rules (e.g., RuleML, LP/Prolog)
  - First Order Logic

An Example Web Ontology Language (OWL)

```
<owl:Class rdf:ID=“Person” />
<owl:Class rdf:ID=“Man”>
        <rdfs:subClassOf rdf:resource=“#Person” />
        <owl:disjointWith rdf:resource=“#Woman” />
</owl:Class>
<owl:Class rdf:ID=“Woman”>
    <rdfs:subClassOf rdf:resource=“#Person” />
    <owl:disjointWith rdf:resource=“#Man” />
</owl:Class>
<owl:Class rdf:ID=“Father”>
    <rdfs:subClassOf rdf:resource=“Man” />
    <owl:Restriction owl:minCardinality="1">
                <owl:onProperty rdf:resource="#hasChild" />
    </owl:Restriction>
</owl:Class>
<owl:ObjectProperty rdf:ID=“hasChild">
                <rdfs:domain rdf:resource="#Parent" />
                <rdfs:range rdf:resource="#Person" />
</owl:ObjectProperty>
```

Versions of OWL

- Depending on the intended usage, OWL provides three increasingly expressive sublanguages



Full: Very expressive,
no computation guarantees

DL (Description Logic): Maximum
expressiveness, computationally complete

Lite: Simple classification hierarchy
with simple constraints.

OWL and KG

- **2004. F-OWL**: Frame-based system to reason with OWL ontologies
- **2006. Minerva**: large-scale OWL ontologies; combines a DL reasoner and a rule engine for ontology inference to improve efficiency
- **2007. OWL-DL Reasoner Pallet**: Support incremental reasoning against dynamic KGs
- **2016. Ontological Pathfinding (OP)**: generalizes to web-scale KBs through optimizations and parallelization techs
- **2016. KGRL based on OWL2 RL inference rules**: more powerful reasoning ability

1. Knowledge reasoning based on **First-order predicate logic rules**
2. Knowledge reasoning based on **Rule**
3. Knowledge reasoning based on **Ontology**
4. Knowledge reasoning based on **Random walk algorithm**

PRA: Path ranking algorithm

Key idea:
Use paths that connect two entities as features to predict potential relations between them

Typical workflow

- Feature extraction: generate and select path features
- Feature computation: Given a node pair (h, t) and a path π, compute the feature value as a random walk probability $P(t|h, \pi)$

$$p(t|h,\pi) = \sum_{e' \in range(\pi')} p(h, e'; \pi') P(t|e'; r_l)$$

- Relation-specific classification: train an individual classifier for each relation

PRA: Path ranking algorithm

Advantages:

- High accuracy
- Improves the computational efficiency
- Good interpretability

Disadvantages:

- Ignore meaningful associations among different relations
- Do not have enough training data for less frequent relations
- Random walk inference is the feature sparsity

CPRA: Coupled Path ranking algorithm

A multi-task learning framework for PRA

Motivation

It will be beneficial for PRA to model certain relations collectively, particularly when the relations are highly related

Key steps

- Relation clustering: which relations should be coupled
- Relation coupling: in what manner they should be coupled

Advantages

- Take into account meaningful relation associations
- Enable implicit data sharing among different relations

Conclusion

Trend: mine rules or features automatically for training models with machine learning methods

Logic rules: represents the knowledge graph as a complex heterogeneous network, so the reasoning tasks can be completed by the transfer probability, shortest path, and breadth-first search algorithms.

Limitations

- High computational complexity
- Long-tailed distribution in nodes, which means a few entities and relations have a higher frequency of occurrence
- Sparsity seriously affects the inference performance
- How to handle multi-hop reasoning problem

Therefore, scholars mainly focus on the reasoning methods based on **distributed representation**, which is not sensitive to data sparsity and is more expandable.