Course: Common Sense Reasoning

10. Learning Common Sense Knowledge

Martin Molina

Common sense knowledge may be learned automatically from large scale data bases

- Large scale data bases (e.g., Yago, Nell) have large amounts of instances corresponding to multiple domains
- This information may be processed by machine learning algorithms to learn general knowledge (e.g., relations between classes)
- Two representative cases are presented:
  - Amie
  - PRA

Amie is a method for learning rules on large RDF knowledge bases

- The method Amie has been able to learn rules efficiently from the large scale data base Yago
- The precision of the learned model is between 60% and 70%

Luis Galárraga

[Galárraga, 2013]
[Galárraga et al., 2014]
[Galárraga et al., 2015]

Previous machine learning methods for learning rules are not appropriate

- Methods based on a single relation (e.g., PRISM or RIPPER) cannot be used because large RDF knowledge bases have multiple relations
- Some relational approaches (e.g., ILP systems) are are too slow to handle the huge amount of data of large RDF knowledge bases

Examples of learned rules

$$isCitizenOf(x, y) \Rightarrow livesIn(x, y)$$
$$hasAdvisor(x, y) \wedge graduatedFrom(x, z) \Rightarrow worksAt(y, z)$$
$$wasBornIn(x, y) \wedge isLocatedIn(y, z) \Rightarrow isCitizenOf(x, z)$$
$$hasWonPrize(x, G.\ W.\ Leibniz) \Rightarrow livesIn(x, Germany)$$

What is the utility of this type of rules?

- Derive new facts (e.g., to extend the content of the knowledge base)
- Check the consistency of the knowledge base
- Help to better understand the data
- Reasoning (e.g., diagnosis)

The input data of the Amie method is a set of facts from the knowledge base

$$r_1(x_1, y_1),\ r_2(x_2, y_2),\ ...,\ r_n(x_n, y_n)$$

- Triples <relation r_i , subject x_j , object y_k >
- Instance data (no classes)
- Only positive statements (no negations)
- Example:

$$livesIn(Adam, Paris)$$
$$livesIn(Adam, Rome)$$
$$livesIn(Bob, Zurich)$$
$$wasBornIn(Adam, Paris)$$
$$wasBornIn(Carl, Rome)$$

Amie assumes partial completeness

- Partial completeness assumption (PCA):
  - If we know one object *y* for a given subject *x* and relation *r*, then we know all objects *y* for that subject *x* and relation *r*.
- This allow us to generate counter-examples in a way that is less restrictive than CWA.

Example of partial completness assumption

nationality(John, USA)
nationality(Mary, France)

OWA (Open World Assumption):

nationality(John, USA) = True
nationality(Mary, USA) = Unknown
nationality(Peter, USA) = Unknown

CWA (Closed World Assumption):

nationality(John, USA) = True
nationality(Mary, USA) = False
nationality(Peter, USA) = False

PCA (Partial Completeness Assumption):

nationality(John, USA) = True
nationality(Mary, USA) = False
nationality(Peter, USA) = Unknown

Formal notation used to describe Amie

A rule $B_1 \wedge B_2 \wedge ... \wedge B_n \Rightarrow r(x, y)$

is represented as $\overrightarrow{B} \Rightarrow r(x, y)$

The notation $\#x \ : \ X$

represents $|\{x \ : \ X \ \in \ \mathcal{K}\}|$

where $\mathcal{K}$ is the input knowledge base

Definition: Confidence and support of a rule

$$conf_{pca}(\overrightarrow{B} \Rightarrow r(x,y)) := \frac{supp(\overrightarrow{B} \Rightarrow r(x,y))}{\#(x,y) : \exists z_1, ..., z_m, y' : \overrightarrow{B} \wedge r(x,y')}$$

$$supp(\overrightarrow{B} \Rightarrow r(x,y)) := \#(x,y) : \exists z_1, ..., z_m : \overrightarrow{B} \wedge r(x,y)$$

Example of support

livesIn(*Adam*, *Paris*)
livesIn(*Adam*, *Rome*)
livesIn(*Bob*, *Zurich*)
wasBornIn(*Adam*, *Paris*)
wasBornIn(*Carl*, *Rome*)

R: *livesIn*(*x*, *y*) → *wasBornIn*(*x*, *y*)

$$supp(\overrightarrow{B} \Rightarrow r(x,y)) := \#(x,y) : \exists z_1, ..., z_m : \overrightarrow{B} \wedge r(x,y)$$

livesIn(*x*, *y*) ∧ *wasBornIn*(*x*, *y*)

livesIn(*Adam*, *Paris*) ∧ *wasBornIn*(*Adam*, *Paris*)

supp(*R*) = 1

Example of confidence

livesIn(*Adam*, *Paris*)
livesIn(*Adam*, *Rome*)
livesIn(*Bob*, *Zurich*)
wasBornIn(*Adam*, *Paris*)
wasBornIn(*Carl*, *Rome*)

R: *livesIn*(*x*, *y*) → *wasBornIn*(*x*, *y*)

$$conf_{pca}(\overrightarrow{B} \Rightarrow r(x,y)) := \frac{supp(\overrightarrow{B} \Rightarrow r(x,y))}{\#(x,y) : \exists z_1, ..., z_m, y' : \overrightarrow{B} \wedge r(x,y')}$$

(Adam, Paris)		*livesIn*(*Adam*, *Paris*) ∧ *wasBornIn*(*Adam*, *Paris*)
(Adam, Rome)	*y'=Paris*	*livesIn*(*Adam*, *Paris*) ∧ *wasBornIn*(*Adam*, *Paris*)
(Bob, Zurich)		~~*livesIn(Bob, ...) ∧ wasBornIn(Bob, ...)*~~
(Carl, Rome)		~~*livesIn(Carl, ...) ∧ wasBornIn(Carl, ...)*~~

Confidence(*R*) = 1/2

Definition: Head coverage of a rule

Quantifies the ratio of the known true facts that are implied by the rule

$$hc(\overrightarrow{B} \Rightarrow r(x,y)) := \frac{supp(\overrightarrow{B} \Rightarrow r(x,y))}{size(r)}$$

$$size(r) := \#(x',y') : r(x',y')$$

Example of head coverage

livesIn(*Adam*, *Paris*)
livesIn(*Adam*, *Rome*)
livesIn(*Bob*, *Zurich*)
wasBornIn(*Adam*, *Paris*)
wasBornIn(*Carl*, *Rome*)

R: *livesIn*(*x*, *y*) → *wasBornIn*(*x*, *y*)

$$size(r) := \#(x', y') : r(x', y')$$

Supp(*R*) = 1	*livesIn*(*Adam*, *Paris*) ∧ *wasBornIn*(*Adam*, *Paris*)
Size(*wasBornIn*) = 2	*wasBornIn*(*Adam*, *Paris*), *wasBornIn*(*Carl*, *Rome*)
hc(*R*) = 1/2	*Supp*(*R*) / *Size*(*wasBornIn*)

Amie algorithm

Take a rule from a queue (initially, all atoms):

1. Check end condition (maximum length, ...).
2. Expand the rule to produce new rules (using mining operators).
3. Prune (metrics, duplicates, etc.).
4. Put the new rule in the queue.

Search space is restricted to closed Horn rules (no free variables).

There are several operators to expand a rule

1. Add Dangling Atom (OD)

   This operator adds a new atom to a rule. The new atom uses a fresh variable for one of its two arguments. The other argument is a variable that is shared with the rule, i.e., it occurs in some other atom of the rule.

2. Add Instantiated Atom (OI)

   This operator adds a new atom to a rule that uses an entity for one argument and shares the other argument (variable) with the rule.

3. Add Closing Atom (OC)

   This operator adds a new atom to a rule so that both of its arguments are shared with the rule.

Example

1. Initially:

$$livesIn(y, z)$$

2. Expand with operator OD (Add dangling atom):

$$isMarriedTo(x, y) \rightarrow livesIn(y, z)$$

3. Expand with operator OC (Add closing atom):

$$livesIn(x, z) \wedge isMarriedTo(x, y) \rightarrow livesIn(y, z)$$

Amie uses several hyper-parameters to decide whether a rule is valid

- Minimum head coverage = 0.01
- Minimum confidence = 0.1
- Maximum length = 3

Amie learn rules with 2 or 3 atoms

Rules with 4 or more atoms are normally wrong or not useful:

$$exports(y,z) \wedge imports(y,z) \wedge livesIn(x,y) \Rightarrow citizenOf(x,y)$$
$$diedIn(x,z) \wedge locatedIn(z,y) \wedge livesIn(x,z) \Rightarrow politician(x,y)$$
$$advisor(z,w) \wedge citizenOf(w,y) \wedge livesIn(z,x) \Rightarrow deals(x,y)$$

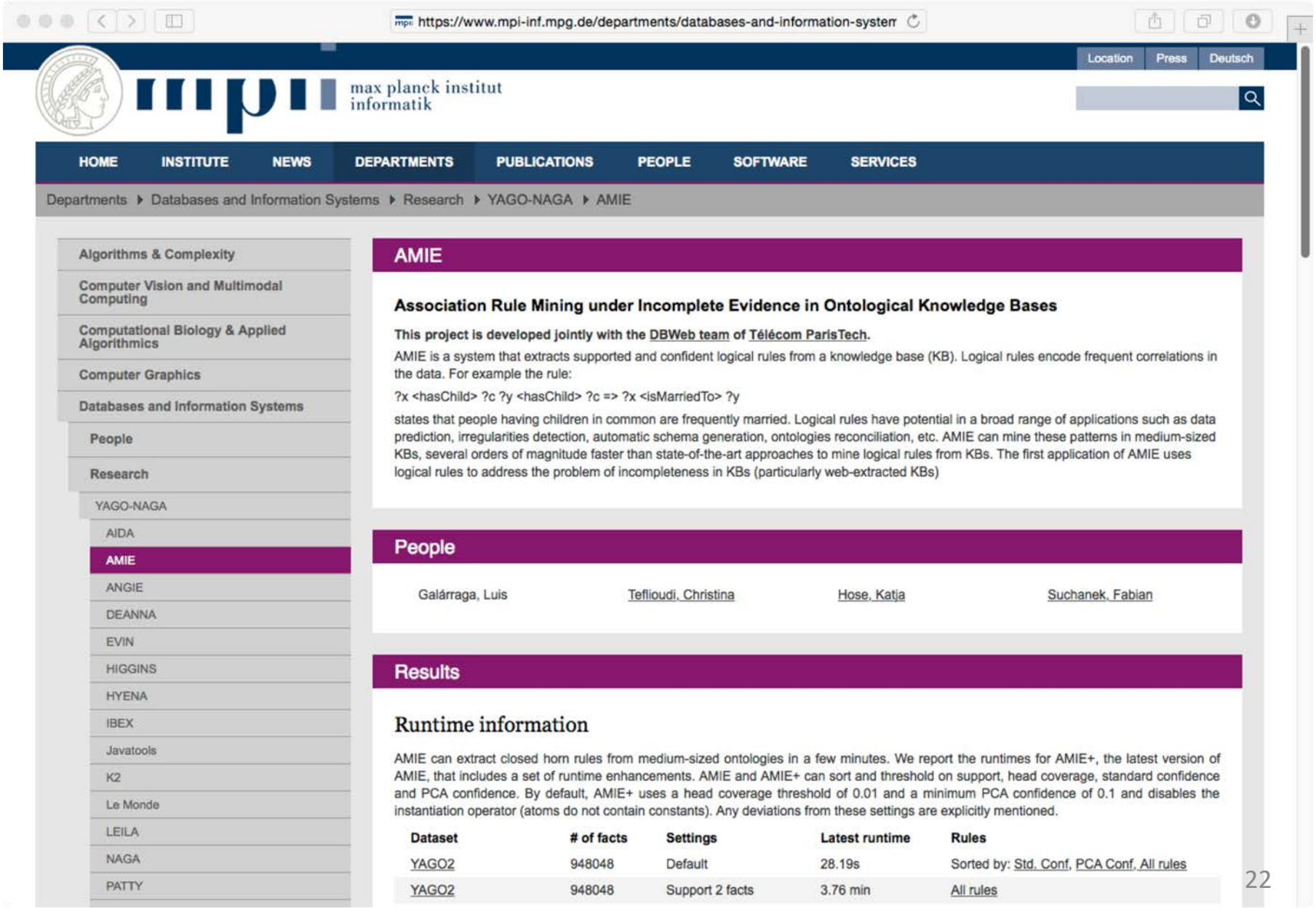
The learning speed is one of the strengths of this method

Dataset	Facts	Runtime	Rules
YAGO2	1M	3.62min	138
YAGO2 (const)	1M	17.76min	18K
Dbpedia (2 atoms)	6.7M	2.89min	6.9K

Practical solutions for efficiency:

- Language bias (search space restricted to closed Horn clauses, etc.)
- Pruning heuristics (only connected facts, etc.)
- Approximation of confidence measures instead of exact calculation
- Optimized storage implementation

Web site: https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/amie



Software implementing Amie is available for free download

Downloads

Software

Download	Date	Description
AMIE+*	2015-08-26	AMIE accepts RDF files in TSV format (like this). To run it, just write in your comand line: ***java -jar amie+.jar [TSV file]*** In case of memory issues, try to increase the virtual machine's memory resources using the arguments -XX:-UseGCOverheadLimit -Xmx[MAX_HEAP_SPACE], e.g: ***java -XX:-UseGCOverheadLimit -Xmx2G -jar amie+.jar [TSV file]*** MAX_HEAP_SPACE depends on your input size and the system's available memory. The package also contains the utilities to generate and evaluate predictions from the rules mined by AMIE. Without additional arguments AMIE+ thresholds using PCA confidence 0.1 and head coverage 0.01. You can change these default settings. Run ***java -jar amie+.jar*** (without an input file) to see a detailed description of the available options.
AMIE+ Source code and documentation	2015-08-26	Package containing AMIE's source code as well as its library dependencies and documentation.

* This program is released under the terms of the Creative Commons Attribution-NonComercial license v3.0

The website shows learning results using different knowledge bases

Results

Runtime information

AMIE can extract closed horn rules from medium-sized ontologies in a few minutes. We report the runtimes for AMIE+, the latest version of AMIE, that includes a set of runtime enhancements. AMIE and AMIE+ can sort and threshold on support, head coverage, standard confidence and PCA confidence. By default, AMIE+ uses a head coverage threshold of 0.01 and a minimum PCA confidence of 0.1 and disables the instantiation operator (atoms do not contain constants). Any deviations from these settings are explicitly mentioned.

Dataset	# of facts	Settings	Latest runtime	Rules
YAGO2	948048	Default	28.19s	Sorted by: Std. Conf, PCA Conf, All rules
YAGO2	948048	Support 2 facts	3.76 min	All rules
YAGO2 sample	46654	Support 2 facts	2.90s	Sorted by PCA conf, All rules
YAGO2	948048	Default + constants	9.93 min	Some interesting examples, All rules
YAGO2s	4122426	Default	59.38 min	Rules
DBpedia 2.0	6704524	Default	46.88 min	Rules
DBpedia 3.8	11024066	Default	7h 6 min	Rules
Wikidata (Dec 2014)	11296834	Default	25.50 min	Rules

PRA is a method for learning rules from large knowledge bases

- PRA: Path Ranking Algorithm
- PRA learns rules in the form of path types
- PRA is limited to data sources represented with triplets

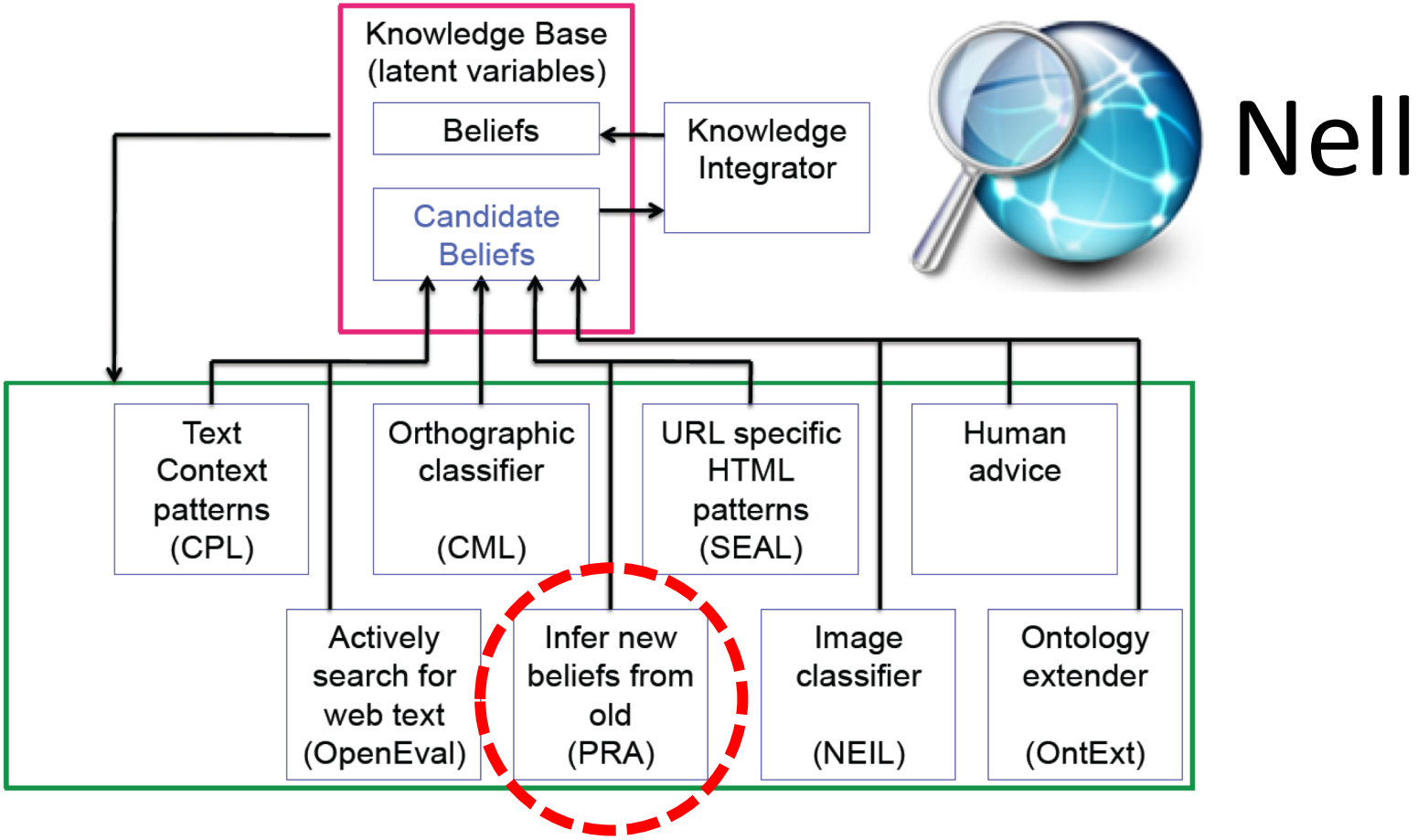
Ni Lao

William Cohen

Carnegie Mellon University

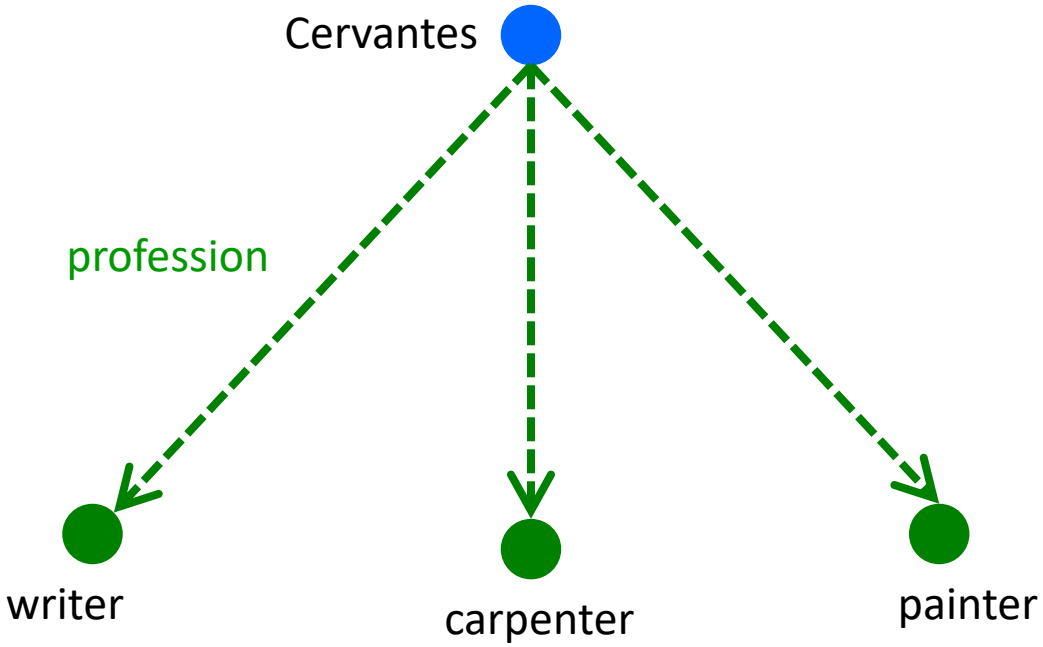
[Lao et al., 2011]
[Lao, 2012]
[Lao et al., 2015]

PRA is part of Nell



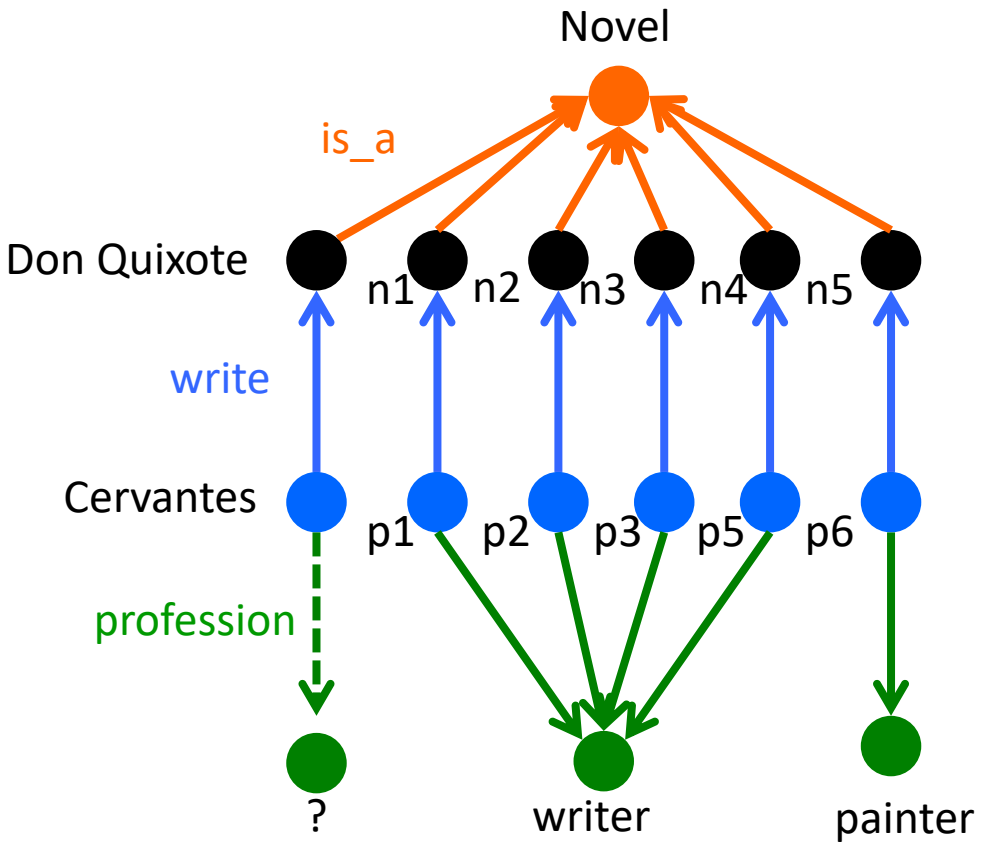
Example: what is the profession of Cervantes?

(Cervantes, profession, ?X)



Walk from «Cervantes» to «writer»

< write , is_a , is_a $^{-1}$, write $^{-1}$, profession >



Definition: path type

- Sequence of edge types <r_1, r_2, ..., r_n> corresponding to relation labels:
- Examples:

  <write, is_a, is_a^{-1}, write^{-1}, profession>

  "The professions of the persons who wrote something of the same class of what was written by the query person"

  <born_in, born_in^{-1}>

  “the persons who were born in the same town as the query person”

  <born_in, born_in^{-1}, nationality>

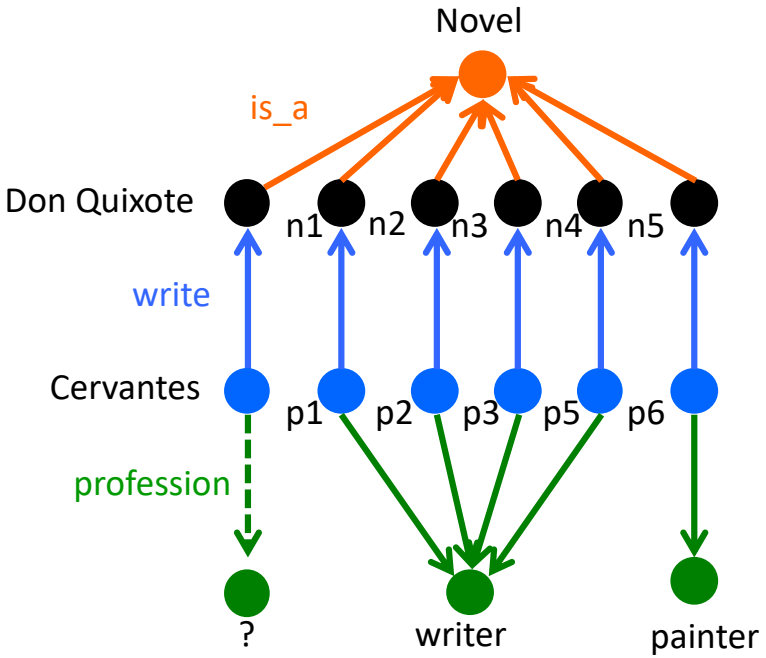
  “the nationalities of persons who were born in the same town as the query person
- IDEA: We can learn path types for candidate relations

Probability of reaching the object t from the subject s by a random walk that instantiates the path type

$$P(s \rightarrow t; \pi)$$

- We can efficiently calculate this probability using sampling techniques
- This probability can be used to predict whether the relation $r(s, t)$ is true

Example



profession(Cervantes, writer): 4/5 = 80% of paths
profession(Cervantes, painter): 1/5 = 20% of paths

P(Cervantes → writer; < write, is_a, is_a $^{-1}$, write $^{-1}$, profession >) = 0.8

P(Cervantes → painter; < write, is_a, is_a $^{-1}$, write $^{-1}$, profession >) = 0.2

How good is a path type?

Accuracy of a path (for relation r):

Probability of reaching any correct answer node following the path:

$$acc(\pi) = \frac{1}{n} \sum_i P(s_i \rightarrow G_i; \pi),$$

Hits of a path (for relation r):

Number of queries for which path leads to any correct answer:

$$hits(\pi) = \sum_i I(P(s_i \rightarrow G_i; \pi) > 0)$$

Where:

$I(\ )$ is the indicator function, and

$$P(s_i \rightarrow G_i; \pi) \equiv \sum_{z \in G_i} P(s_i \rightarrow z; \pi)$$

When is a path type selected?

A path type is selected when:

acc(π) ≥ a (e.g., a = 0.01)

hits(π) ≥ h (e.g., h = 2)

Example:

Number of paths in PRA of maximum path length ℓ, averaged over 96 tasks

	ℓ=3	ℓ=4
all paths up to length ℓ	15,376	1,906,624
+query accuracy$\geq \alpha = 0.01$	522	5016
+query support ≥ 2	136	792

RULE 11: *teamHomeStadium(x, s) ∧ stadiumLocatedInCity(s, c) → teamPlaysInCity (x, c)*

ID	PRA Path (Comment)
	athletePlaysForTeam
1	⟨*athletePlaysInLeague*, *leagueathletes*, *athletePlaysForTeam*⟩ (teams with many athletes in the athlete's league)
2	⟨*athletePlaysInLeague*, *leagueTeams*, *teamAgainstTeam*⟩ (teams that play against many teams in the athlete's league)
	athletePlaysInLeague
3	⟨*athletePlaysSport*, *athletes*, *athletePlaysInLeague*⟩ (the league that athletes of a certain sport belong to)
4	⟨*isa*, isa^{-1}, *athletePlaysInLeague*⟩ (popular leagues with all athletes)
	athletePlaysSport
5	⟨*isa*, isa^{-1}, *athletePlaysSport*⟩ (popular sports of all athletes)
6	⟨*athletePlaysInLeague*, *superpartOfOrganization*, *teamPlaysSport*⟩ (popular sports of a certain league)
	stadiumLocatedInCity
7	⟨*stadiumHomeTeam*, *teamHomeStadium*, *stadiumLocatedInCity*⟩ (city of the stadium with the same team)
8	⟨*latitudeLongitude*, *latitudeLongitudeOf*, *stadiumLocatedInCity*⟩ (city of the stadium with the same location)
	teamHomeStadium
9	⟨*teamPlaysInCity*, *cityStadiums*⟩ (stadiums located in the same city with the query team)
10	⟨*teamMember*, *athletePlaysForTeam*, *teamHomeStadium*⟩ (home stadium of teams which share athletes with the query)
	teamPlaysInCity
11	⟨*teamHomeStadium*, *stadiumLocatedInCity*⟩ (city of the team's home stadium)
12	⟨*teamHomeStadium*, *stadiumHomeTeam*, *teamPlaysInCity*⟩ (city of teams with the same home stadium as the query)
	teamPlaysInLeague
13	⟨*teamPlaysSport*, *athletes*, *athletePlaysInLeague*⟩ (the league that the query team's members belong to)
14	⟨*teamPlaysAgainstTeam*, *teamPlaysInLeague*⟩ (the league that the query team's competing team belongs to)
	teamPlaysSport
15	⟨*isa*, isa^{-1}, *teamPlaysSport*⟩ (sports played by many teams)
16	⟨*teamPlaysInLeague*, *leagueTeams*, *teamPlaysSport*⟩ (the sport played by other teams in the league)

«Knowledge Vault» was proposed for building a knowledge base using machine learning

- A Google initiative to construct a large knowledge base using relational machine learning

[Dong et al., 2014]

[Nickel et al., 2015]

Course “Common sense reasoning”.

Work citation in APA style:

> Molina, M. (2019). Common sense reasoning [Lecture slides]. OpenCourseWare, Universidad Politécnica de Madrid. Retrieved from http://ocw.upm.es/course