Rule Mining in Knowledge Bases

Luis Galárraga

Télécom ParisTech, DBWeb

September 29th, 2016

Institut
Mines-Télécom

max planck institut
informatik

Overview

Rule Mining in Knowledge Bases

citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

- Knowledge Bases
- Rule Mining
    - Challenges
    - The AMIE system
    - Experimental evaluation

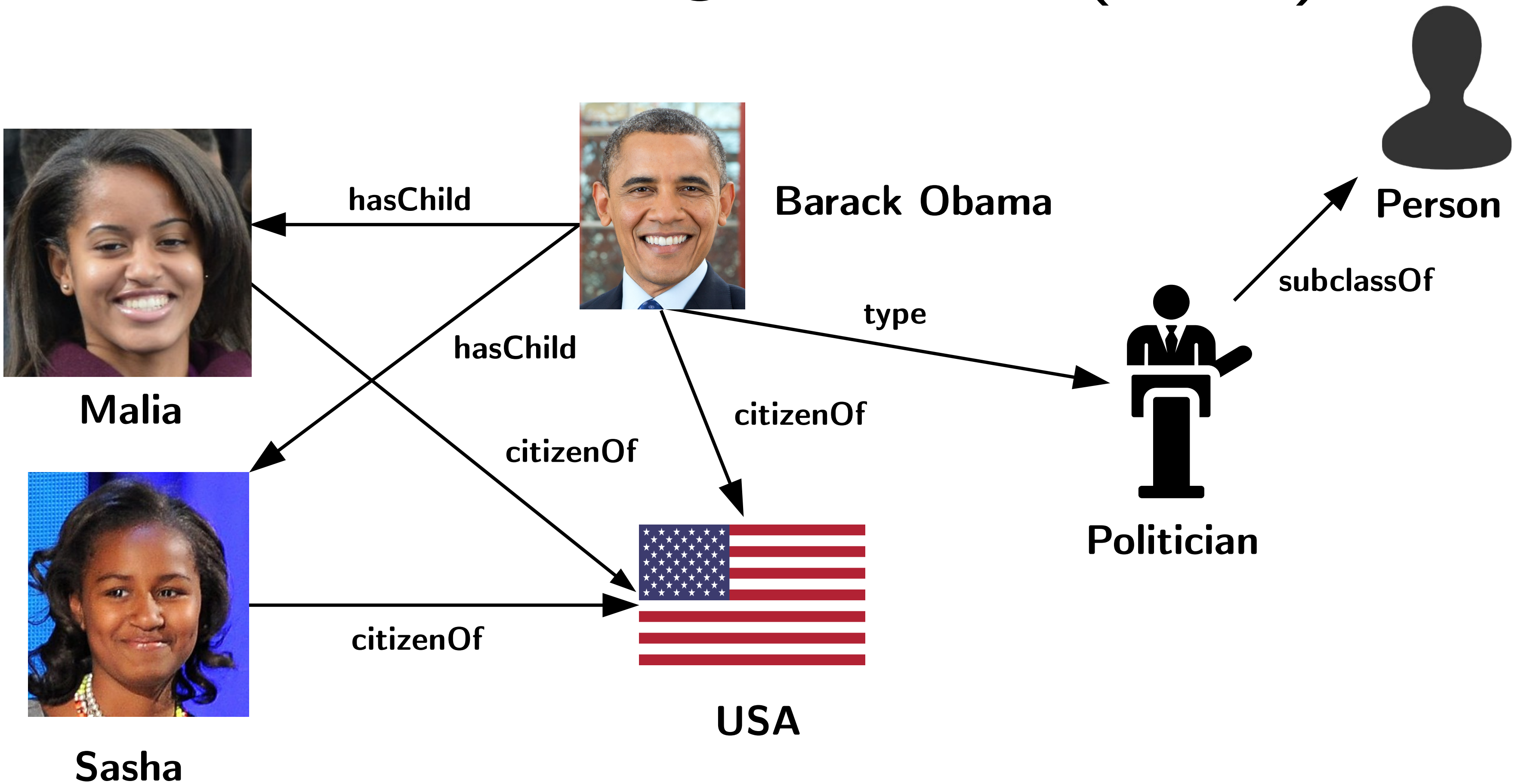
Applications

- Semantifying wikilinks
- Schema alignment
- Canonicalization of open KBs
- Prediction of completeness

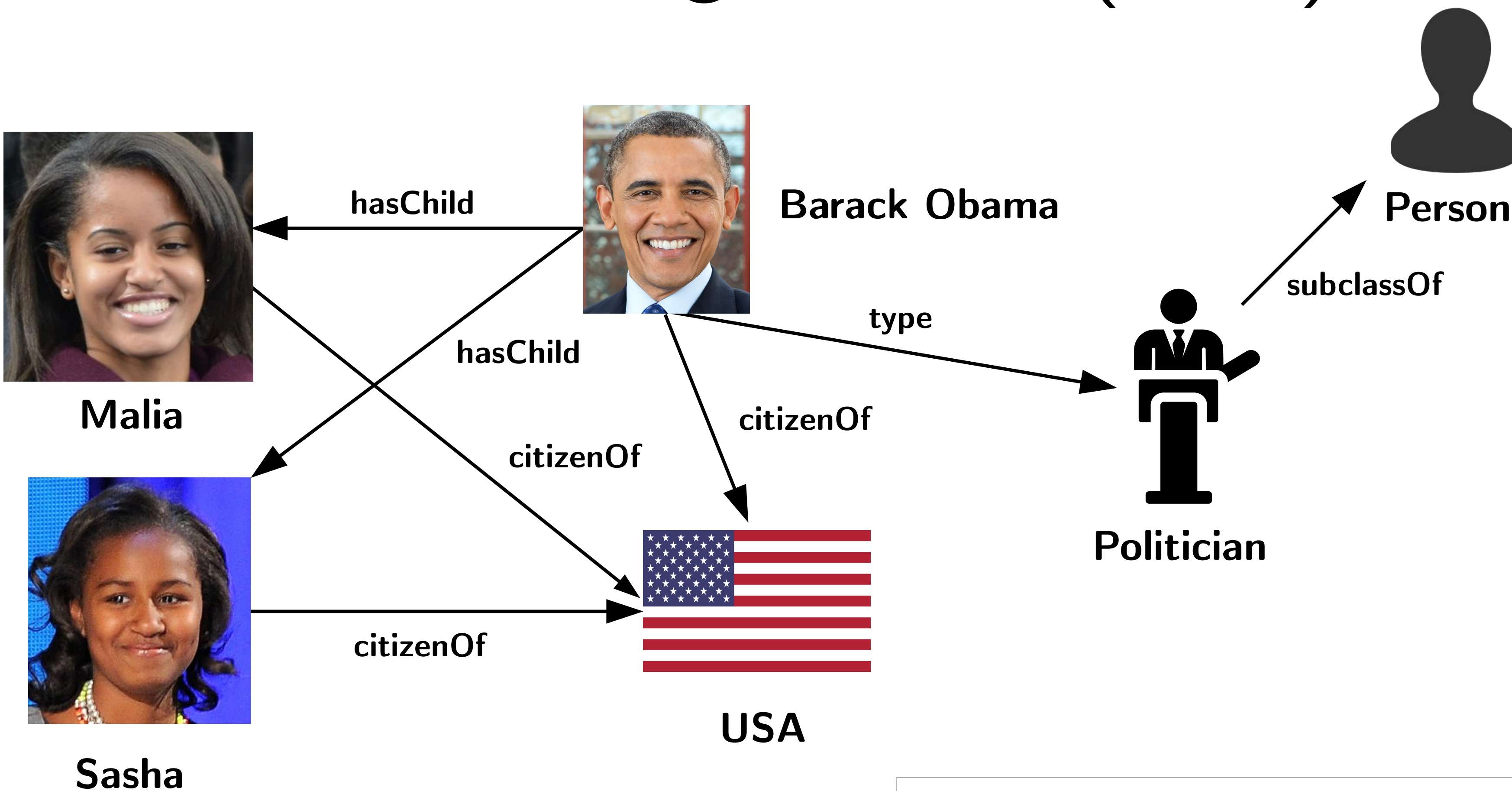
Rule Mining in Knowledge Bases

Rule Mining in **Knowledge Bases**

Knowledge Bases (KBs)

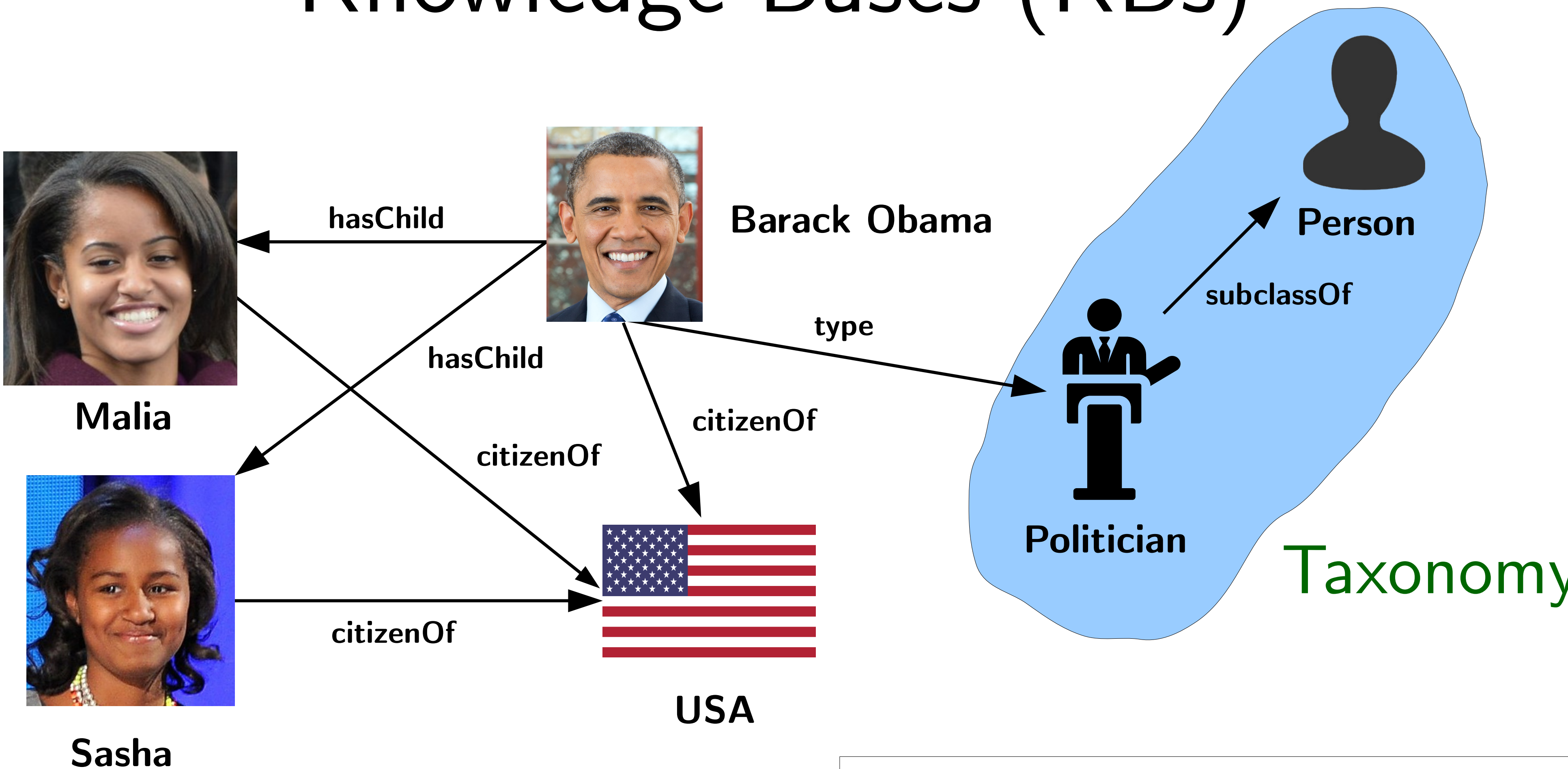


Knowledge Bases (KBs)



hasChild(Barack Obama, Malia)
hasChild(Barack Obama, Sasha)
...

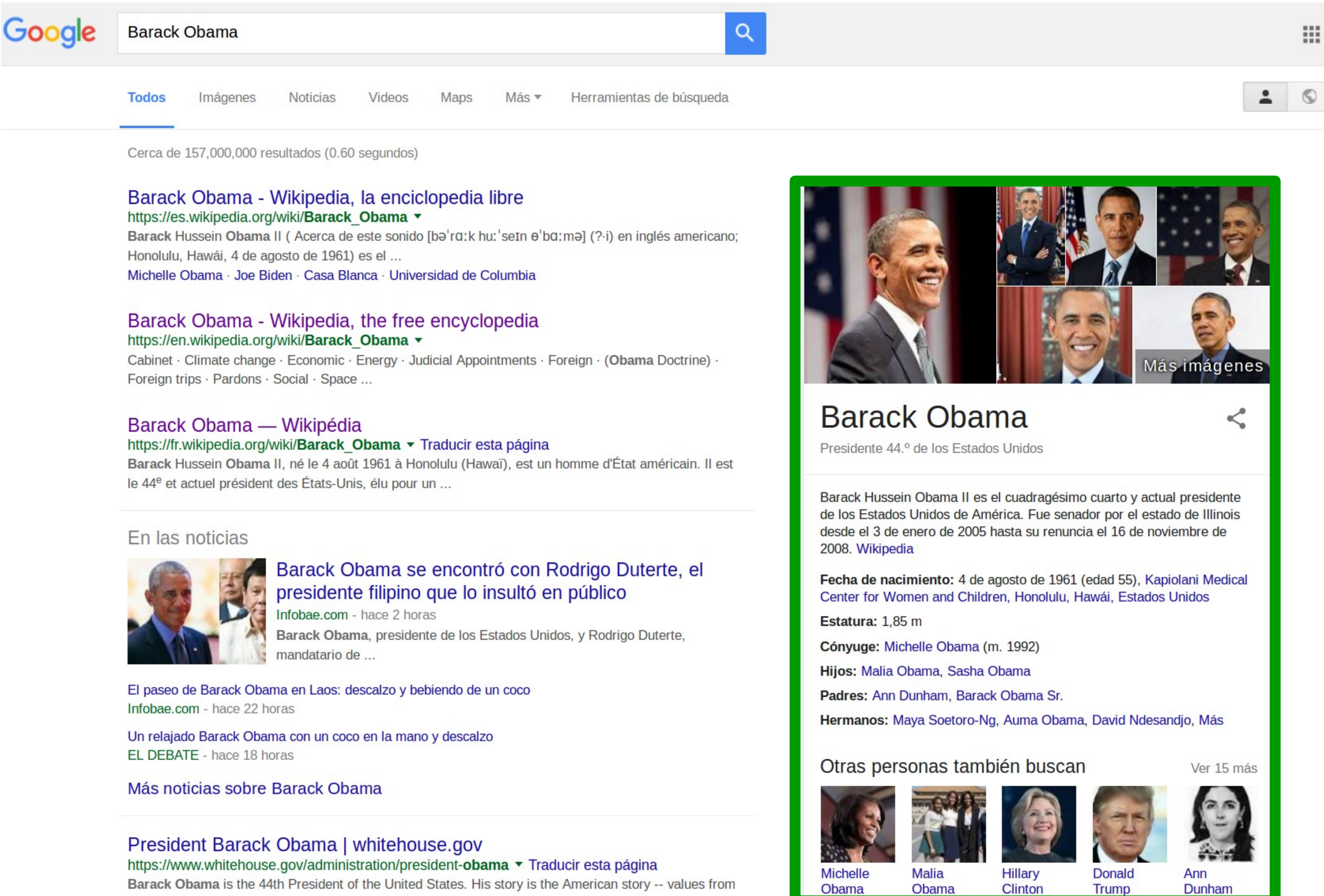
Knowledge Bases (KBs)



hasChild(Barack Obama, Malia)
hasChild(Barack Obama, Sasha)
...

KBs in action

KBs in action



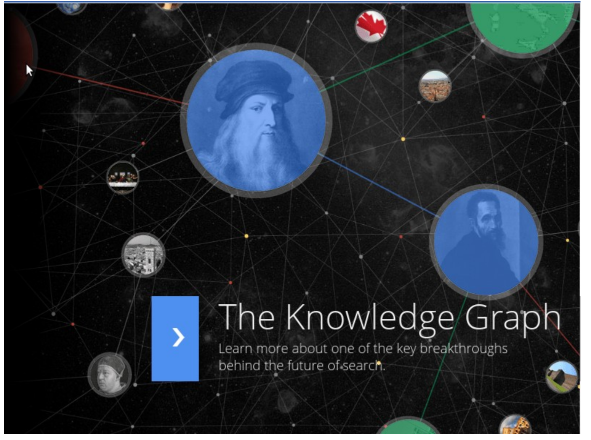
Some popular KBs

yago
select knowledge

DBpedia

Facebook Graph Search

WolframAlpha®

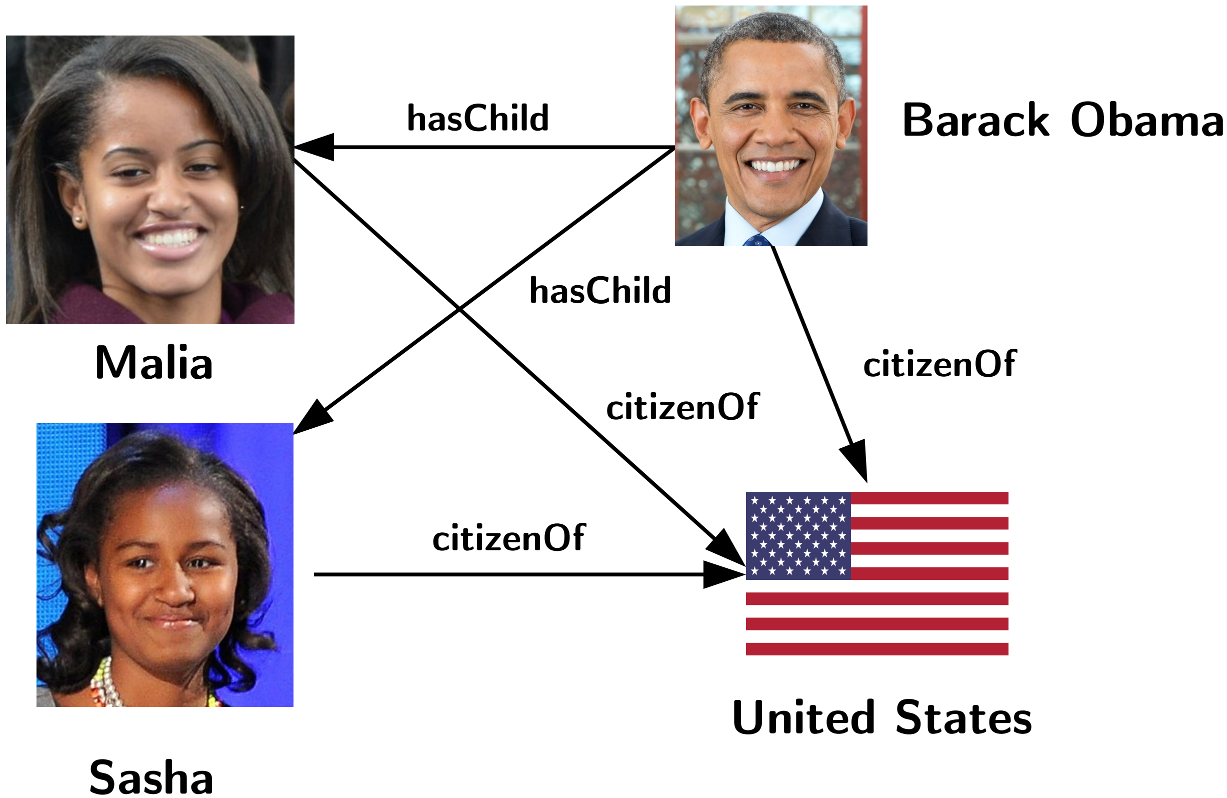


WIKIDATA

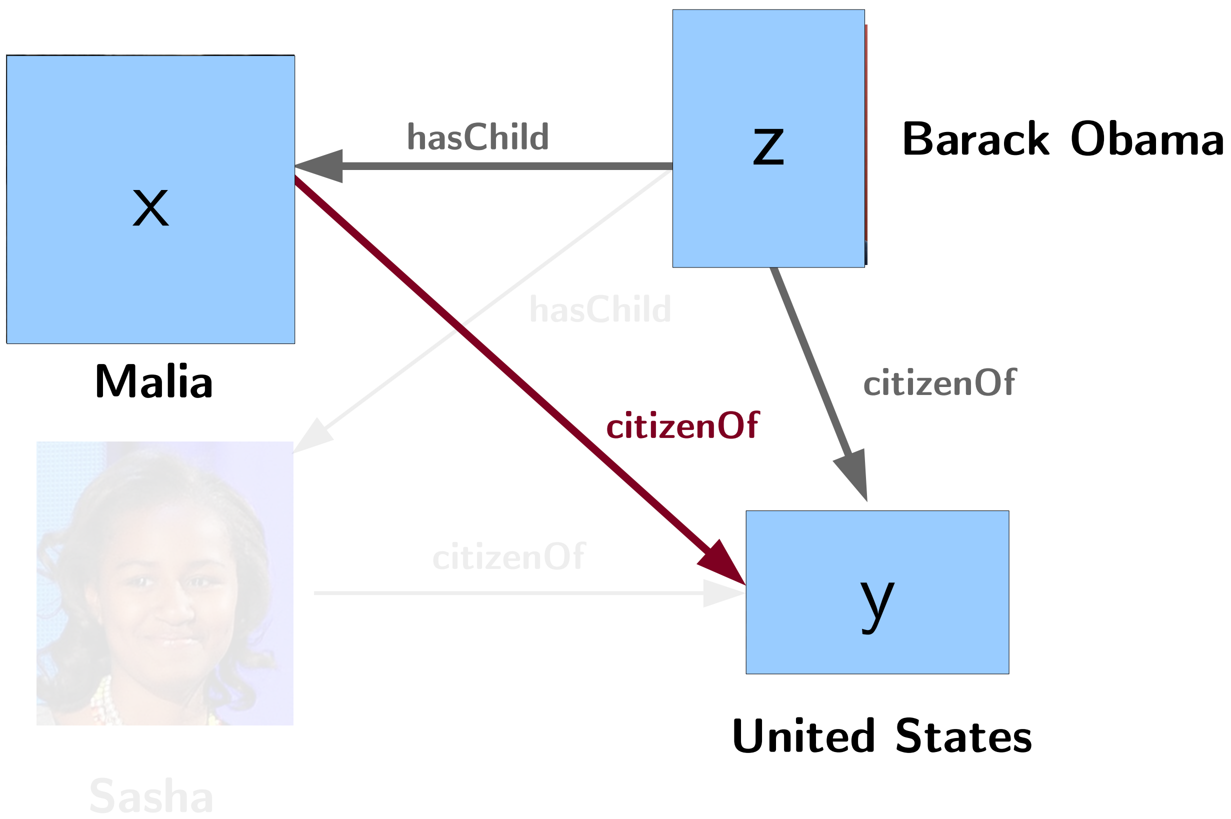
Freebase

Rule Mining in Knowledge Bases

Rule Mining in KBs

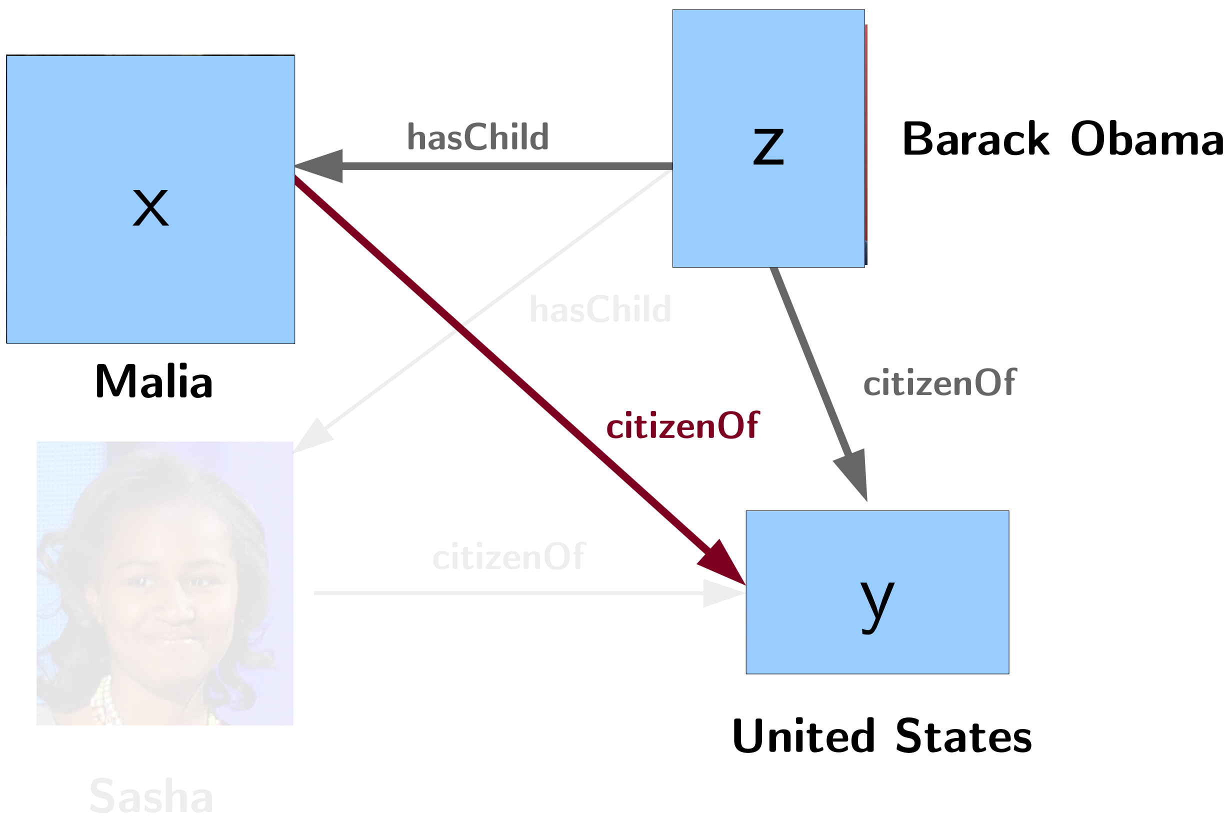


Rule Mining in KBs



citizenOf(z, y), hasChild(z, x) $\Rightarrow$ citizenOf(x, y)

Rule Mining in KBs



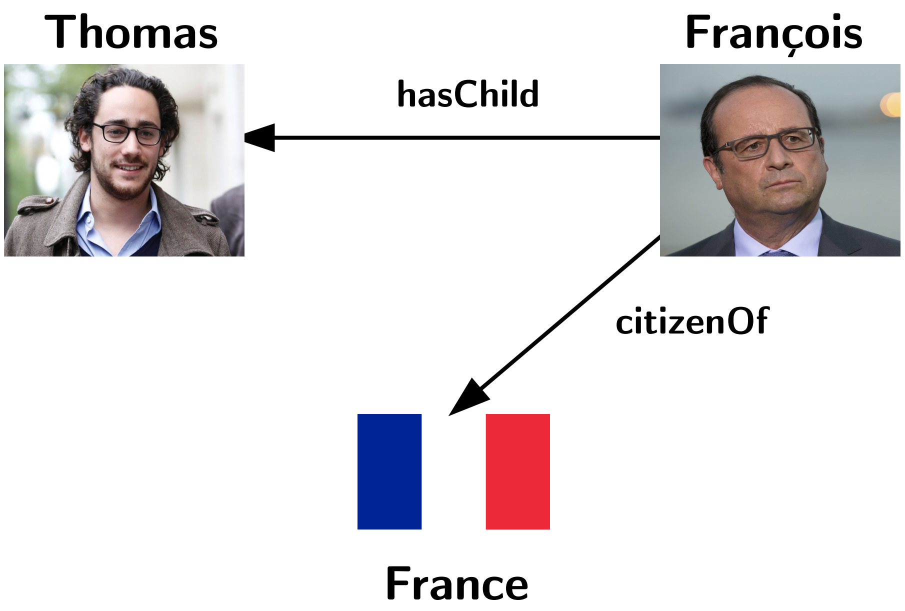
$$\underbrace{\text{citizenOf}(z, y), \text{hasChild}(z, x)}_{\textbf{Body}} \Rightarrow \underbrace{\text{citizenOf}(x, y)}_{\textbf{Head}}$$

Applications of Rule Mining

- Fact prediction

Applications of Rule Mining

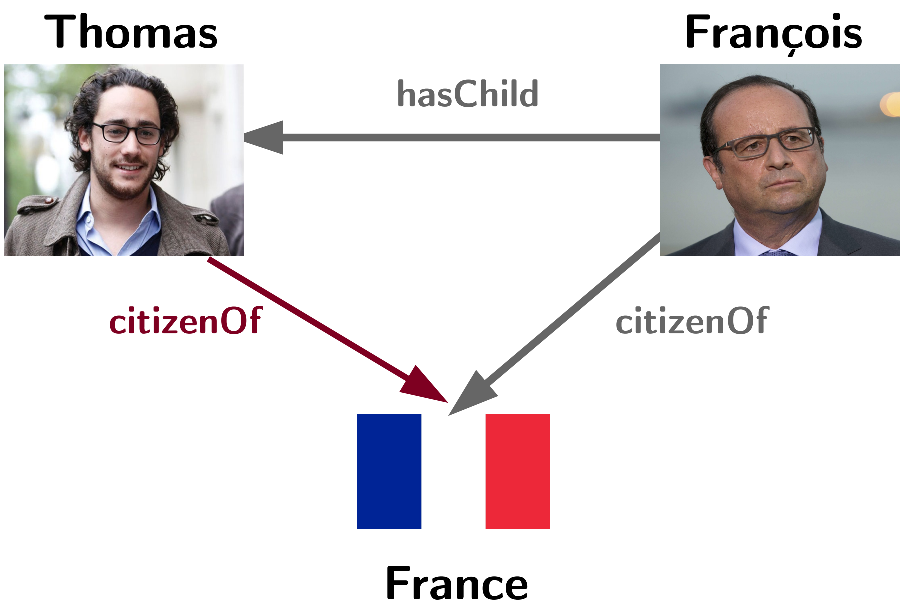
- Fact prediction



citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

Applications of Rule Mining

- Fact prediction



citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

Applications of Rule Mining

- Fact prediction
- Domain description

Applications of Rule Mining

- Fact prediction
- Domain description
    - Finding trends in KBs

Applications of Rule Mining

- Fact prediction
- Domain description
  - Finding trends in KBs

in(c, Europe), president(x, c) ⇒ male(x) [80%]

Applications of Rule Mining

- Fact prediction
- Domain description
  - Finding trends in KBs
- Data engineering and maintenance

Applications of Rule Mining

- Fact prediction
- Domain description
  - Finding trends in KBs
- Data engineering and maintenance
  - Schema mining

Applications of Rule Mining

- Fact prediction
- Domain description
  - Finding trends in KBs
- Data engineering and maintenance
  - Schema mining

marriedTo(x, y) ⇒ marriedTo(y, x)

livesIn(x, y) ⇒ type(x, Person)

livesIn(x, y) ⇒ type(y, City)

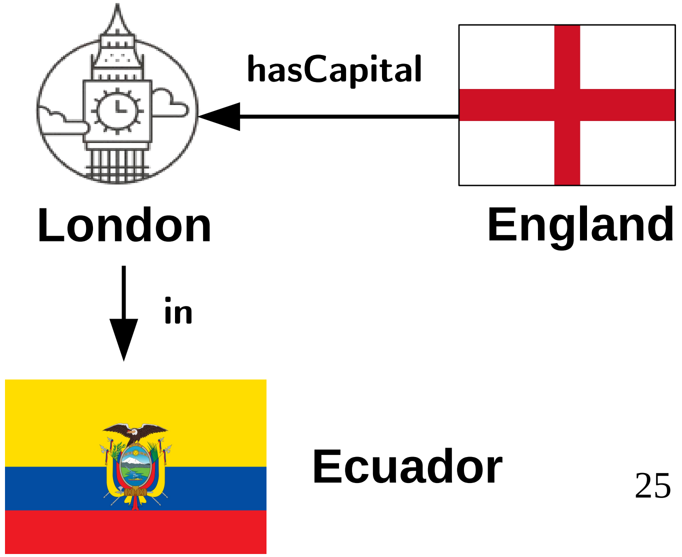
Applications of Rule Mining

- Fact prediction
- Domain description
  - Finding trends in KBs
- Data engineering and maintenance
  - Schema mining
  - Data correction

Applications of Rule Mining

- Fact prediction
- Domain description
  - Finding trends in KBs
- Data engineering and maintenance
  - Schema mining
  - Data correction

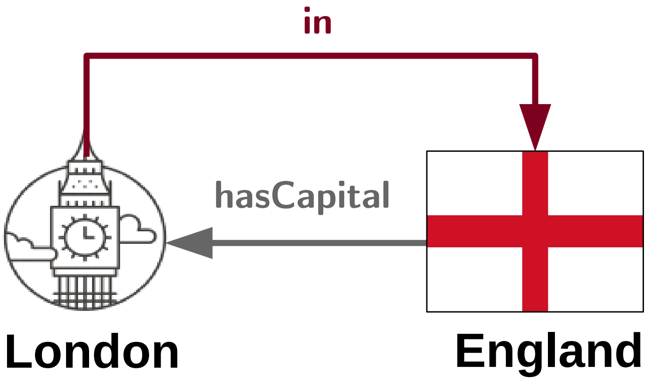
hasCapital(x, y) ⇒ in(y, x)



Applications of Rule Mining

- Fact prediction
- Domain description
  - Finding trends in KBs
- Data engineering and maintenance
  - Schema mining
  - Data correction

    hasCapital(x, y) ⇒ in(y, x)



Ecuador

Applications of Rule Mining

- Fact prediction
- Domain description

Goal: Mine rules that draw concrete and correct conclusions

- Data correction

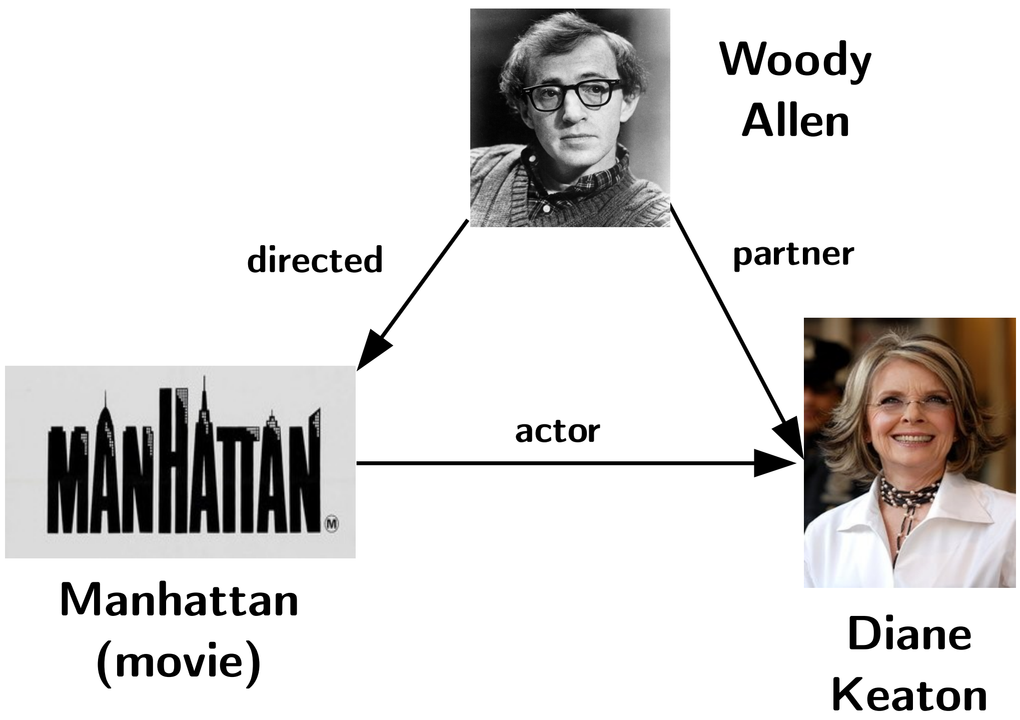
Challenges

Generate counter-evidence

Counter-examples are required to evaluate the quality of rules

Generate counter-evidence

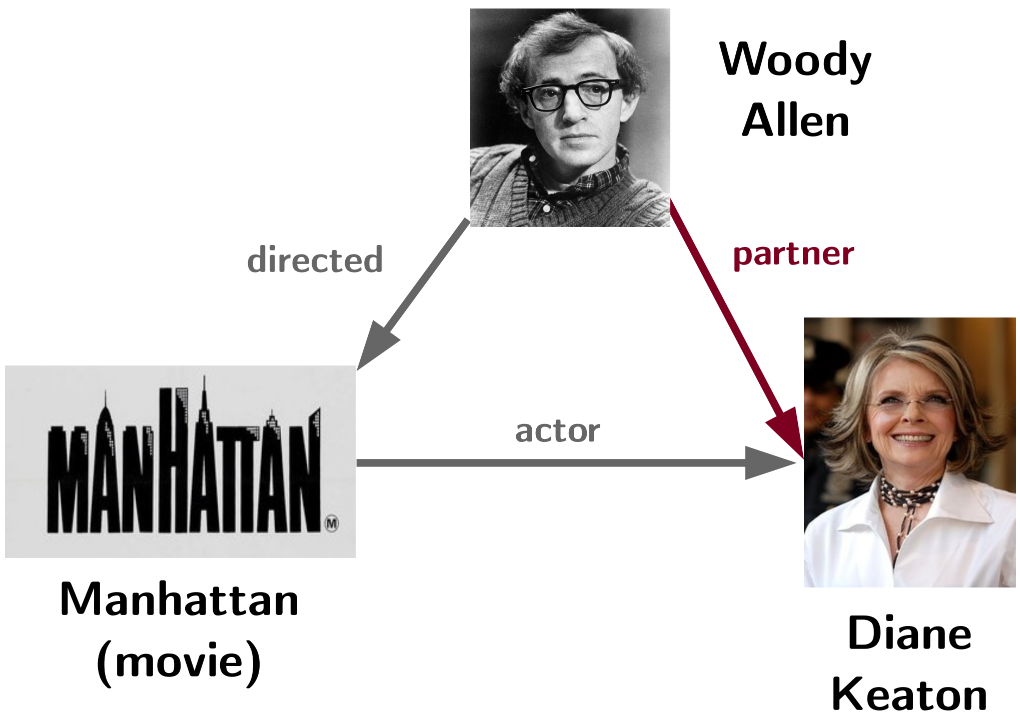
Counter-examples are required to evaluate the quality of rules



directed(x, z), actor(z, y) $\Rightarrow$ partner(x, y)

Generate counter-evidence

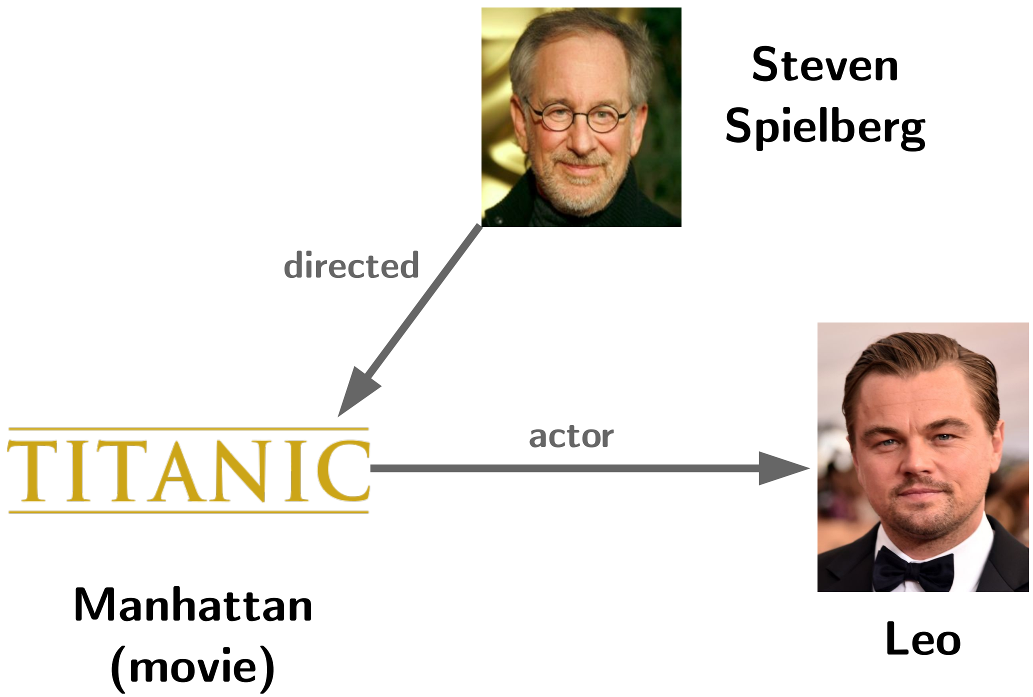
Counter-examples are required to evaluate the quality of rules



$$\text{directed}(x, z), \text{actor}(z, y) \Rightarrow \text{partner}(x, y)$$

Generate counter-evidence

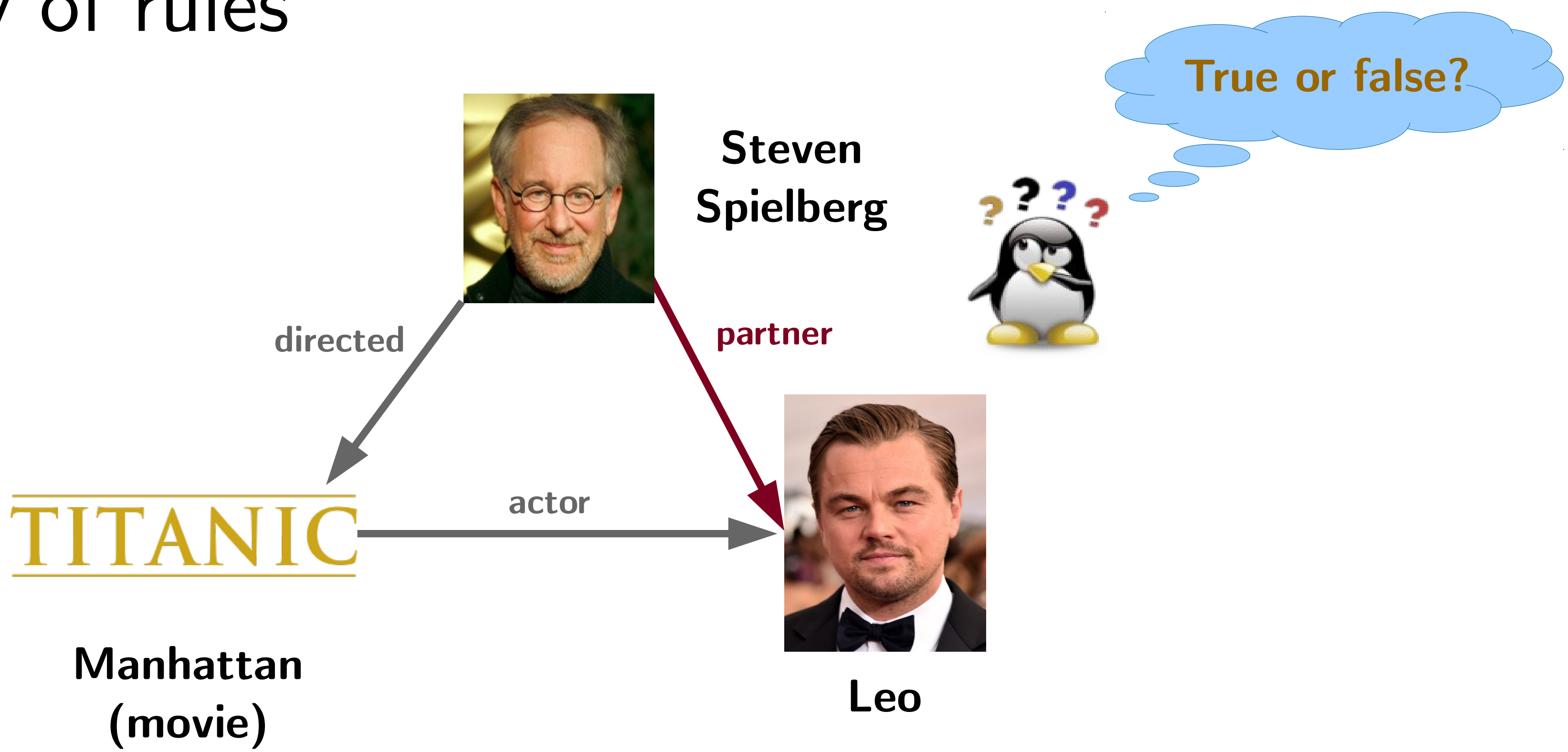
Counter-examples are required to evaluate the quality of rules



directed(x, z), actor(z, y) $\Rightarrow$ partner(x, y)

Generate counter-evidence

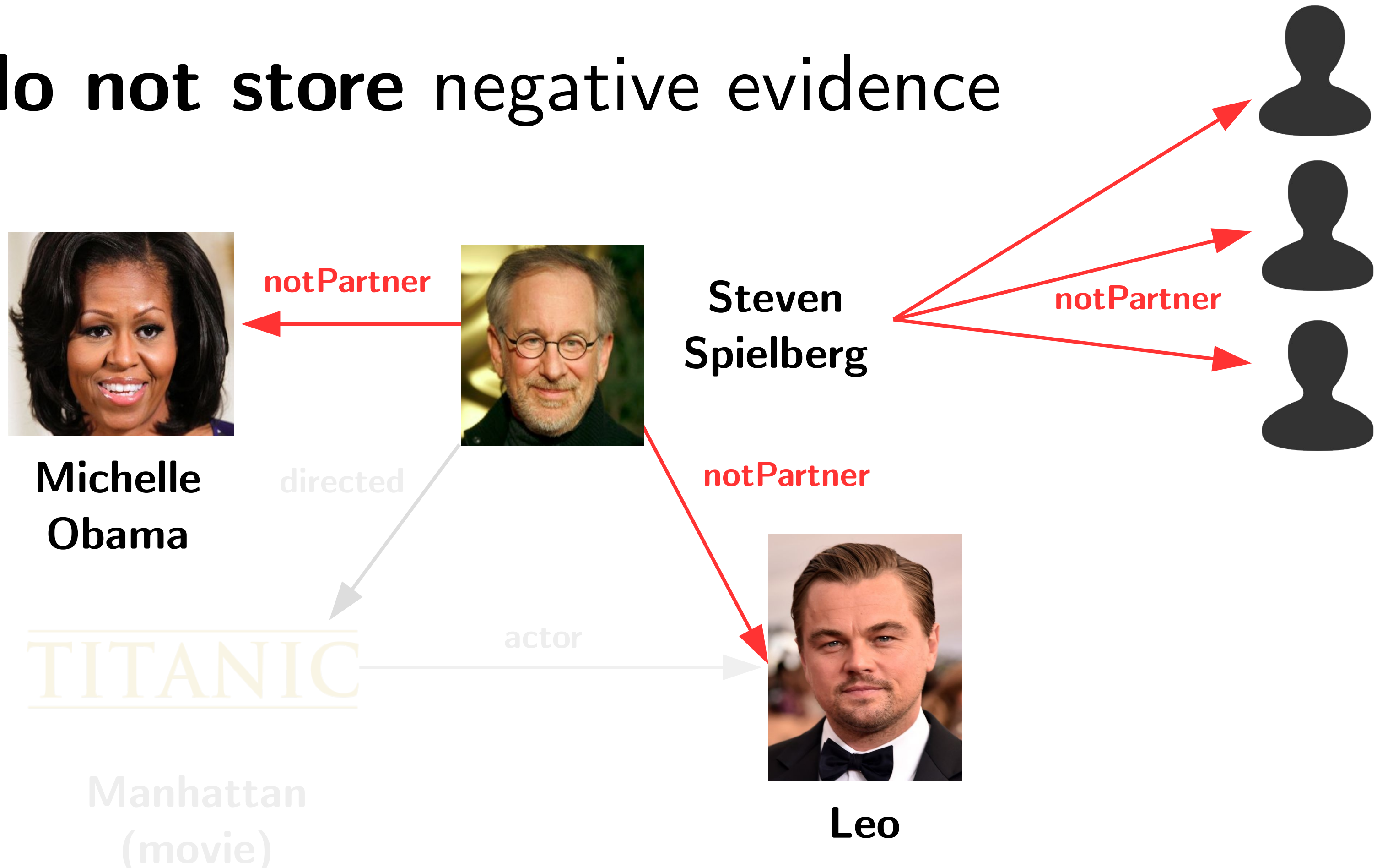
Counter-examples are required to evaluate the quality of rules



directed(x, z), actor(z, y) ⇒ partner(x, y)

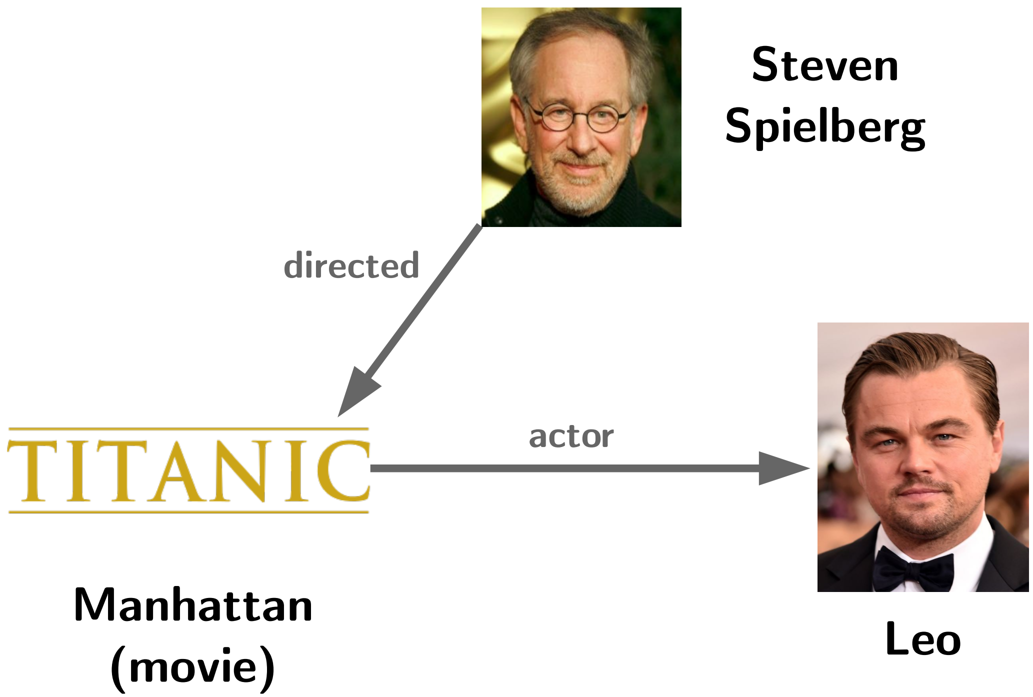
Generate counter-evidence

KBs **do not store** negative evidence



How to generate counter-evidence?

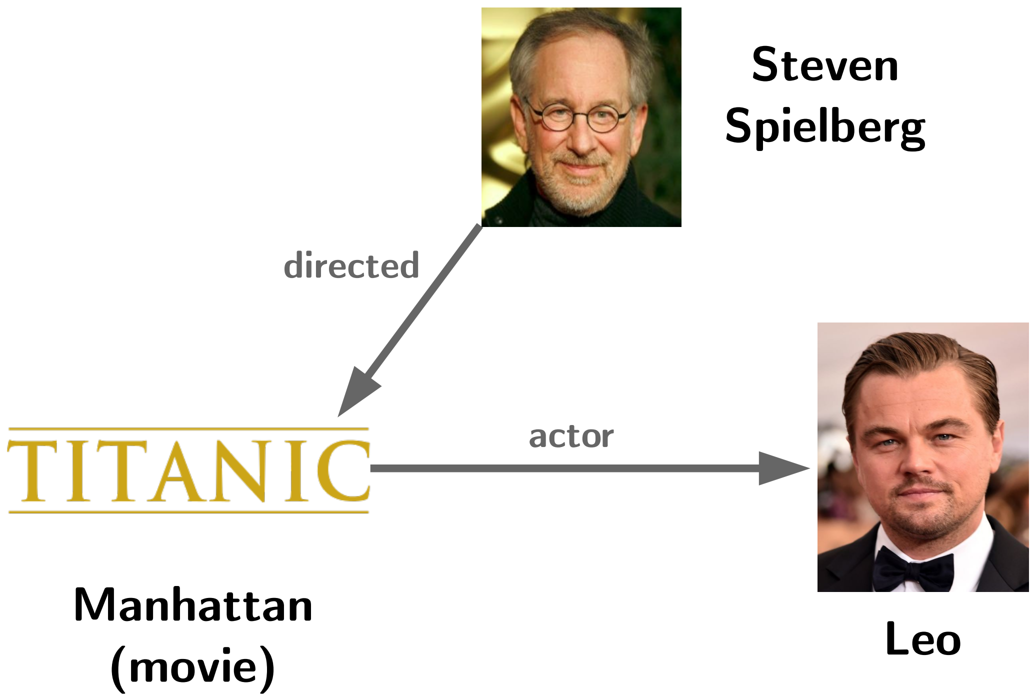
- Closed World Assumption (CWA)



directed(x, z), actor(z, y) $\Rightarrow$ partner(x, y)

How to generate counter-evidence?

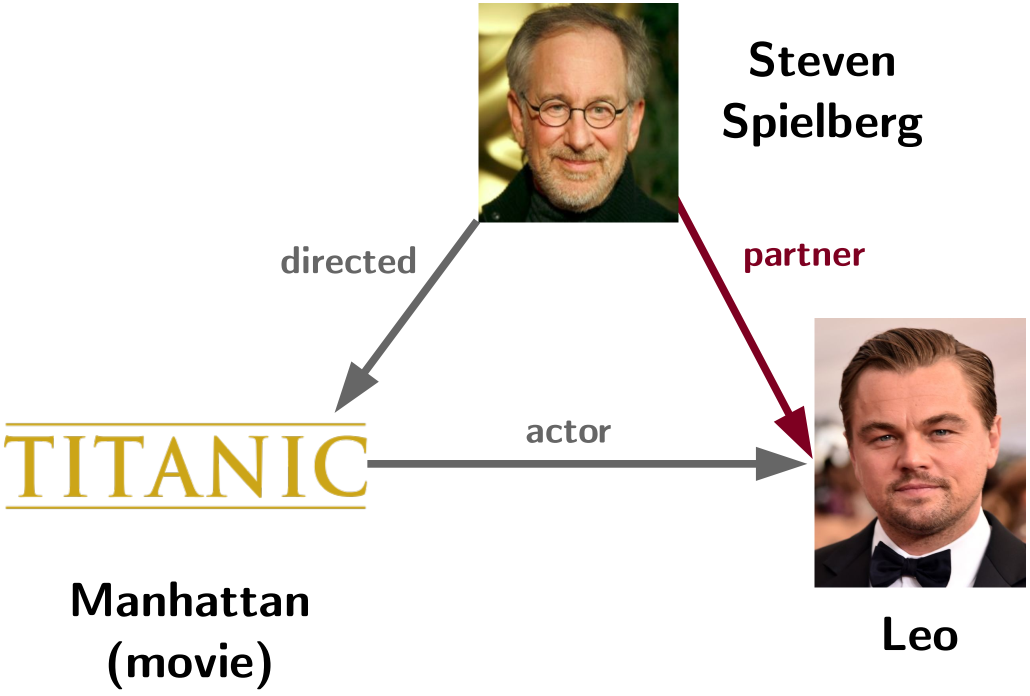
- Closed World Assumption (CWA)
  - Missing predictions are used as counter-evidence



directed(x, z), actor(z, y) $\Rightarrow$ partner(x, y)

How to generate counter-evidence?

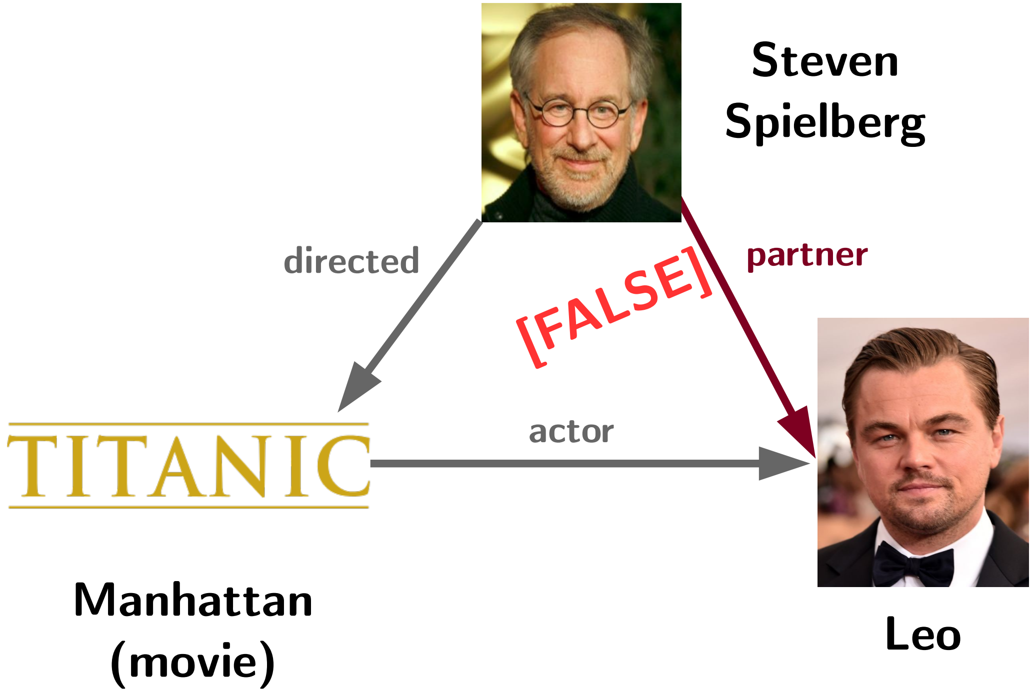
- Closed World Assumption (CWA)
  - Missing predictions are used as counter-evidence



directed(x, z), actor(z, y) $\Rightarrow$ partner(x, y)

How to generate counter-evidence?

- Closed World Assumption (CWA)
  - Missing predictions are used as counter-evidence

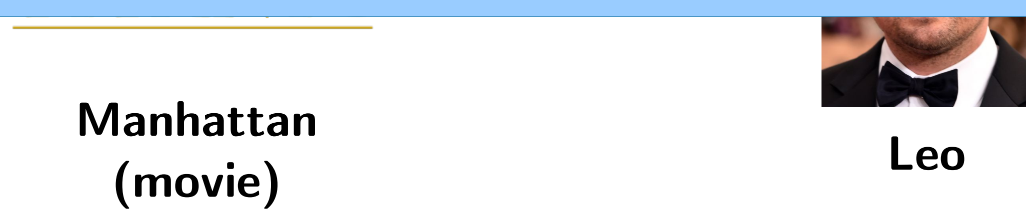


directed(x, z), actor(z, y) $\Rightarrow$ partner(x, y)

How to generate counter-evidence?

- Closed World Assumption (CWA)
  - Missing predictions are used as counter-evidence

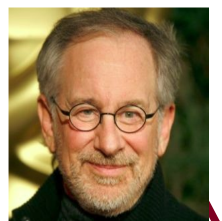
$$\text{conf}_{\text{std}}(B \Rightarrow r(x, y)) = \frac{\#\text{ predictions in KB}}{\#\text{ all predictions}}$$



directed(x, z), actor(z, y) ⇒ partner(x, y)

How to generate counter-evidence?

- Closed World Assumption (CWA)
  - Missing predictions are used as counter-evidence

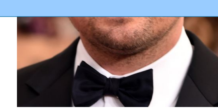


Steven Spielberg

directed

partner

$$\mathrm{conf}_{std}(B \Rightarrow r(x, y)) = \frac{\mathrm{support}(B \Rightarrow r(x, y))}{\#(x, y) : \exists\, z_1, \ldots z_n : B}$$



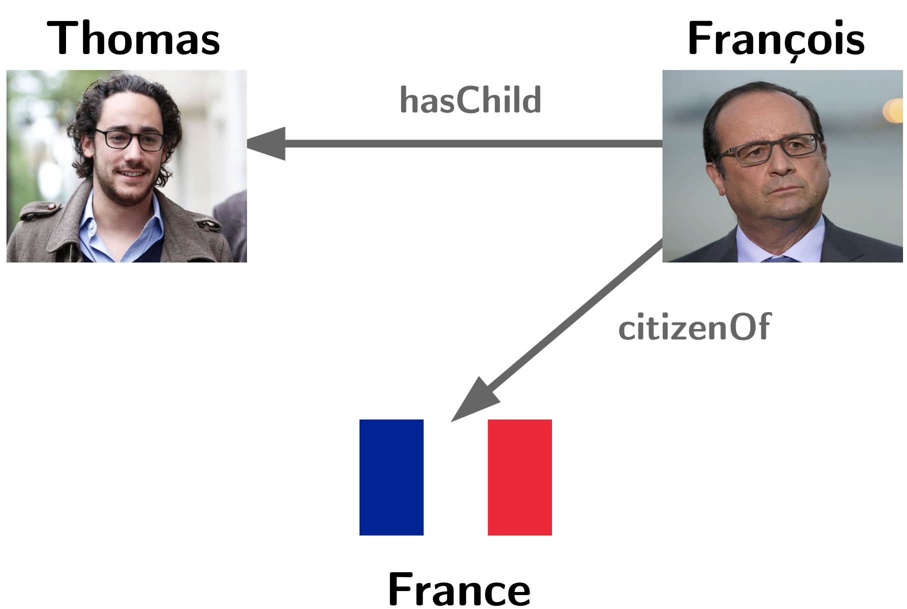
Manhattan (movie)

Leo

directed(x, z), actor(z, y) ⇒ partner(x, y)

How to generate counter-evidence?

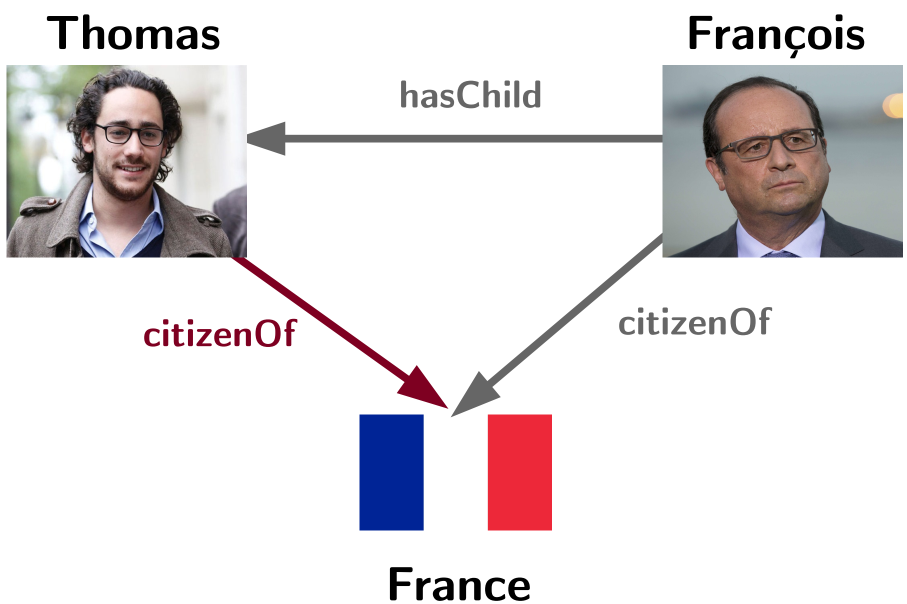
- Closed World Assumption (CWA)
  - It is too restrictive most of the times



citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

How to generate counter-evidence?

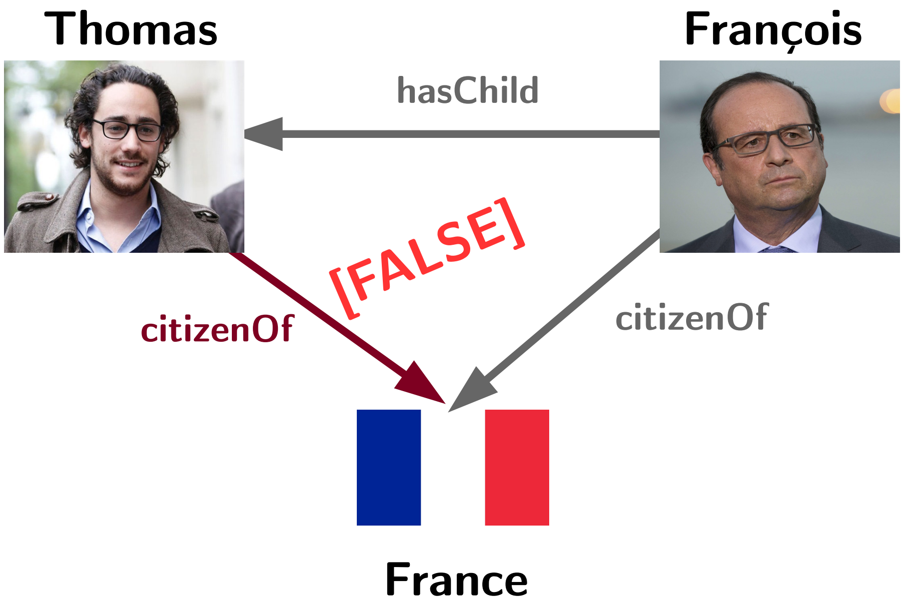
- Closed World Assumption (CWA)
    - It is too restrictive most of the times



citizenOf(z, y), hasChild(z, x) $\Rightarrow$ citizenOf(x, y)

How to generate counter-evidence?

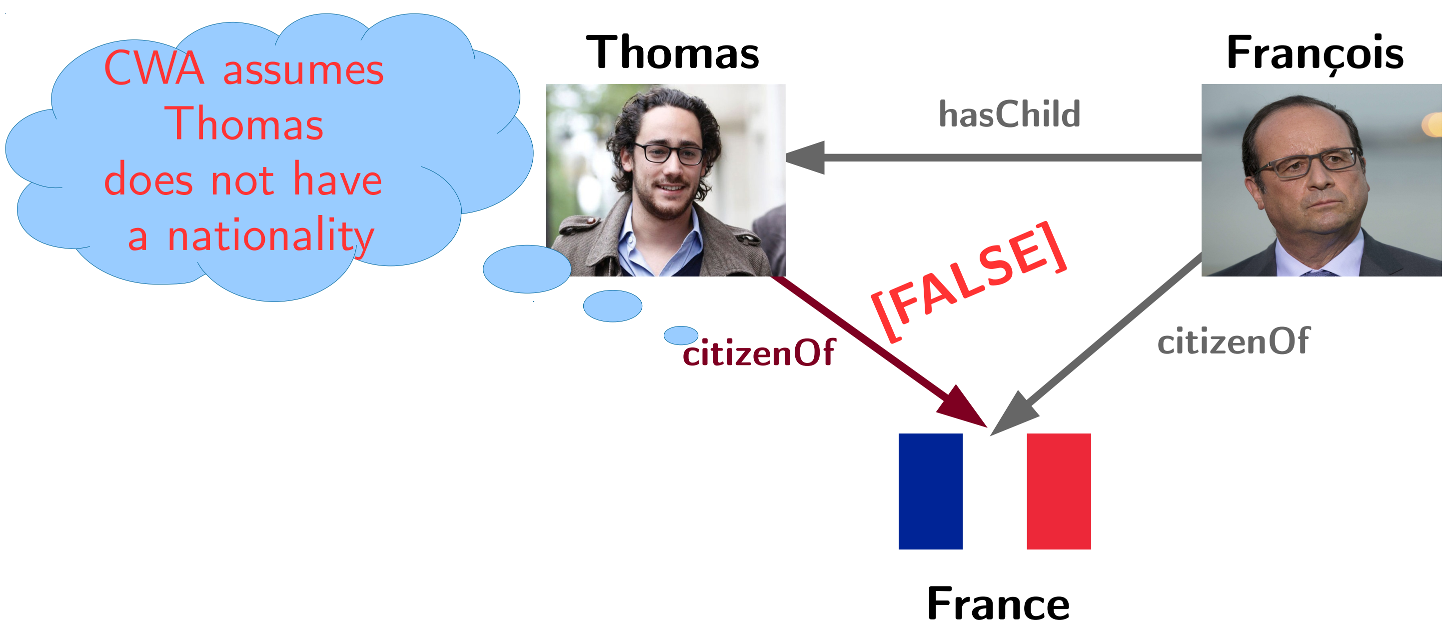
- Closed World Assumption (CWA)
  - It is too restrictive most of the times



$$\text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y)$$

How to generate counter-evidence?

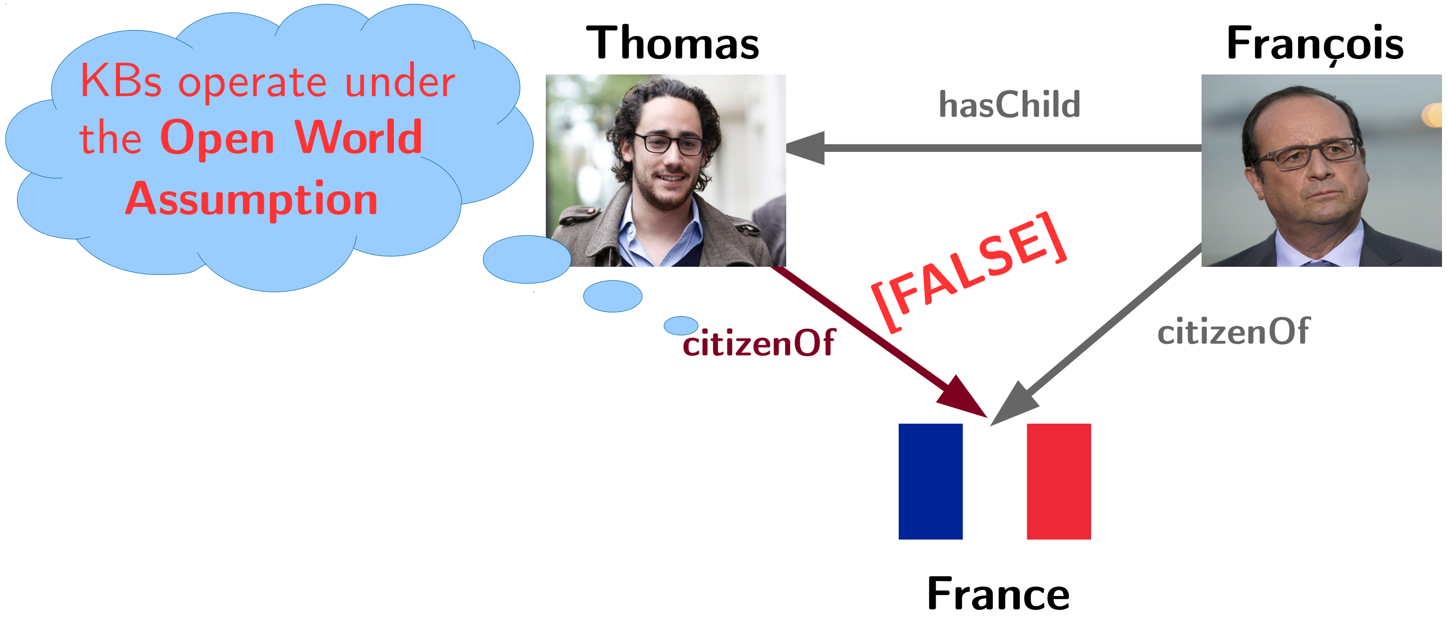
- Closed World Assumption (CWA)
  - It is too restrictive most of the times



citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

How to generate counter-evidence?

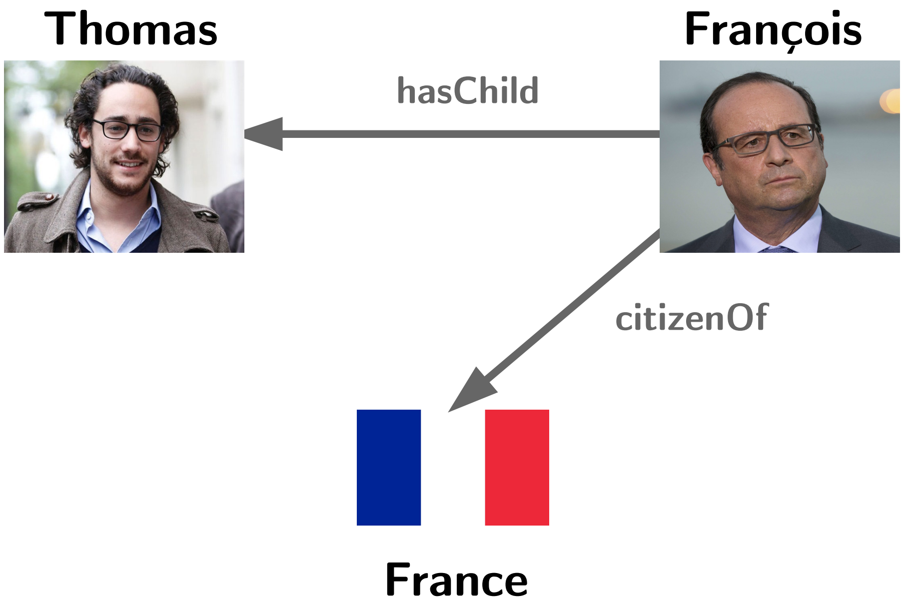
- Closed World Assumption (CWA)
  - It is too restrictive most of the times



citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

How to generate counter-evidence?

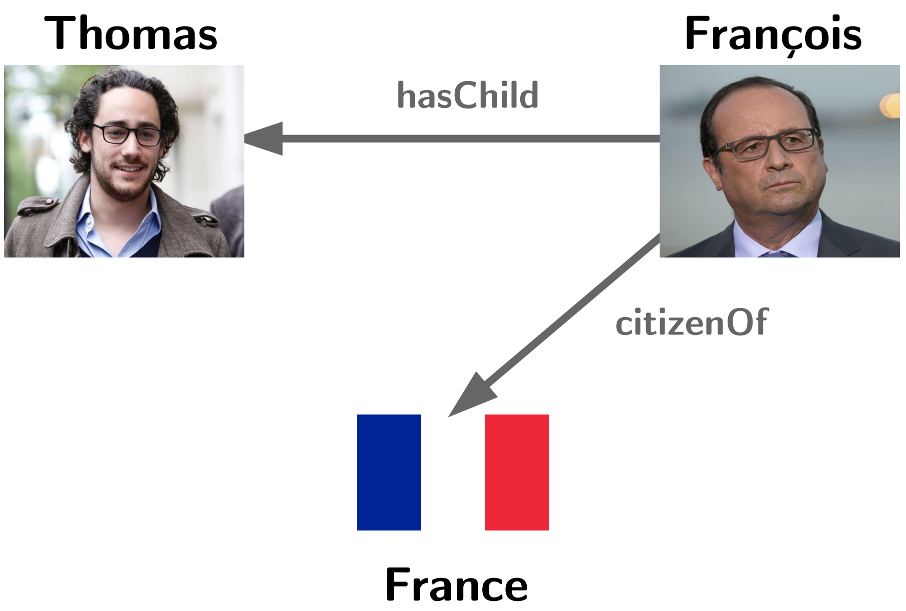
- Partial Completeness Assumption (PCA)



citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

How to generate counter-evidence?

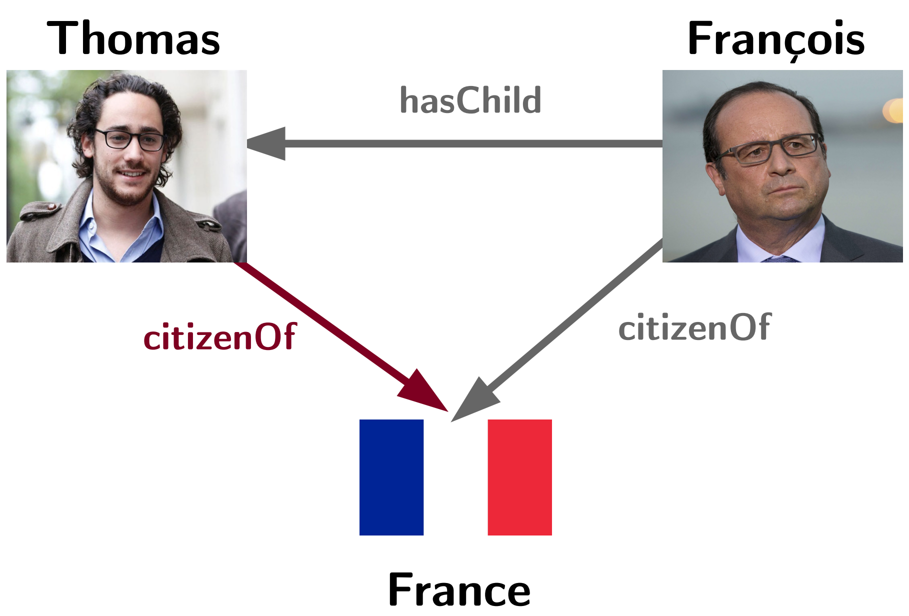
- Partial Completeness Assumption (PCA)
  - If we know at least one object, we know them all



citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

How to generate counter-evidence?

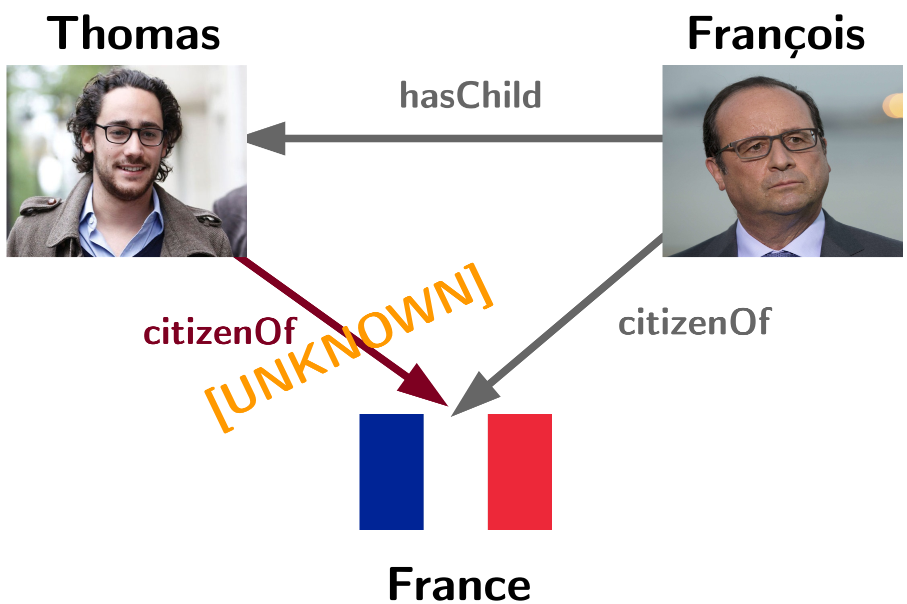
- Partial Completeness Assumption (PCA)
  - If we know at least one object, we know them all



citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

How to generate counter-evidence?

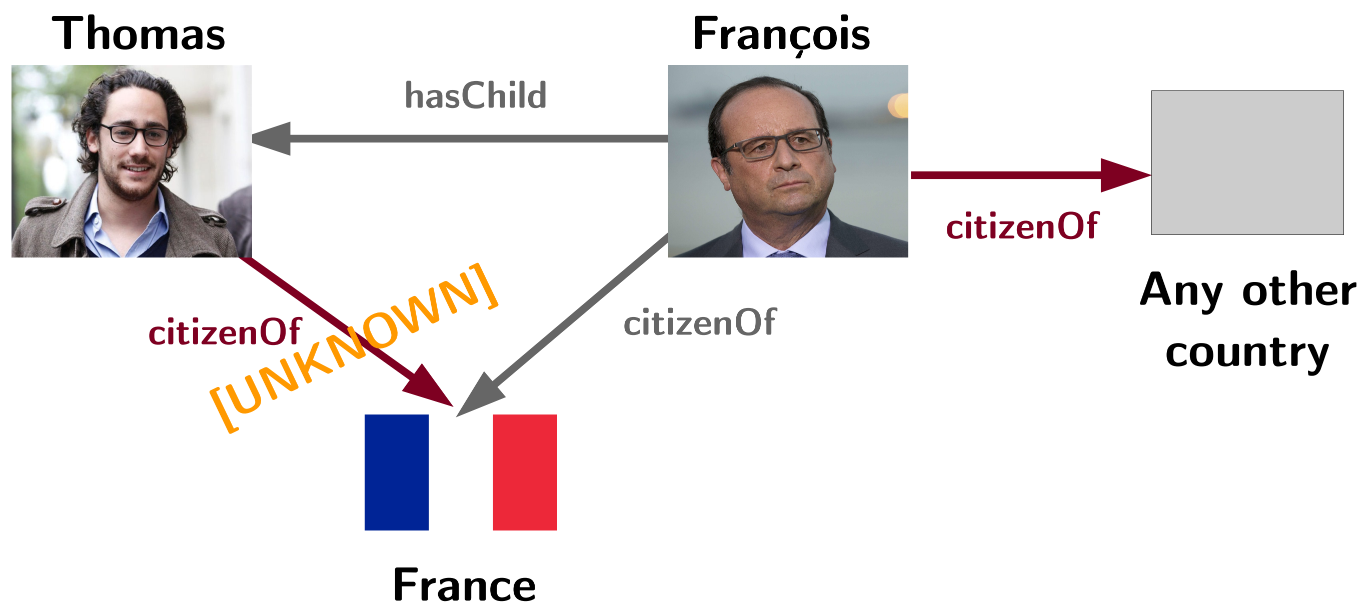
- Partial Completeness Assumption (PCA)
  - If we know at least one object, we know them all



citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

How to generate counter-evidence?

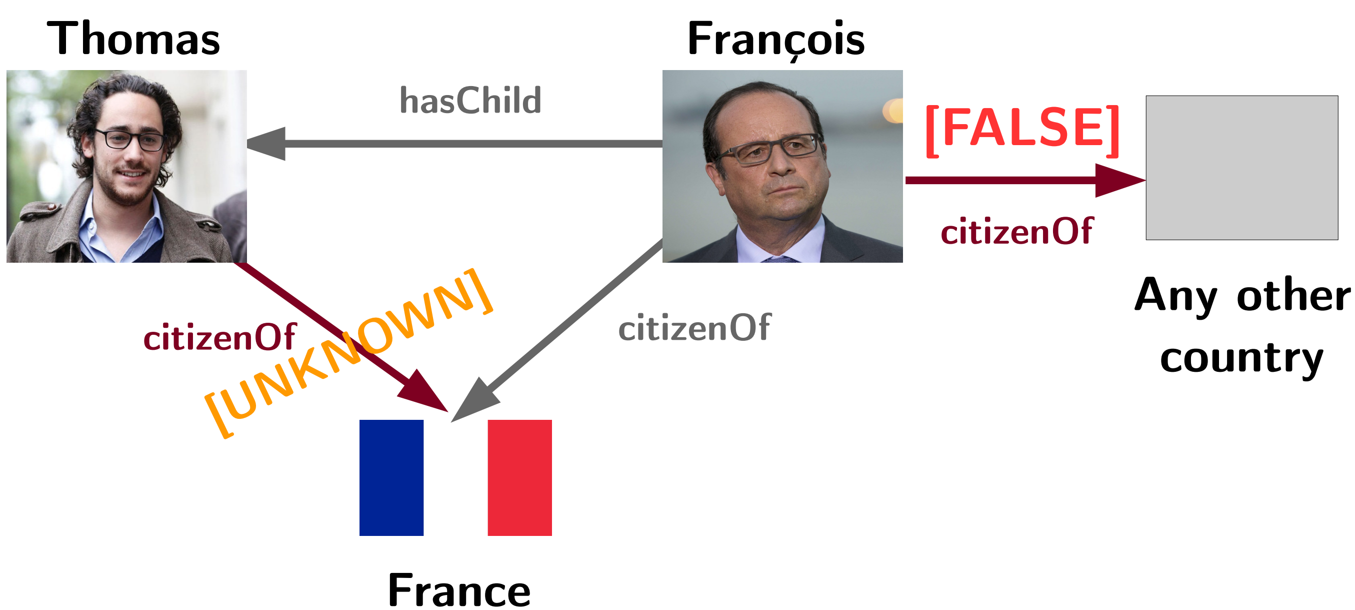
- Partial Completeness Assumption (PCA)
  - If we know at least one object, we know them all



citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

How to generate counter-evidence?

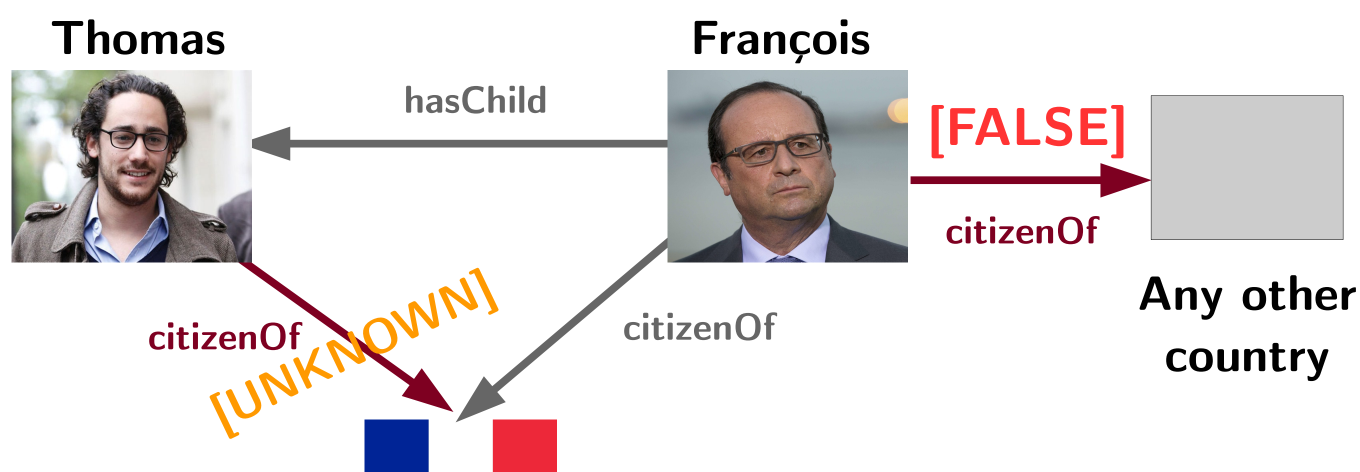
- Partial Completeness Assumption (PCA)
  - If we know at least one object, we know them all



citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

How to generate counter-evidence?

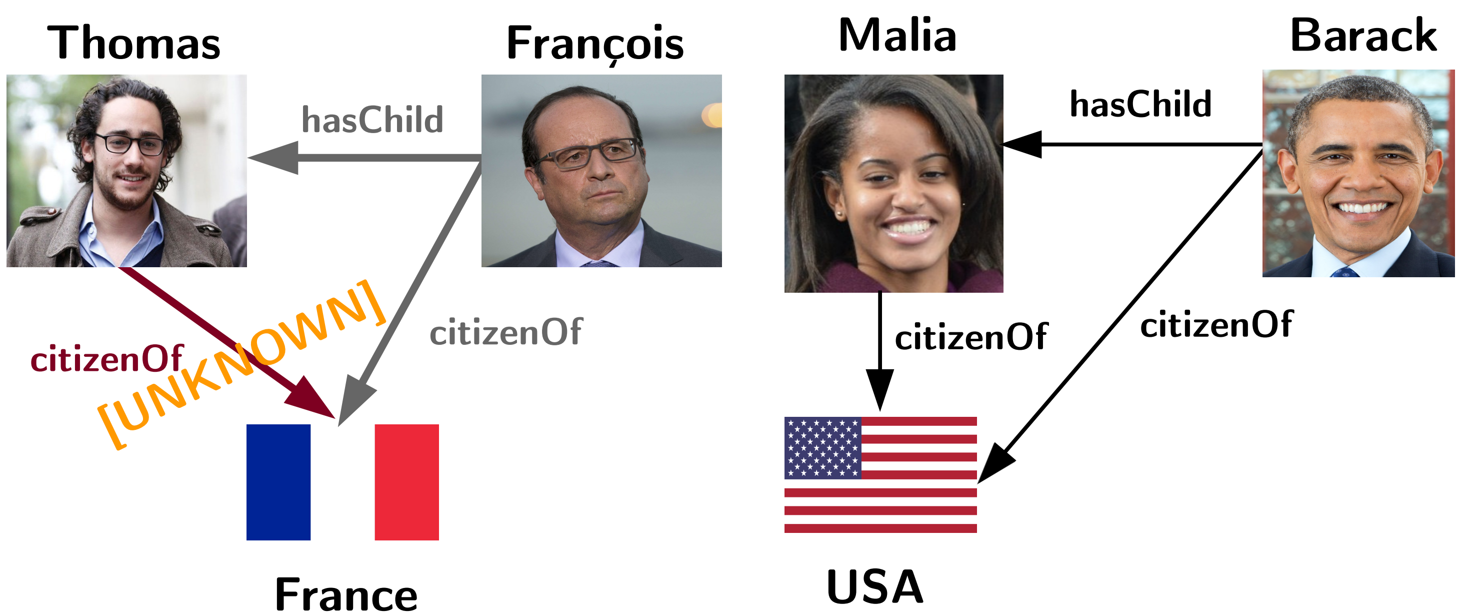
- Partial Completeness Assumption (PCA)
  - If we know at least one object, we know them all



$$\text{conf}_{\text{pca}}(B \Rightarrow r(x, y)) = \frac{\#\ \text{positives}}{\#\ \text{positives} + \#\ \text{PCA negatives}}$$

$$\text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y)$$

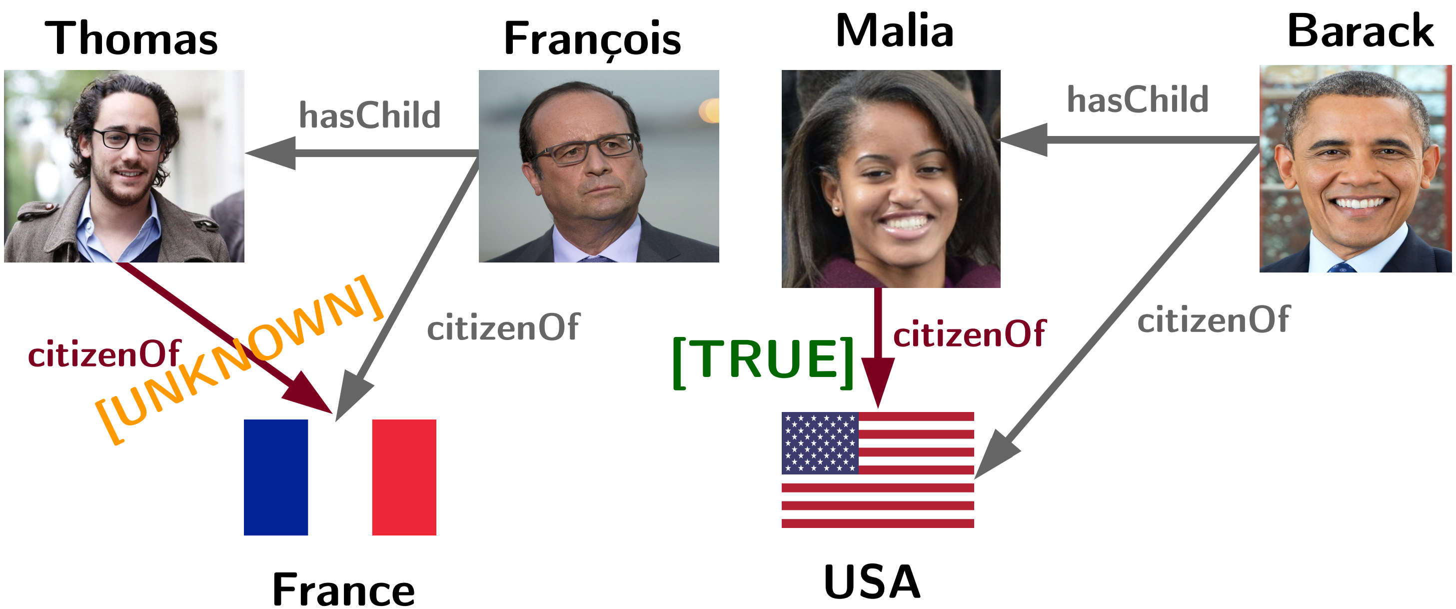
PCA Confidence



$$\mathrm{conf}_{pca}(B \Rightarrow r(x, y)) = \frac{\#\ \text{positives}}{\#\ \text{positives} + \#\ \text{PCA negatives}}$$

citizenOf(y, z), hasChild(y, x) ⇒ citizenOf(x, z)

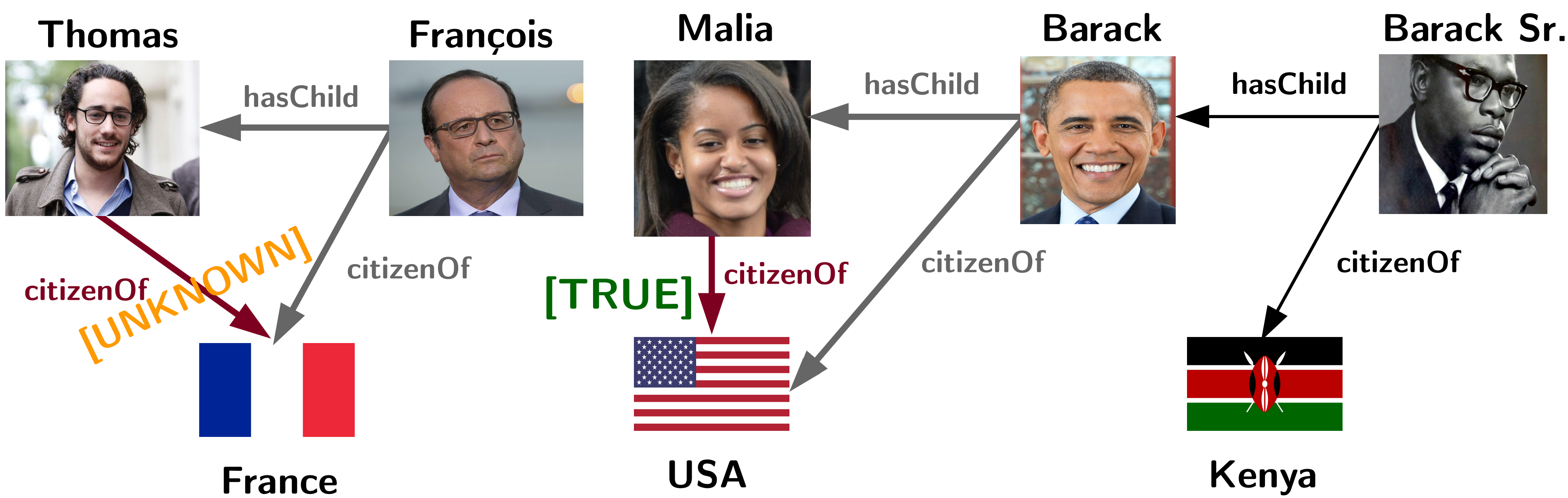
PCA Confidence



$$\mathrm{conf}_{pca}(B \Rightarrow r(x, y)) = \frac{\#\ \text{positives}}{\#\ \text{positives} + \#\ \text{PCA negatives}} \qquad \frac{1}{1}$$

citizenOf(y, z), hasChild(y, x) ⇒ citizenOf(x, z)

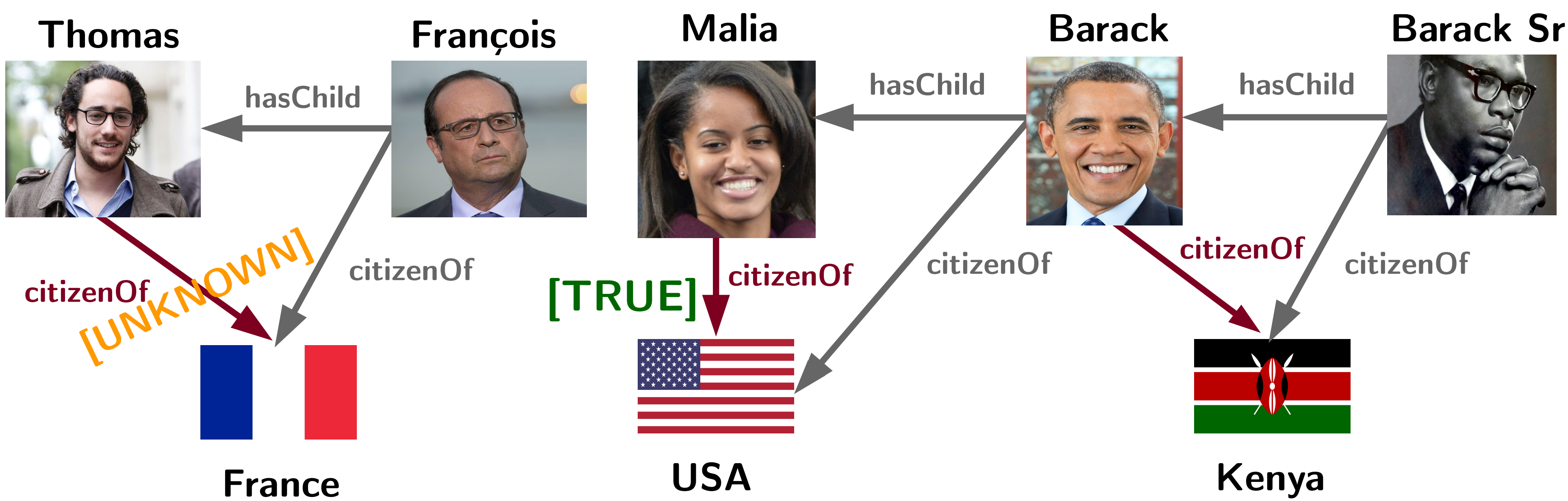
PCA Confidence



$$\mathrm{conf}_{pca}(B \Rightarrow r(x, y)) = \frac{\#\ \text{positives}}{\#\ \text{positives} + \#\ \text{PCA negatives}} \quad \frac{1}{1}$$

$$\mathrm{citizenOf}(y, z),\ \mathrm{hasChild}(y, x) \Rightarrow \mathrm{citizenOf}(x, z)$$

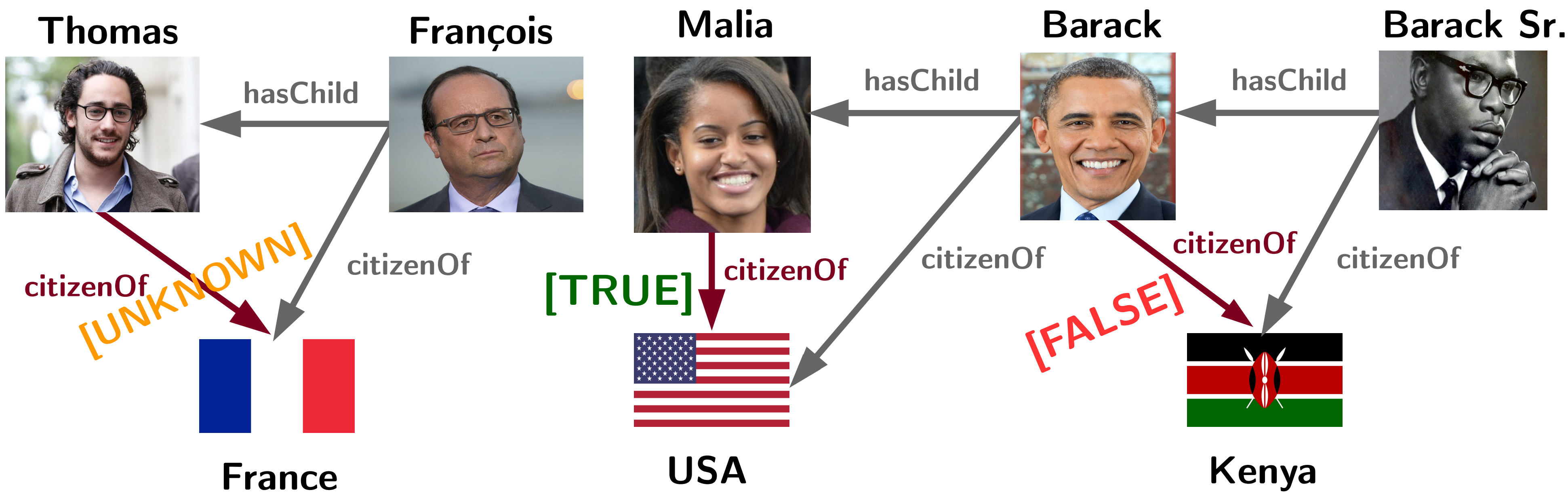
PCA Confidence



$$\text{conf}_{\text{pca}}(B \Rightarrow r(x, y)) = \frac{\#\ \text{positives}}{\#\ \text{positives} + \#\ \text{PCA negatives}} \qquad \frac{1}{1}$$

citizenOf(y, z), hasChild(y, x) ⇒ citizenOf(x, z)

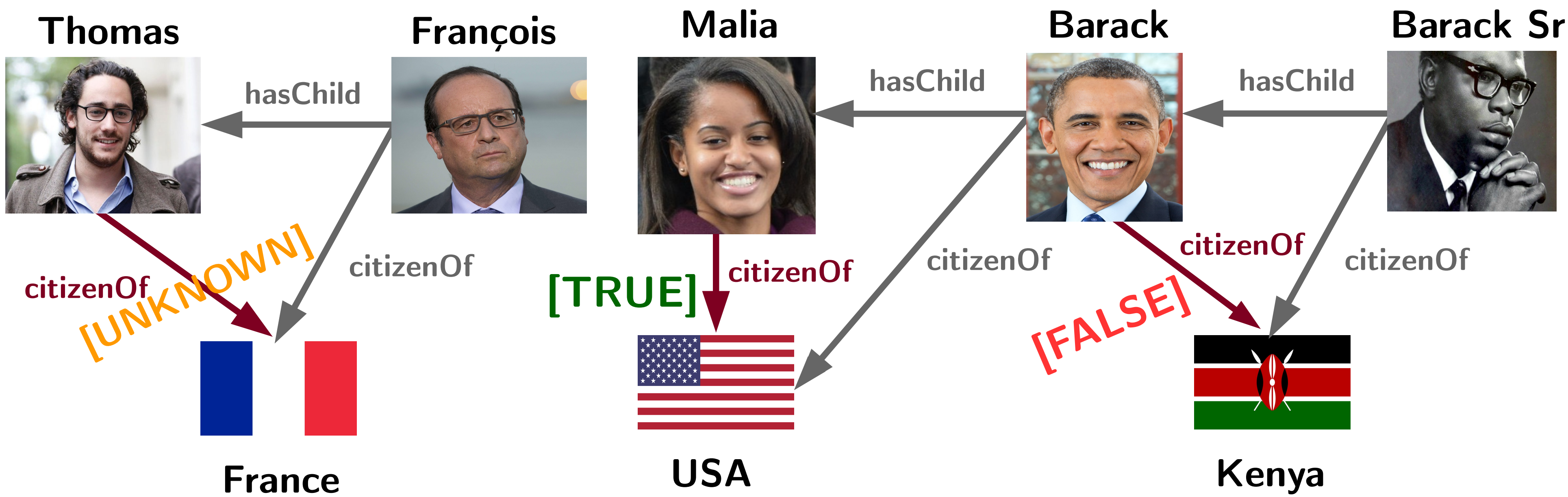
PCA Confidence



$$\mathrm{conf}_{pca}(B \Rightarrow r(x, y)) = \frac{\#\ \text{positives}}{\#\ \text{positives} + \#\ \text{PCA negatives}} \qquad \frac{1}{2}$$

citizenOf(y, z), hasChild(y, x) ⇒ citizenOf(x, z)

PCA Confidence



$$\mathrm{conf}_{pca}(B \Rightarrow r(x, y)) = \frac{\mathrm{support}(B \Rightarrow r(x, y))}{\#(x, y) : \exists\, z_1,\ldots z_n, y' : B \wedge r(x, y')} \qquad \frac{\mathbf{1}}{\mathbf{2}}$$

$\mathrm{citizenOf}(y, z), \mathrm{hasChild}(y, x) \Rightarrow \mathrm{citizenOf}(x, z)$

KBs are large

Dataset	Facts	Entities
YAGO	120M	10M
Dbpedia	6.9B	38M
Wikidata	100M	20M

State-of-the-art approaches do not scale

Dataset	# facts	WARMR	ALEPH
YAGO core	1M	-	5s to 1d
YAGO (sample)	47K	18h	0.05s to 1d

How to mine rules efficiently?

How to mine rules efficiently?

- Start with all possible rules of the form $\Rightarrow$ **r(x,y)**
  - Refine the rules iteratively by means of mining operators:
    - Add dangling atom (O_D)
    - Add closing atom (O_C)
    - Add instantiated atom (O_I)

How to mine rules efficiently?

citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

⇒ citizenOf(x, y)

How to mine rules efficiently?

$\text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y)$

$\Rightarrow \text{citizenOf}(x, y)$

Add dangling atom $?r(z, \mathbf{x}) \Rightarrow \text{citizenOf}(\mathbf{x}, y)$

How to mine rules efficiently?

citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

⇒ citizenOf(x, y)

Add dangling atom

?r(z, **x**) ⇒ citizenOf(**x**, y)

hasChild
influences
....

How to mine rules efficiently?

citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

⇒ citizenOf(x, y)

Add dangling atom

?r(z, **x**) ⇒ citizenOf(**x**, y)

hasChild
influences
....

How to mine rules efficiently?

citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

⇒ citizenOf(x, y)

Add dangling atom ?r(z, x) ⇒ citizenOf(x, y)

hasChild(z, x) ⇒ citizenOf(x, y)

How to mine rules efficiently?

citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

⇒ citizenOf(x, y)

Add dangling atom ?r(z, x) ⇒ citizenOf(x, y)

hasChild(z, x) ⇒ citizenOf(x, y)

Add closing atom ?r(**z**, **y**) hasChild(**z**, x) ⇒ citizenOf(x, **y**)

citizenOf
livesIn
...

How to mine rules efficiently?

citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

⇒ citizenOf(x, y)

Add dangling atom ?r(z, x) ⇒ citizenOf(x, y)

hasChild(z, x) ⇒ citizenOf(x, y)

Add closing atom ?r(**z**, **y**) hasChild(**z**, x) ⇒ citizenOf(x, **y**)

citizenOf
livesIn
...

How to mine rules efficiently?

$\boxed{\text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y)}$

$\Rightarrow \text{citizenOf}(x, y)$

Add dangling atom $?r(z, x) \Rightarrow \text{citizenOf}(x, y)$

$\text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y)$

Add closing atom $?r(z, y)\ \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y)$

$\text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y)$

How to mine rules efficiently?

citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

⇒ citizenOf(x, y)

Add dangling atom ?r(z, x) ⇒ citizenOf(x, y)

hasChild(z, x) ⇒ citizenOf(x, y)

Add closing atom ?r(z, y) hasChild(z, x) ⇒ citizenOf(x, y)

citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

Add instantiated atom adds atoms of the form ?r(x, C) where C is a constant, e.g., livesIn(x, USA)

How to mine rules efficiently?

- Prune the search space

How to mine rules efficiently?

- Prune the search space
  - Using monotonic definition of support and a threshold

$\text{support}(B \Rightarrow r(x, y)) = \#(x, y) : \exists\, z_1, \ldots z_n : B \wedge r(x, y)$

How to mine rules efficiently?

- Prune the search space
  - Using monotonic definition of support and a threshold

$$\text{support}(B \Rightarrow r(x, y)) = \#(x, y) : \exists\, z_1, \ldots z_n : B \wedge r(x, y)$$

Support = 3 citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

How to mine rules efficiently?

- Prune the search space
  - Using monotonic definition of support and a threshold

    support(B ⇒ r(x, y)) = #(x, y) : ∃ $z_1,\ldots z_n$: B ∧ r(x, y)

Support = 3 citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

bornIn(z, y), citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

Support = 2

How to mine rules efficiently?

- Apply a language bias that complies with our goal

How to mine rules efficiently?

- Apply a language bias that complies with our goal
  - Goal: rules that make correct and concrete predictions

How to mine rules efficiently?

- Apply a language bias that complies with our goal
  - Goal: rules that make correct and concrete predictions
  - Avoid existentially quantified conclusions

  citizenOf(z, y), hasChild(z, x) $\Rightarrow$ citizenOf(x, w)

How to mine rules efficiently?

- Apply a language bias that complies with our goal
  - Goal: rules that make correct and concrete predictions
  - Avoid existentially quantified conclusions

Existentially quantified

citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, **w**)

How to mine rules efficiently?

- Apply a language bias that complies with our goal
  - Goal: rules that make correct and concrete predictions
  - Avoid existentially quantified conclusions

$\text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \exists\, w : \text{citizenOf}(x, w)$

How to mine rules efficiently?

- Apply a language bias that complies with our goal
  - Goal: rules that make correct and concrete predictions
  - Avoid existentially quantified conclusions

    citizenOf(z, y), hasChild(z, x) ⇒ ∃ w : citizenOf(x, w)
  - Focus on **closed** Horn rules

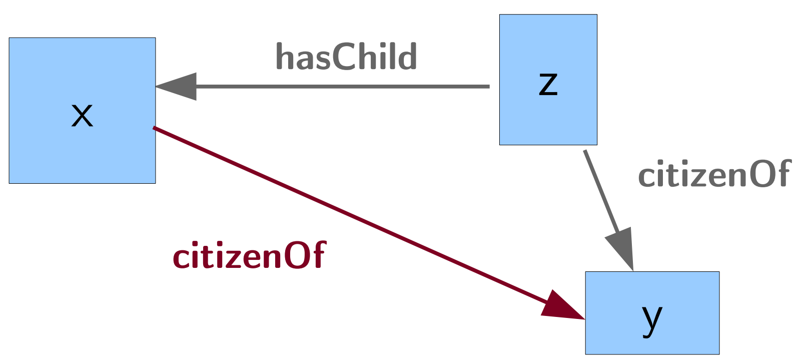
How to mine rules efficiently?

- Apply a language bias that complies with our goal
  - Goal: rules that make correct and concrete predictions
  - Avoid existentially quantified conclusions

    citizenOf(z, y), hasChild(z, x) ⇒ ∃ w : citizenOf(x, w)

  - Focus on **closed** Horn rules

    citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)



How to mine rules efficiently?

- Do not specialize rules with 100% confidence

How to mine rules efficiently?

- Do not specialize rules with 100% confidence
- Use efficient confidence approximation

How to mine rules efficiently?

- Do not specialize rules with 100% confidence
- Use efficient confidence approximation
  - To discard rules with low confidence in advance

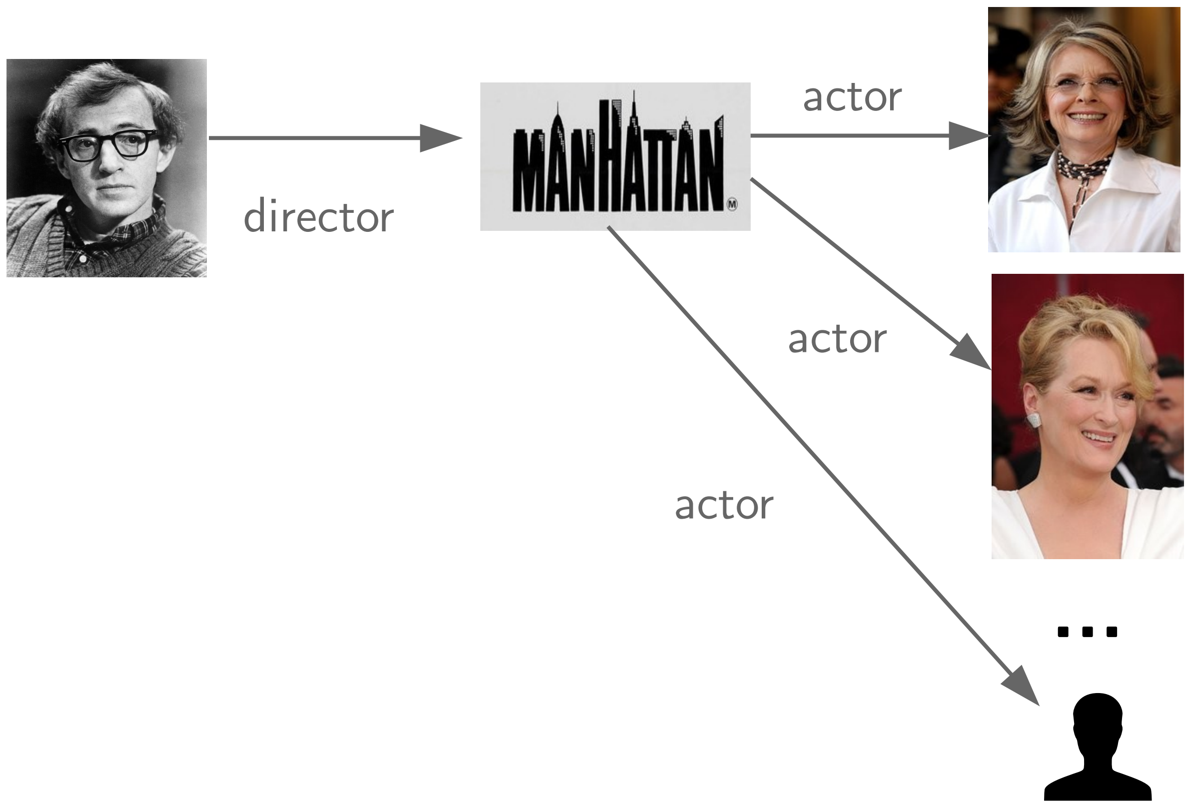
Confidence approximation

- Bad rules make a lot of false predictions per entity

Confidence approximation

- Bad rules make a lot of false predictions per entity

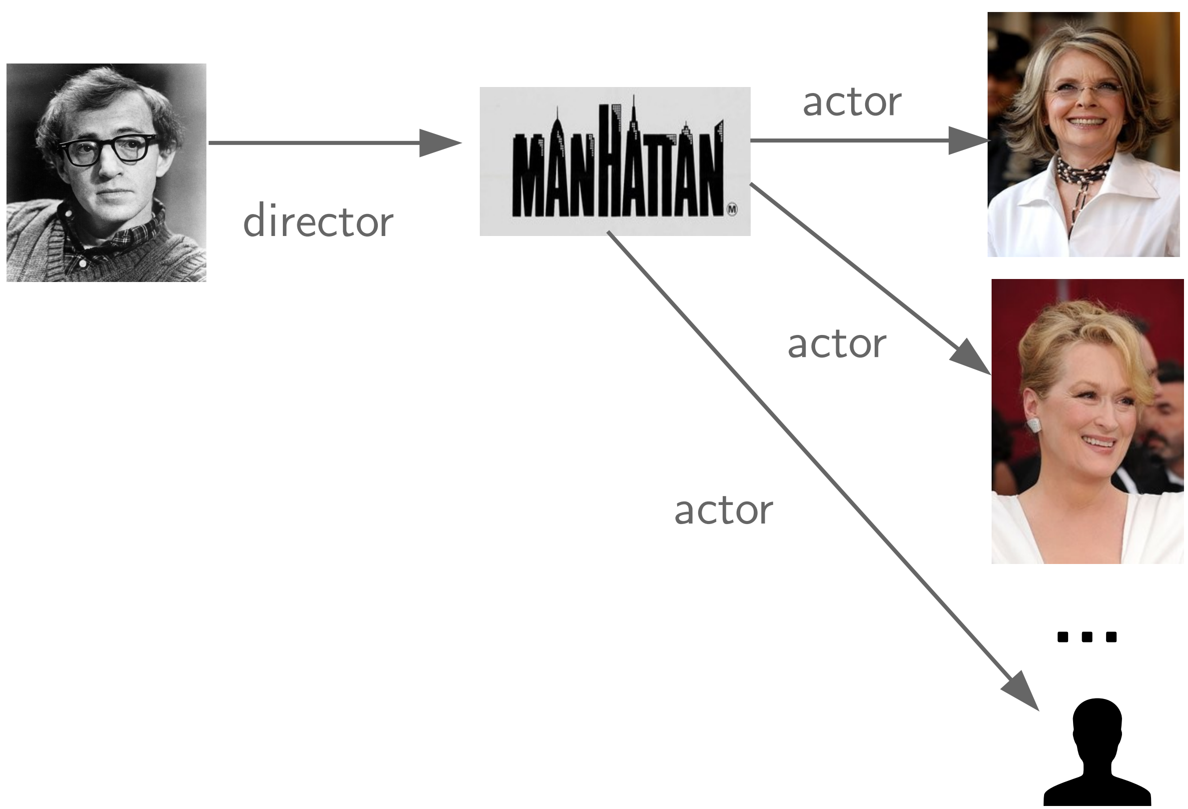
$$\text{director}(x, z), \text{actor}(z, y) \Rightarrow \text{partner}(x, y)$$



Confidence approximation

- Bad rules make a lot of false predictions per entity
  - Director is partnered with all actors of his movies

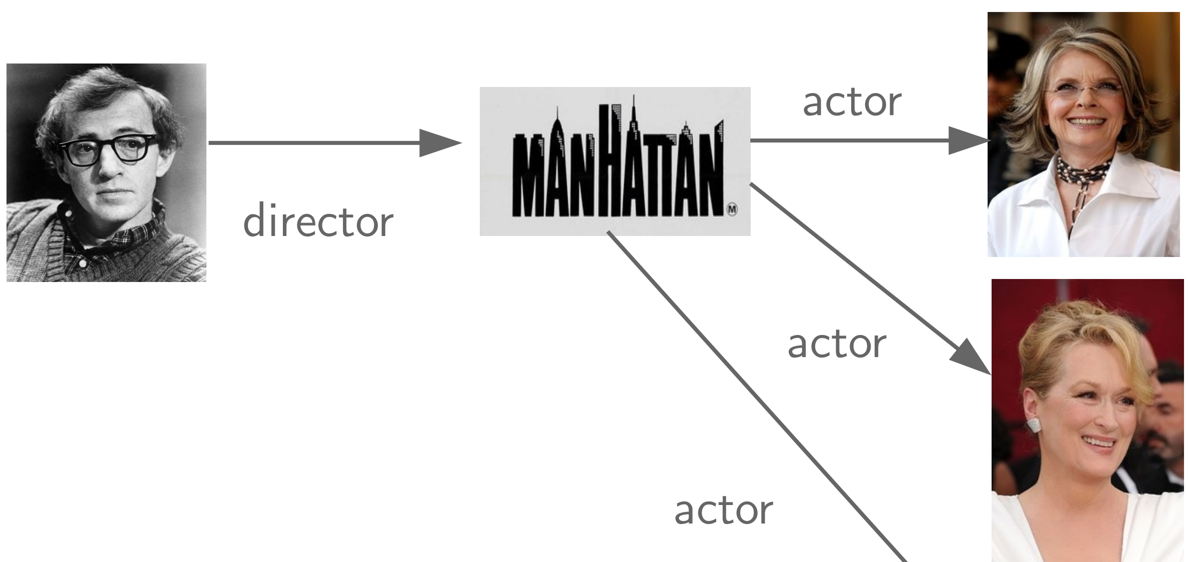
$$\text{director}(x, z), \text{actor}(z, y) \Rightarrow \text{partner}(x, y)$$

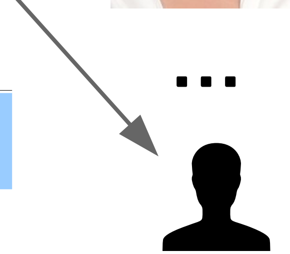


Confidence approximation

- Bad rules make a lot of false predictions per entity
  - They are counted in the denominator of the confidence

director(x, z), actor(z, y) ⇒ partner(x, y)



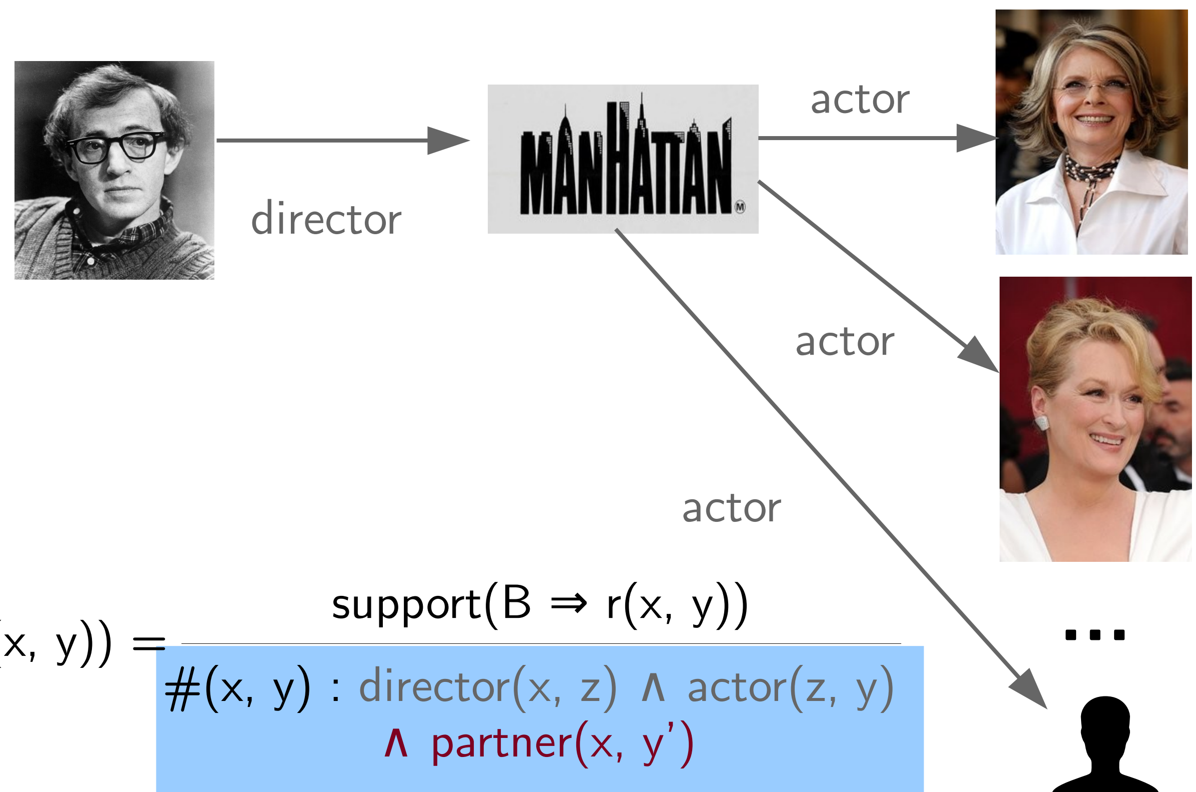


$$conf_{pca}(B \Rightarrow r(x, y)) = \frac{support(B \Rightarrow r(x, y))}{\#(x, y) : \exists\, z_1,\dots z_n, y' : B \wedge r(x, y')}$$

Confidence approximation

- Bad rules make a lot of false predictions per entity
  - They are counted in the denominator of the confidence

$$\text{director}(x, z), \text{actor}(z, y) \Rightarrow \text{partner}(x, y)$$

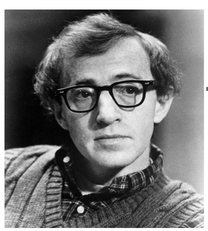


$$\text{conf}_{\text{pca}}(B \Rightarrow r(x, y)) = \frac{\text{support}(B \Rightarrow r(x, y))}{\#(x, y) : \text{director}(x, z) \wedge \text{actor}(z, y) \wedge \text{partner}(x, y')}$$

Confidence approximation

- Use statistics to estimate conf_{pca} denominator

$$\text{director}(x, z), \text{actor}(z, y) \Rightarrow \text{partner}(x, y)$$



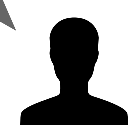
director

actor

actor

actor

...

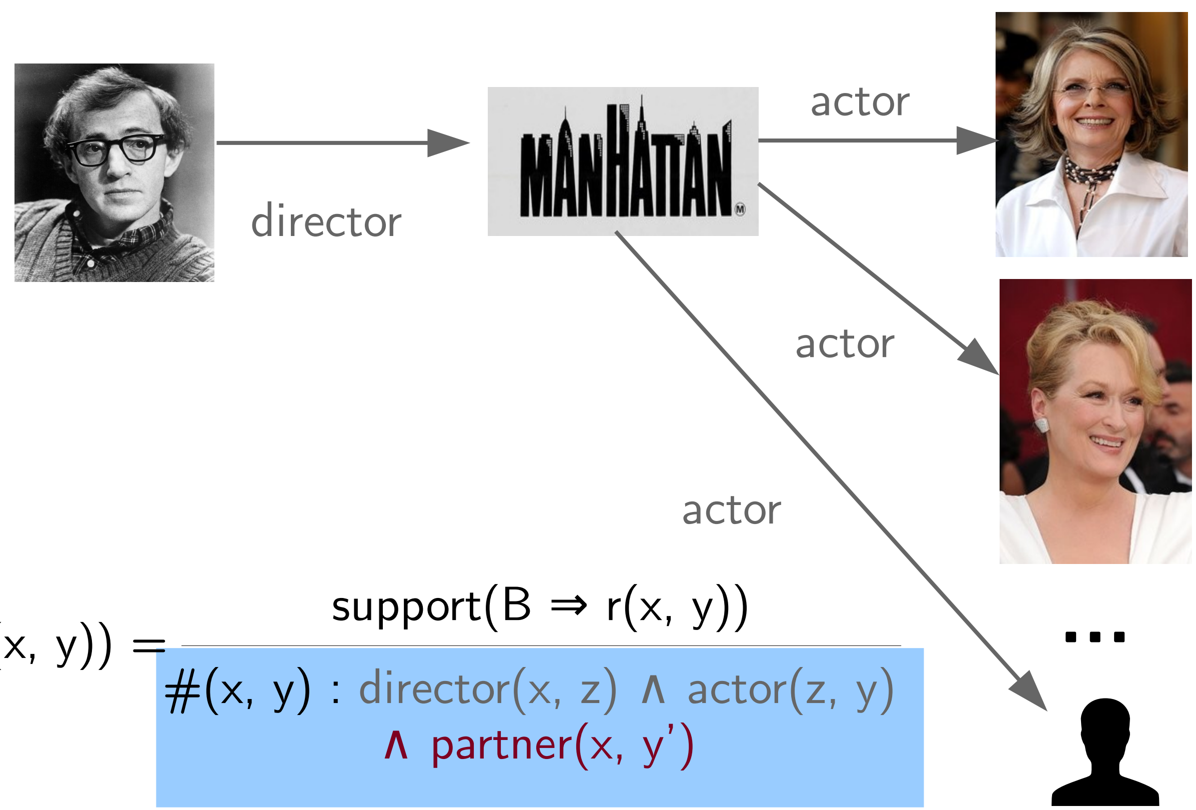


$$\text{conf}_{\text{pca}}(B \Rightarrow r(x, y)) = \frac{\text{support}(B \Rightarrow r(x, y))}{\#(x, y) : \text{director}(x, z) \wedge \text{actor}(z, y) \wedge \text{partner}(x, y')}$$

Confidence approximation

- Use statistics to estimate conf_{pca} denominator
  - (# of actors per director) × (# of partnered directors)

director(x, z), actor(z, y) ⇒ partner(x, y)

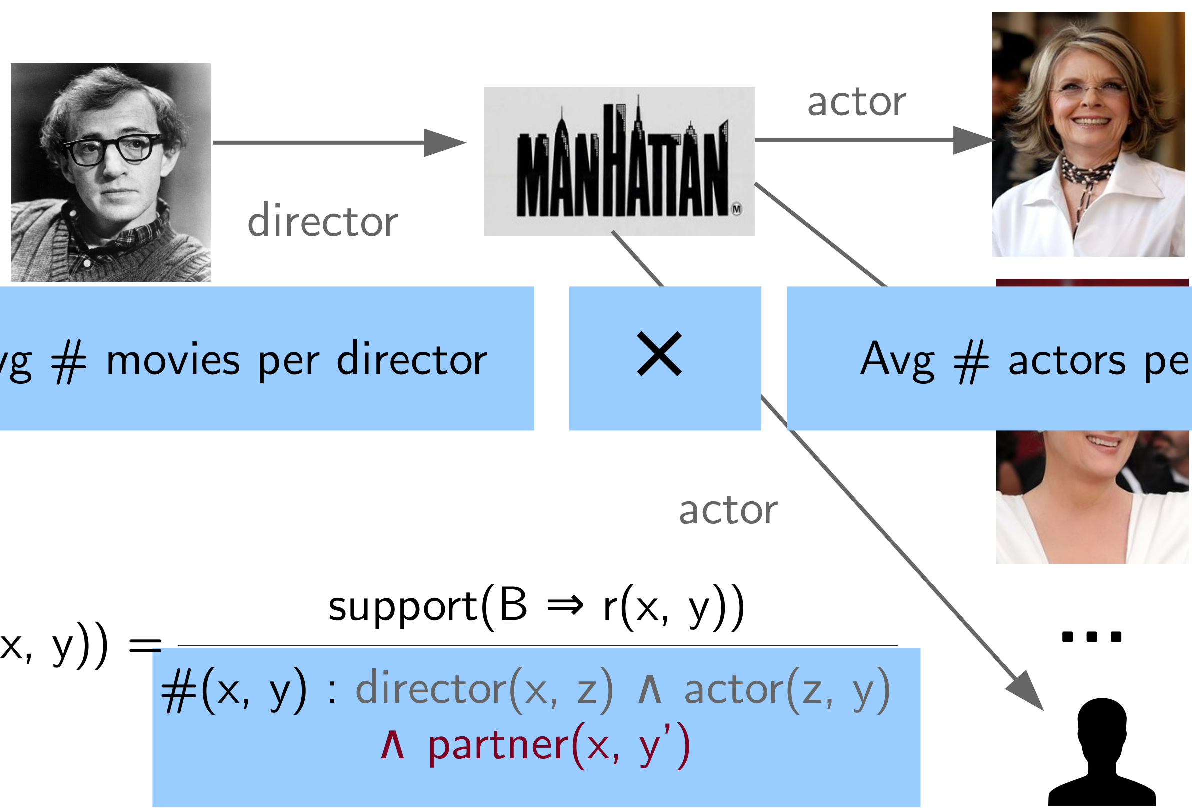


$$\text{conf}_{pca}(B \Rightarrow r(x, y)) = \frac{\text{support}(B \Rightarrow r(x, y))}{\#(x, y) : \text{director}(x, z) \wedge \text{actor}(z, y) \wedge \text{partner}(x, y')}$$

Confidence approximation

- Use statistics to estimate conf_{pca} denominator
  - (# of actors per director) × (# of partnered directors)

director(x, z), actor(z, y) ⇒ partner(x, y)



Avg # movies per director × Avg # actors per movie

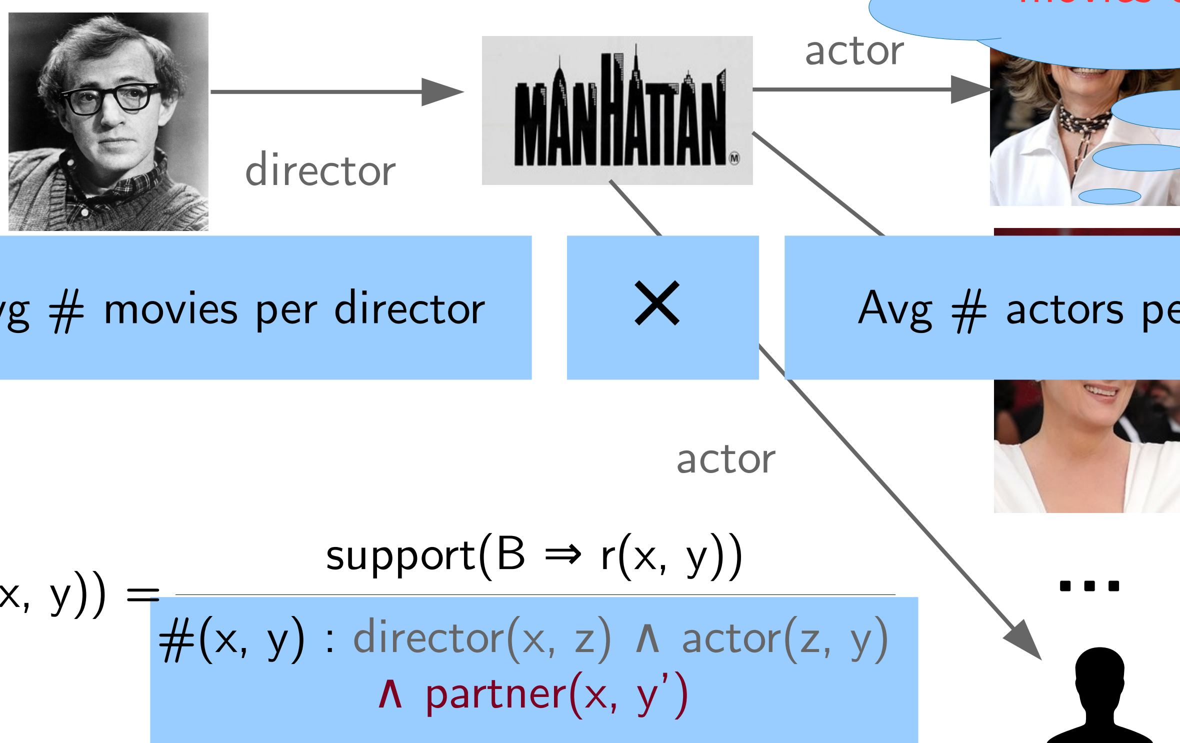
$$\text{conf}_{\text{pca}}(B \Rightarrow r(x, y)) = \frac{\text{support}(B \Rightarrow r(x, y))}{\#(x, y) : \text{director}(x, z) \wedge \text{actor}(z, y) \wedge \text{partner}(x, y')}$$

Confidence approximation

- Use statistics to estimate conf_{pca} denominator
  - (# of actors per director) × (# of partnered directors)

director(x, z), actor(z, y) ⇒

Actors can play in several movies of the same director



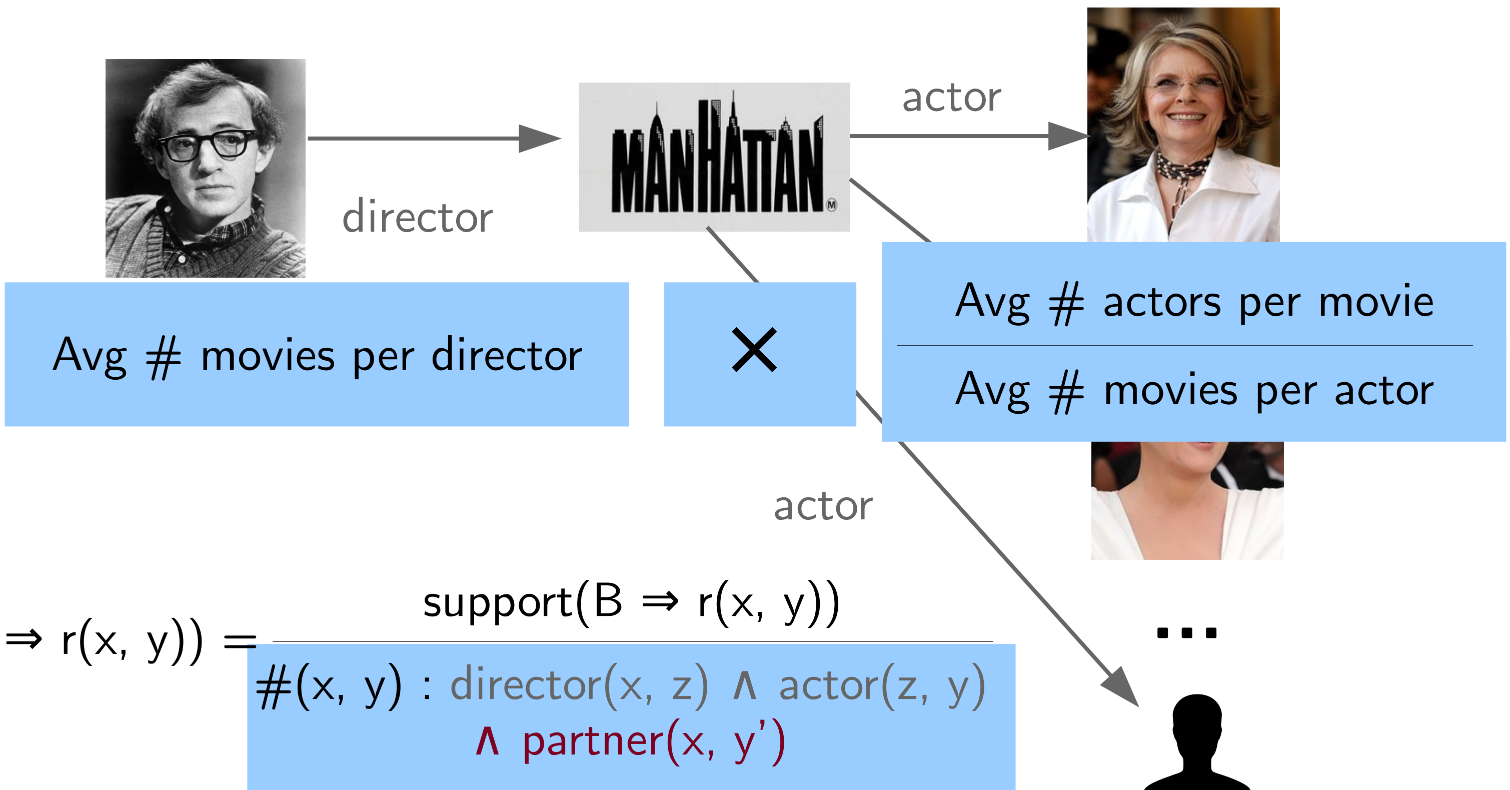
Avg # movies per director × Avg # actors per movie

$$\text{conf}_{pca}(B \Rightarrow r(x, y)) = \frac{\text{support}(B \Rightarrow r(x, y))}{\#(x, y) : \text{director}(x, z) \wedge \text{actor}(z, y) \wedge \text{partner}(x, y')}$$

Confidence approximation

- Use statistics to estimate conf_{pca} denominator
  - (# of actors per director) × (# of partnered directors)

$$\text{director}(x, z), \text{actor}(z, y) \Rightarrow \text{partner}(x, y)$$

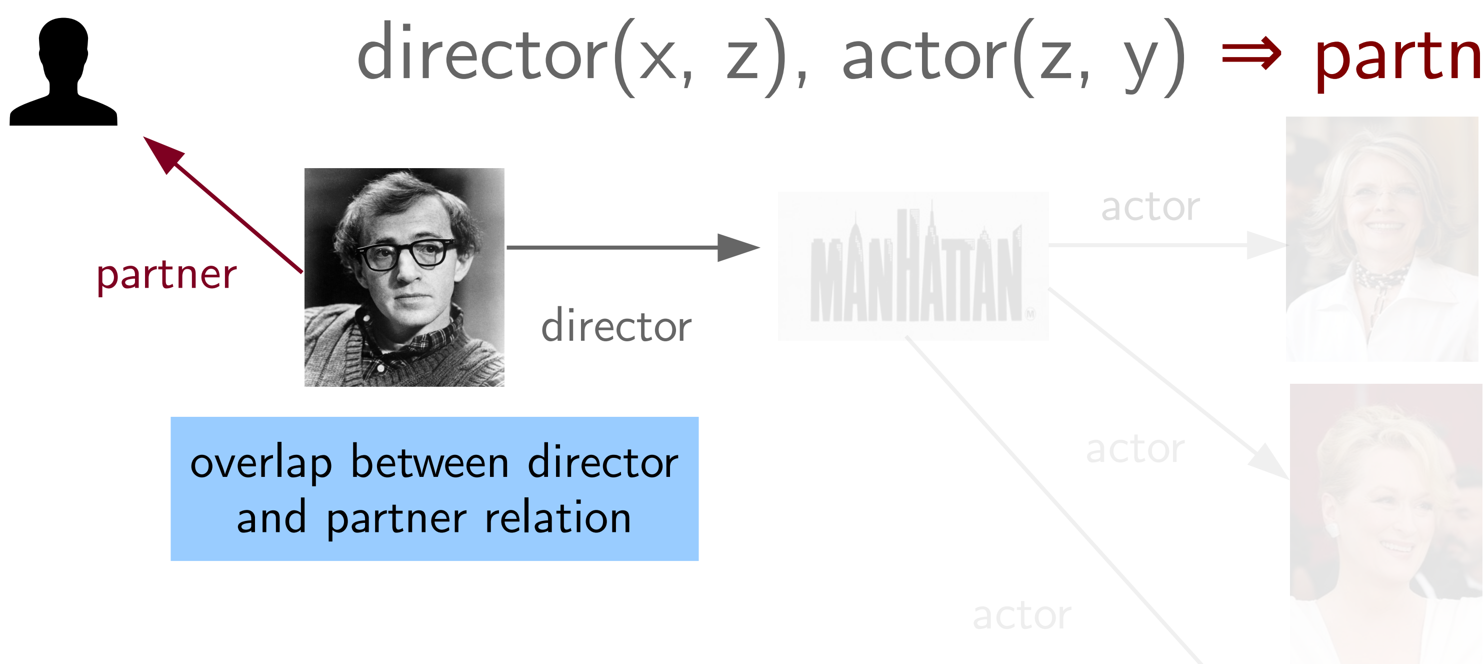


$$\text{conf}_{pca}(B \Rightarrow r(x, y)) = \frac{\text{support}(B \Rightarrow r(x, y))}{\#(x, y) : \text{director}(x, z) \land \text{actor}(z, y) \land \text{partner}(x, y')}$$

Confidence approximation

- Use statistics to estimate conf_{pca} denominator
  - (# of actors per director) × (# of partnered directors)

$$\text{director}(x, z), \text{actor}(z, y) \Rightarrow \text{partner}(x, y)$$

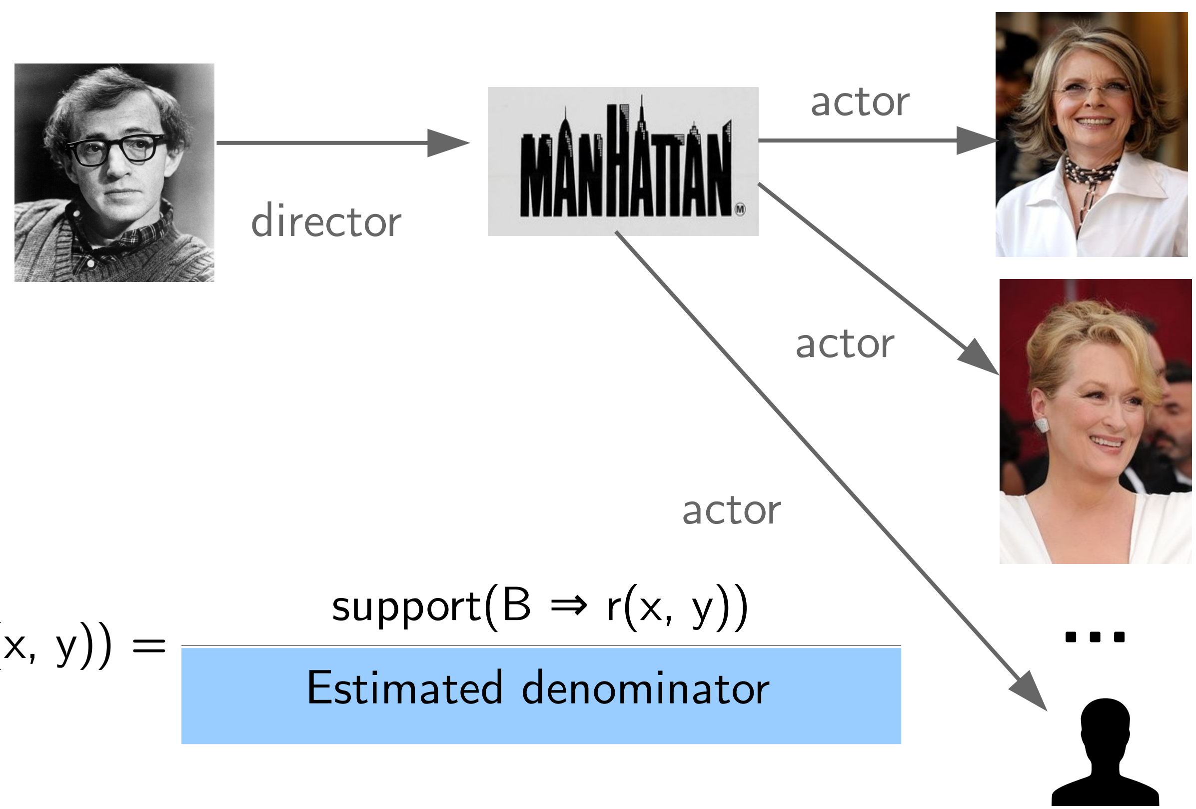


$$\text{conf}_{\text{pca}}(B \Rightarrow r(x, y)) = \frac{\text{support}(B \Rightarrow r(x, y))}{\#(x, y) : \text{director}(x, z) \wedge \text{actor}(z, y) \wedge \text{partner}(x, y')}$$

Confidence approximation

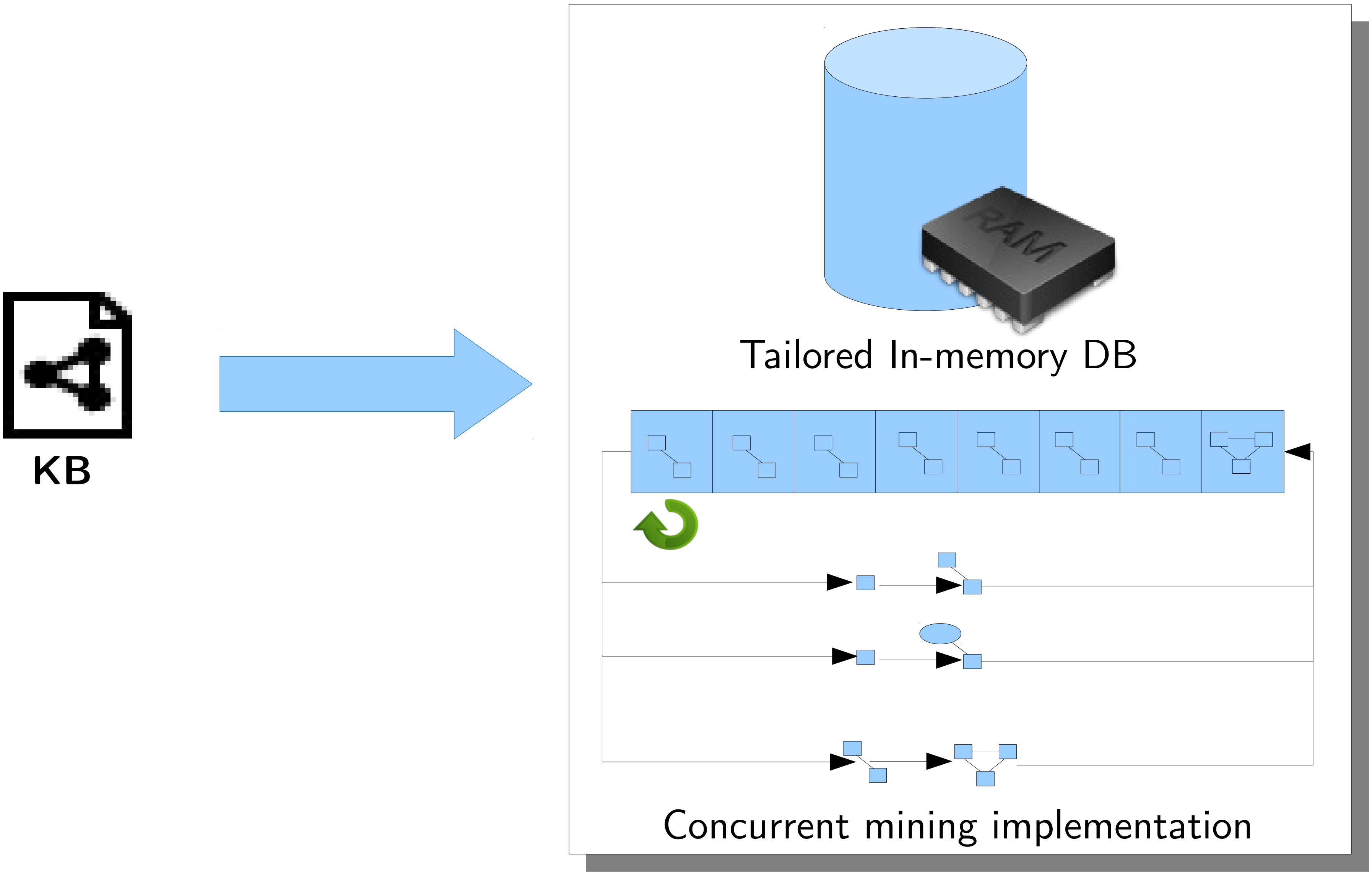
- If estimation is below threshold, discard the rule.
  - (# of actors per director) × (# of partnered directors)

$$\text{director}(x, z), \text{actor}(z, y) \Rightarrow \text{partner}(x, y)$$



$$\text{conf}_{\text{pca}}(B \Rightarrow r(x, y)) = \frac{\text{support}(B \Rightarrow r(x, y))}{\text{Estimated denominator}}$$

AMIE: Association Rule Mining Under Incomplete Evidence



AMIE's runtime

AMIE is 3 order of magnitude faster than state-of-the-art approaches.

Dataset	# facts	WARMR	ALEPH	AMIE
YAGO core	1M	-	5s to 1d	3.17min
YAGO (sample)	47K	18h	0.05s to 1d	2.59s, 2.90s

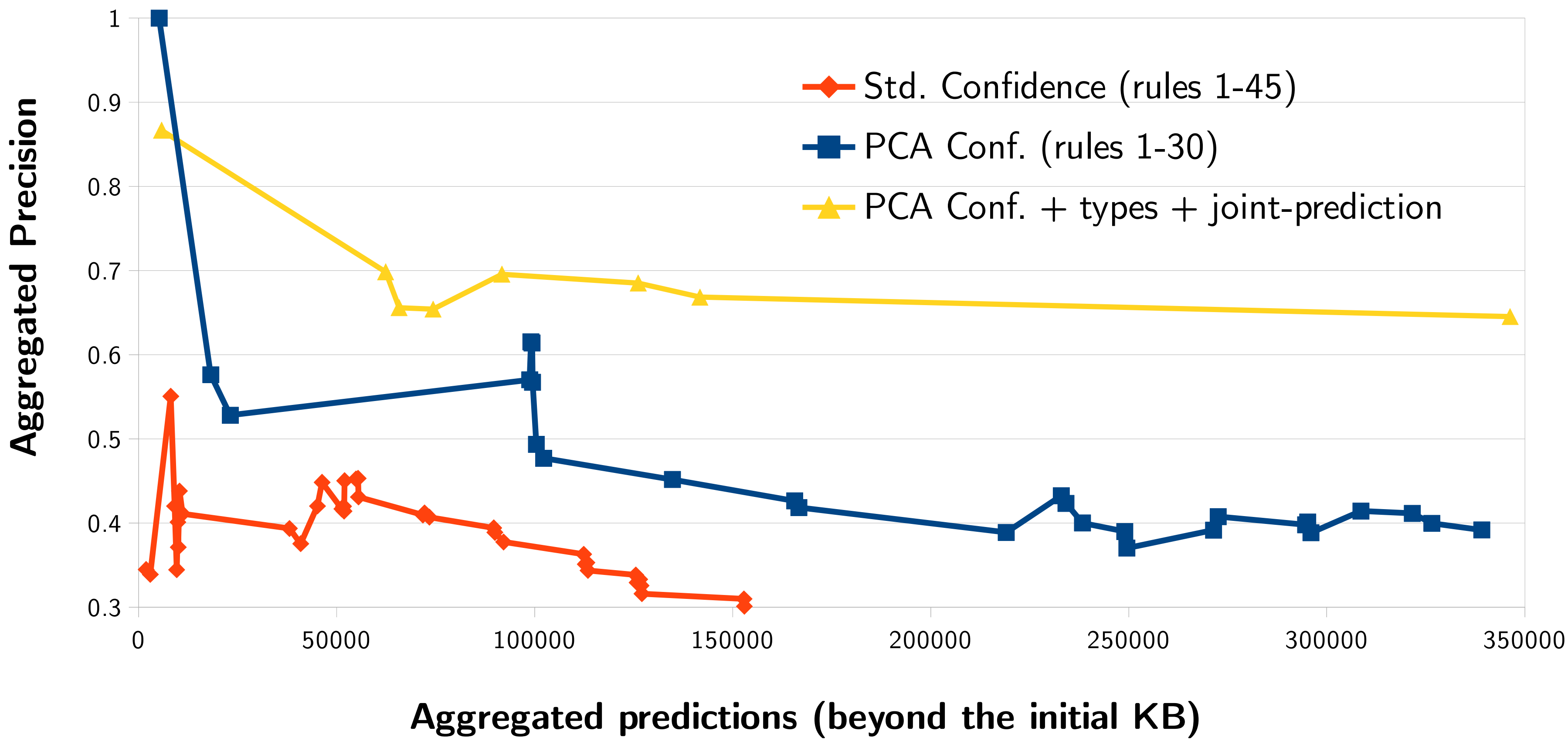
AMIE's runtime

AMIE can mine rules in large ontologies up to 11M facts and more than 1500 relations.

Dataset	Facts	Relations	Runtime	Rules
YAGO2	948K	32	28.19s	138
YAGO2 (const)			9.93min	18K
YAGO2 (l=4)			8.35min	6.9K
YAGO2s	4.12M	37	59.38min	94
Dbpedia 2.0	6.7M	1595	46.88min	113K
Dbpedia 3.8	11.02M	650	7h 6min	2.47K
Wikidata	8.4M	431	25.50min	889

AMIE's output quality

PCA confidence suitable at ranking predictive rules.



Some rules found by AMIE

- YAGO
  - hasWonPrize(x, Leibniz Prize) $\Rightarrow$ livesIn(x, Germany)
  - hasAdvisor(x, y), graduatedFrom(x, z) $\Rightarrow$ worksAt(y, z)
- DBpedia
  - countySeat(x, y) $\Rightarrow$ largestCity(x, y)
- Wikidata
  - relative(y, z), sister(z, x) $\Rightarrow$ relative(x, y)

Summary

- Pruning strategies in combination with custom DB implementation allow for scalable rule mining
- PCA more suitable at generating counter-evidence

Summary

- Pruning strategies in combination with custom DB implementation allow for scalable rule mining
- PCA more suitable at generating counter-evidence

Luis Galárraga, Christina Teflioudi, Katja Hose, Fabian Suchanek.
AMIE: Association Rule Mining Under Incomplete Evidence in Ontological Knowledge Bases.
WWW, 2013. **Best student paper award**.

Luis Galárraga, Christina Teflioudi, Katja Hose, Fabian Suchanek.
Fast Rule Mining in Ontological Knowledge Bases with AMIE+.
VLDB Journal.




max planck institut
informatik

Applications of Rule Mining

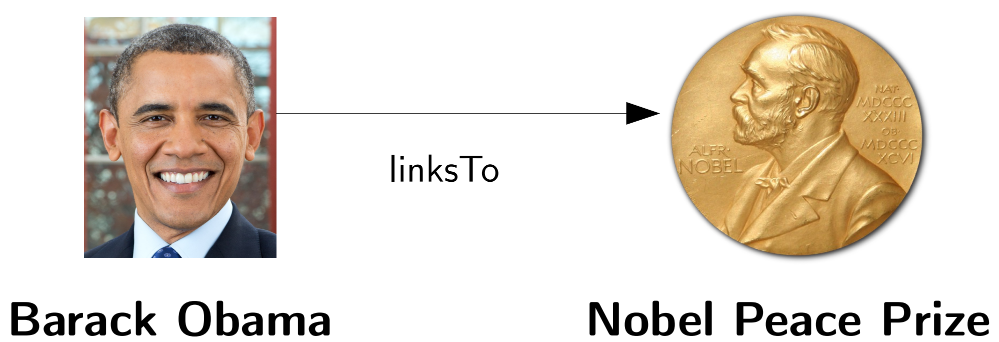
Semantifying wikilinks

Semantifying wikilinks

- KBs store the hyperlinks structure of Wikipedia articles

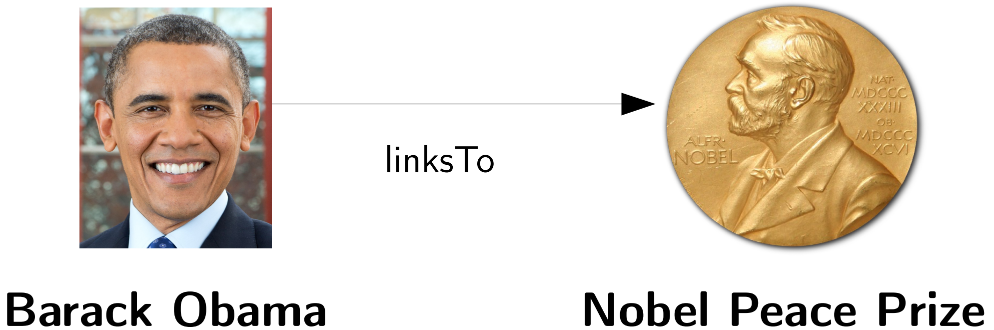
Semantifying wikilinks

- KBs store the hyperlinks structure of Wikipedia articles



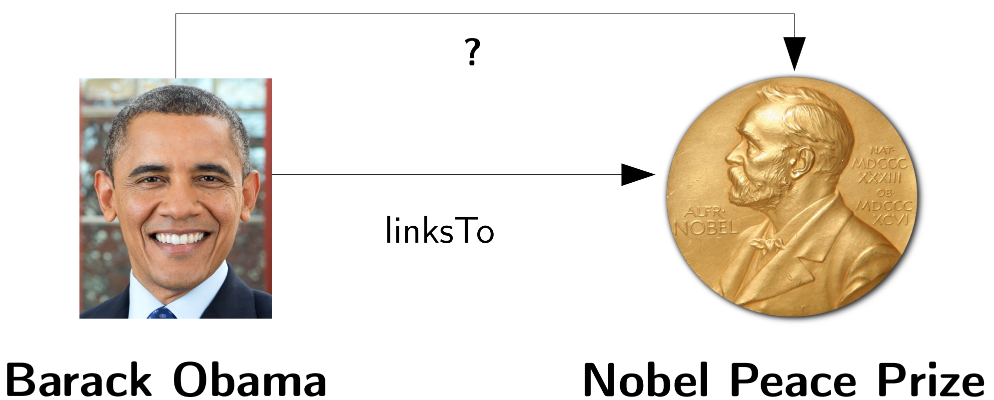
Semantifying wikilinks

- KBs store the hyperlinks structure of Wikipedia articles
- Usually the semantics of the relation are unknown



Semantifying wikilinks

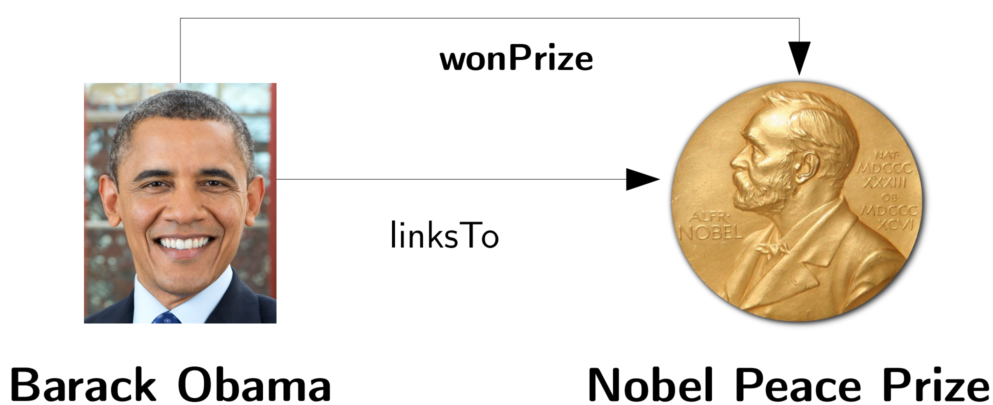
- KBs store the hyperlinks structure of Wikipedia articles
- Usually the semantics of the relation are unknown
  - These are the unsemantified wikilinks



Semantifying wikilinks

- KBs store the hyperlinks structure of Wikipedia

Goal: Find the relations that hold between the endpoints of wikilinks.

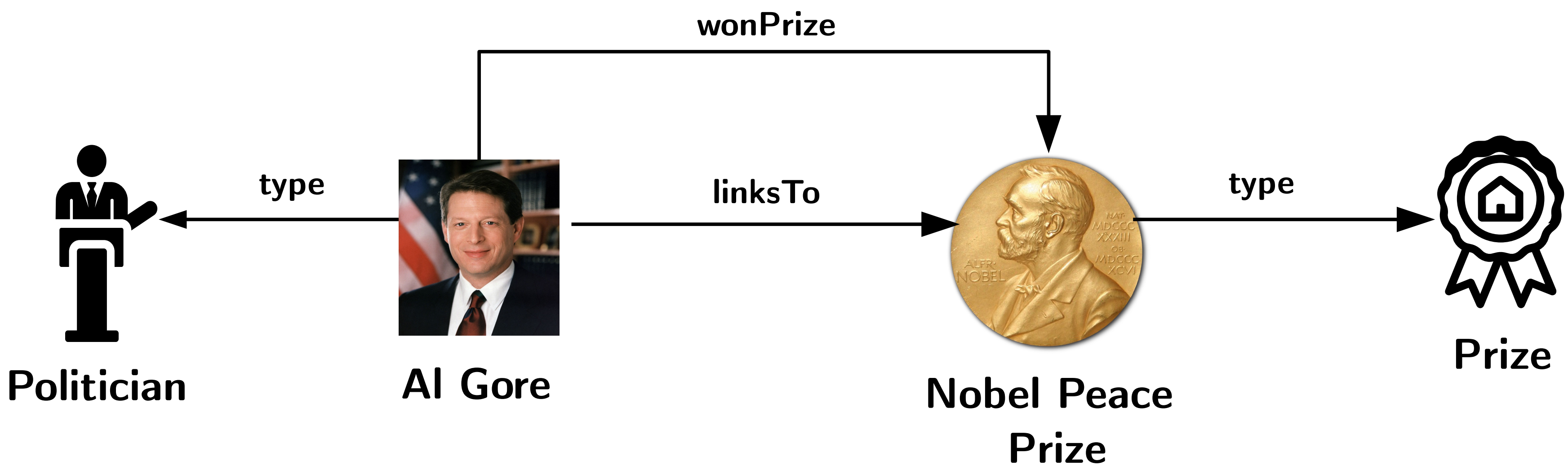


Approach

Use semantified wikilinks to learn rules.

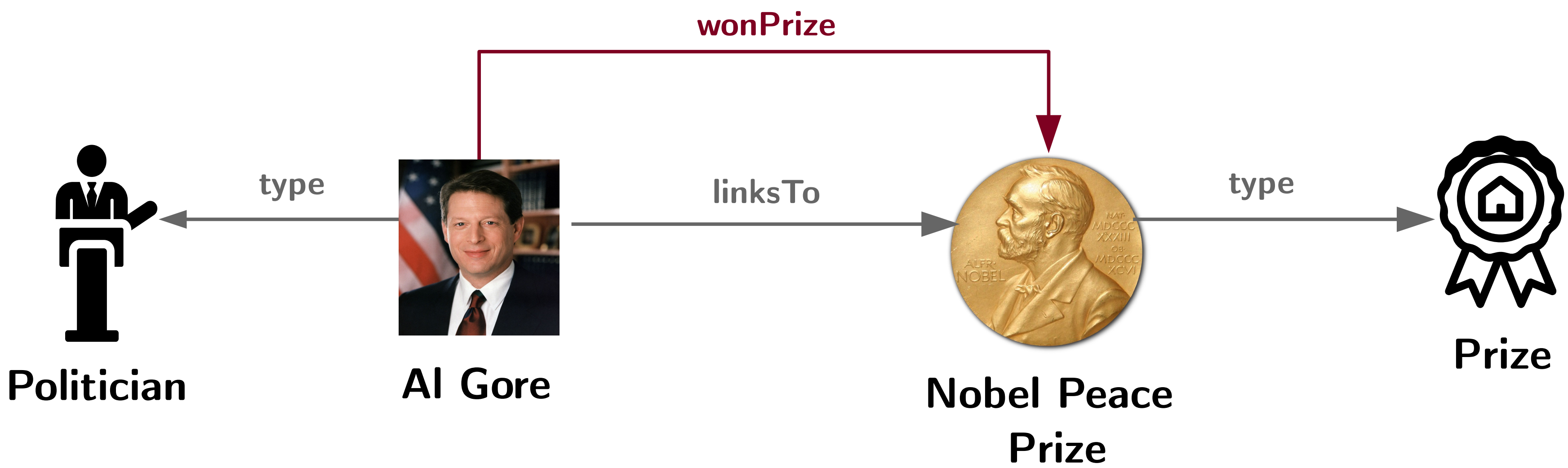
Approach

Use semantified wikilinks to learn rules.



Approach

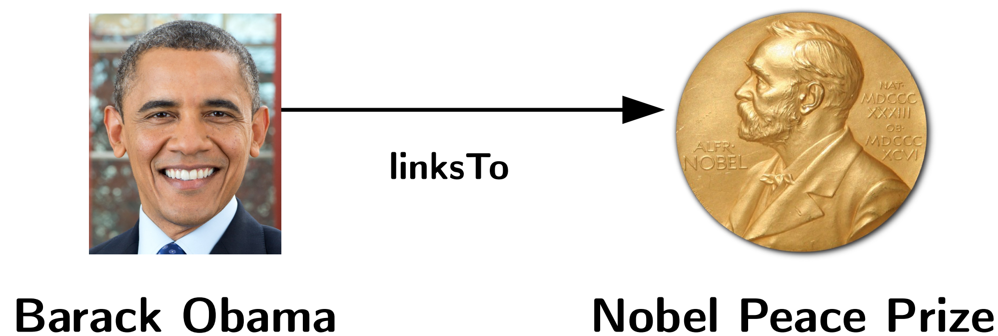
Use semantified wikilinks to learn rules.



linksTo(x, y), type(x, Politician), type(y, Prize) $\Rightarrow$ wonPrize(x, y)

Approach

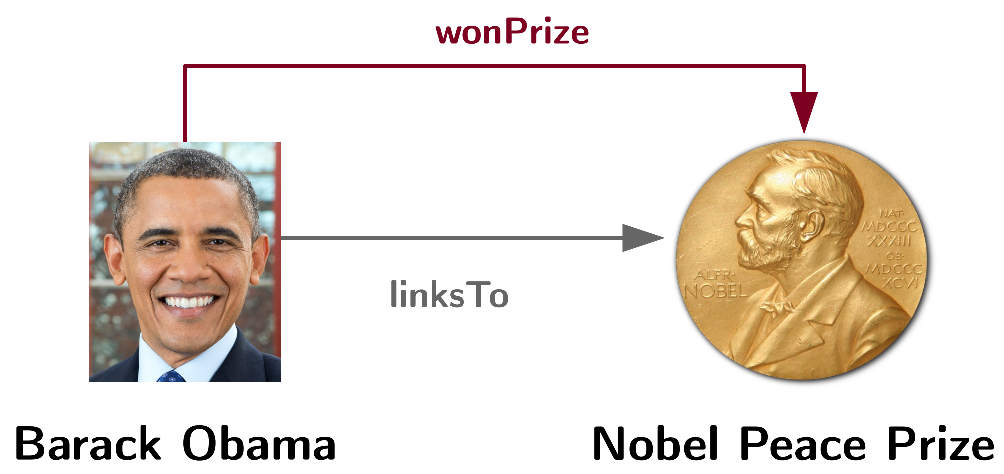
Apply rules to predict the meaning of wikilinks



linksTo(x, y), type(x, Politician), type(y, Prize) $\Rightarrow$ wonPrize(x, y)

Approach

Apply rules to predict the meaning of wikilinks



linksTo(x, y), type(x, Politician), type(y, Prize) ⇒ wonPrize(x, y)

Experimental evaluation

Wikilinks semantification

- Experiments on DBpedia 3.8

Wikilinks semantification

- Experiments on DBpedia 3.8
  - 2M+ wikilinks, 18M facts in total

Wikilinks semantification

- Experiments on DBpedia 3.8
  - 2M+ wikilinks, 18M facts in total
- AMIE for rule mining

Wikilinks semantification

- Experiments on DBpedia 3.8
  - 2M+ wikilinks, 18M facts in total
- AMIE for rule mining
  - 3500+ semantification rules

Wikilinks semantification

- Experiments on DBpedia 3.8
  - 2M+ wikilinks, 18M facts in total
- AMIE for rule mining
  - 3500+ semantification rules
- 180K unsemantified wikilinks

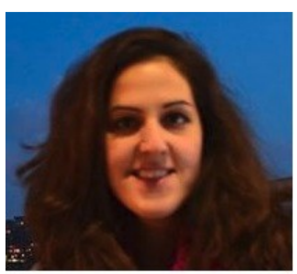
Wikilinks semantification

- Experiments on DBpedia 3.8
  - 2M+ wikilinks, 18M facts in total
- AMIE for rule mining
  - 3500+ semantification rules
- 180K unsemantified wikilinks
  - For each wikilink we generated a ranking of possible relations.
  - 77% precision @top1, 67% @top3

Summary: Wikilinks

Rule Mining is an effective method for the semantification of wikilinks

Luis Galárraga, Danai Symeonidou, Jean-Claude Moissinac.
Rule Mining for Semantifying Wikilinks.
In LOWD, 2015

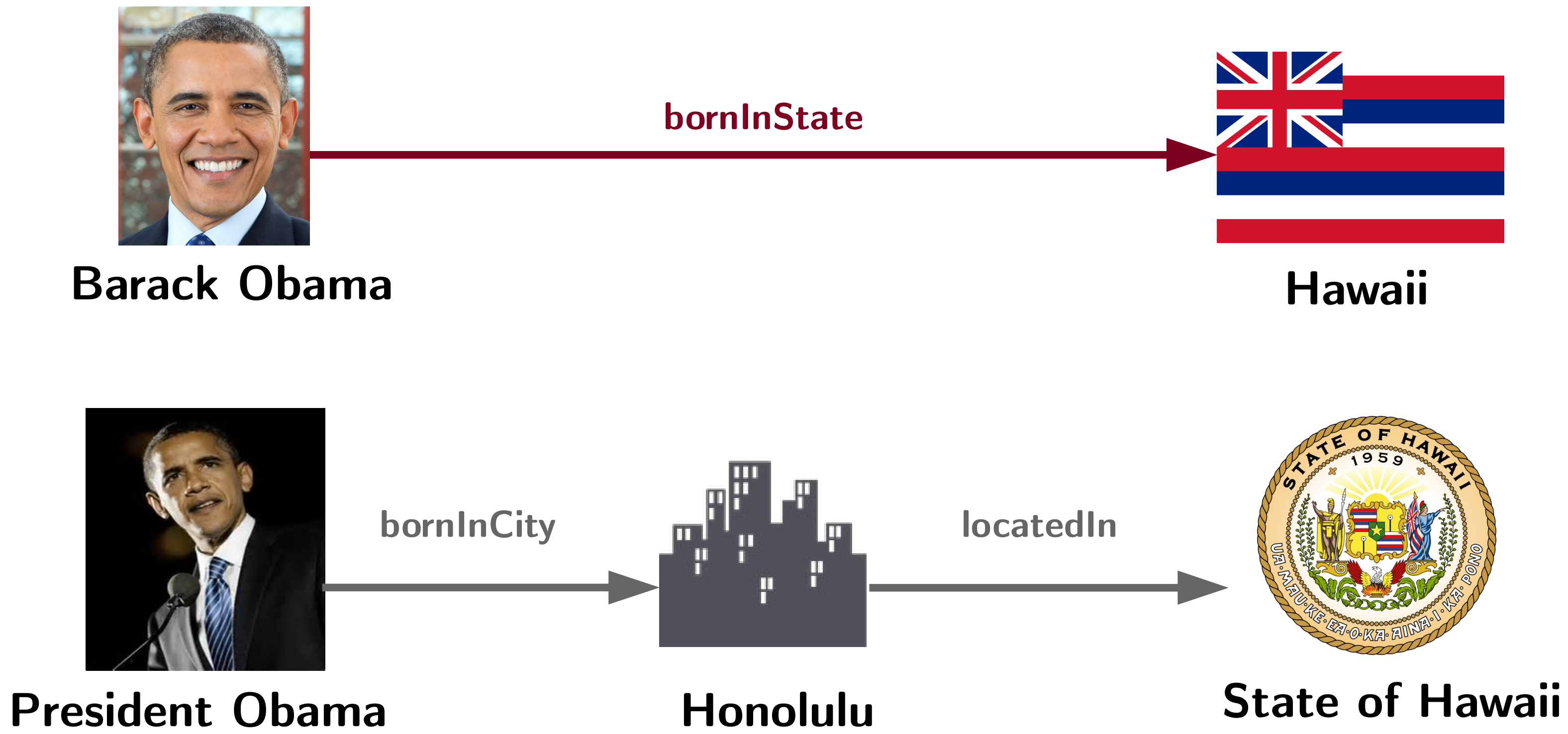




TELECOM ParisTech
Institut
Mines-Télécom

Schema Alignment

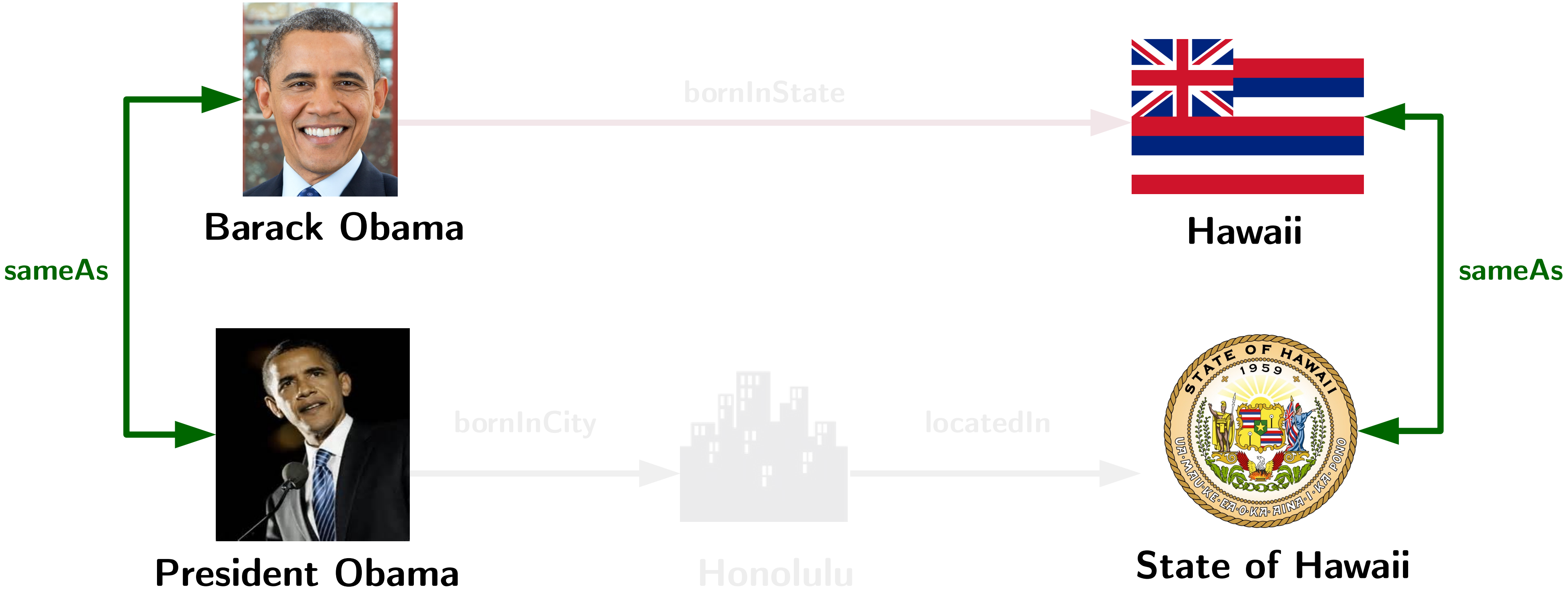
Schema Alignment

KBs in the semantic web speak in different "languages" about the same things.



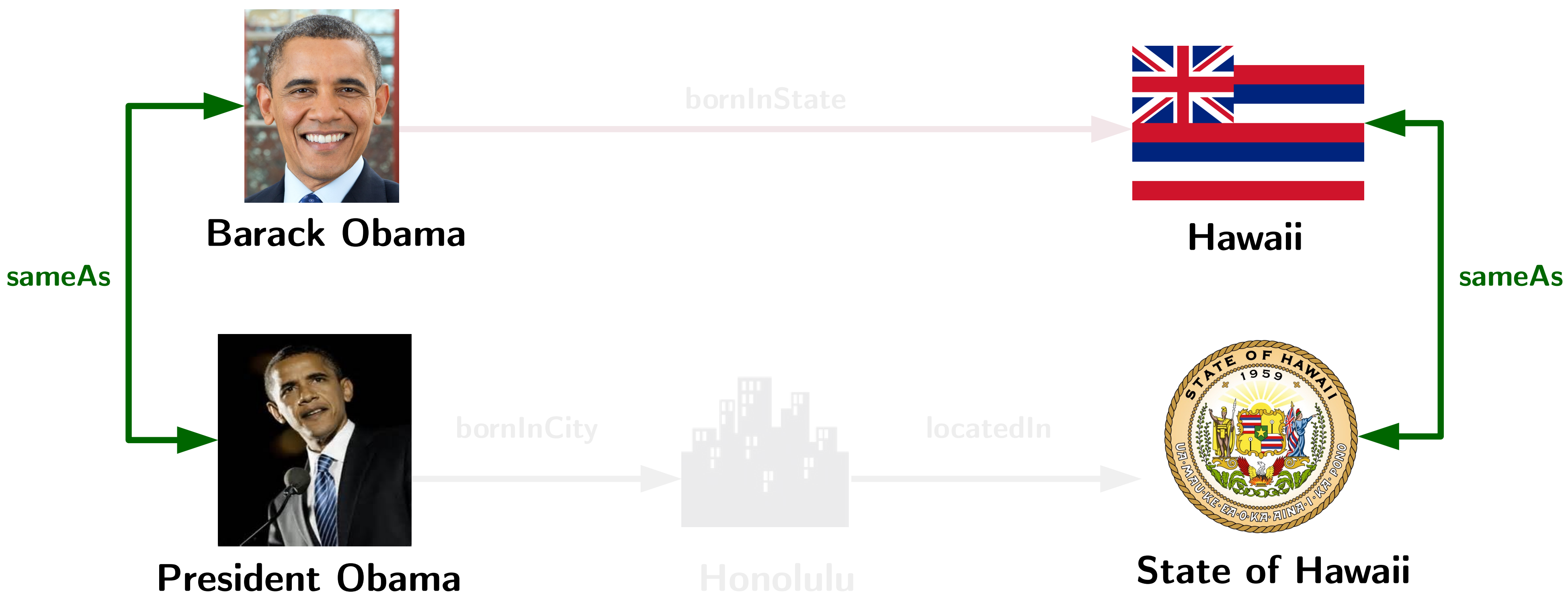
Schema Alignment

Some instances has been aligned.



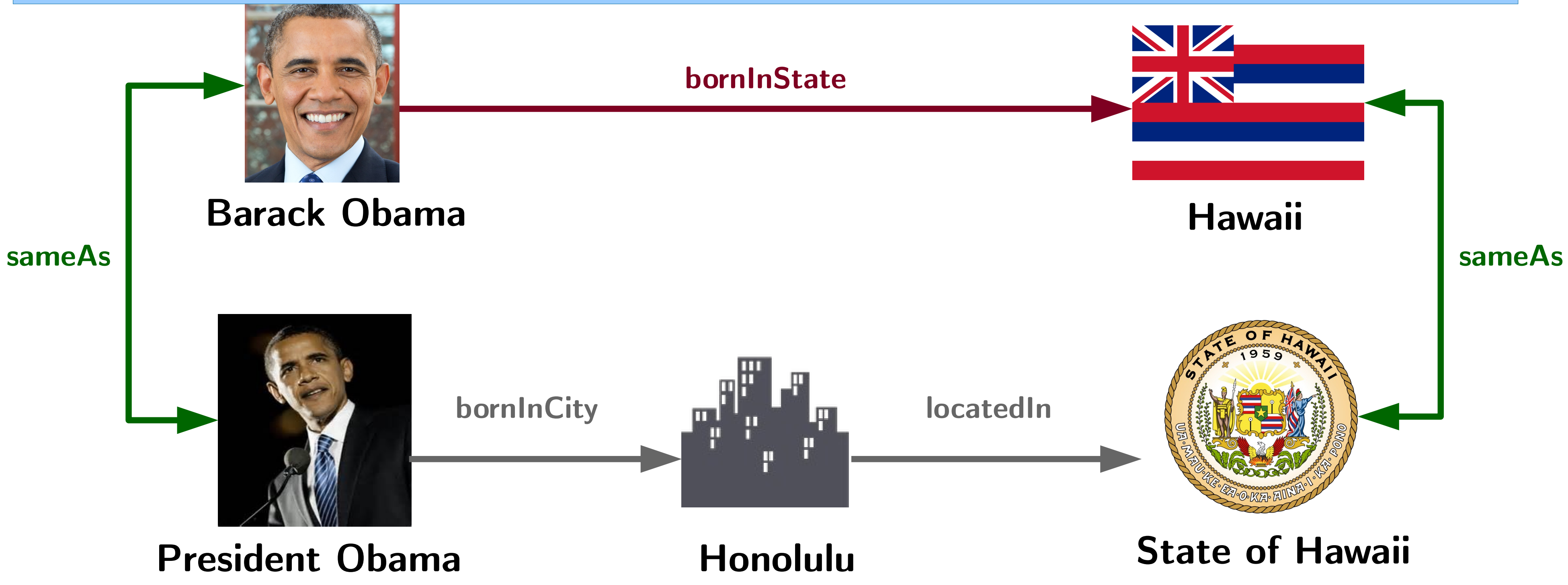
Schema Alignment

This does not suffice for a full data integration



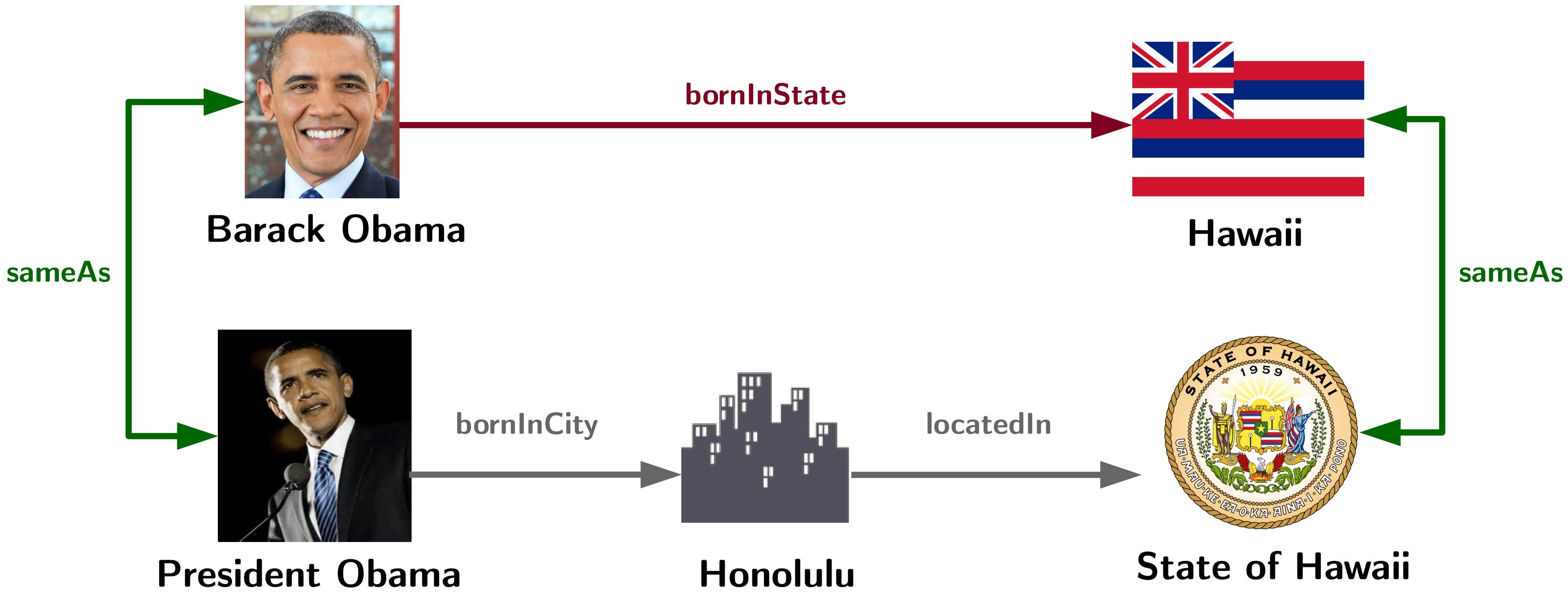
Schema Alignment

Goal: Find schema alignments between two KBs



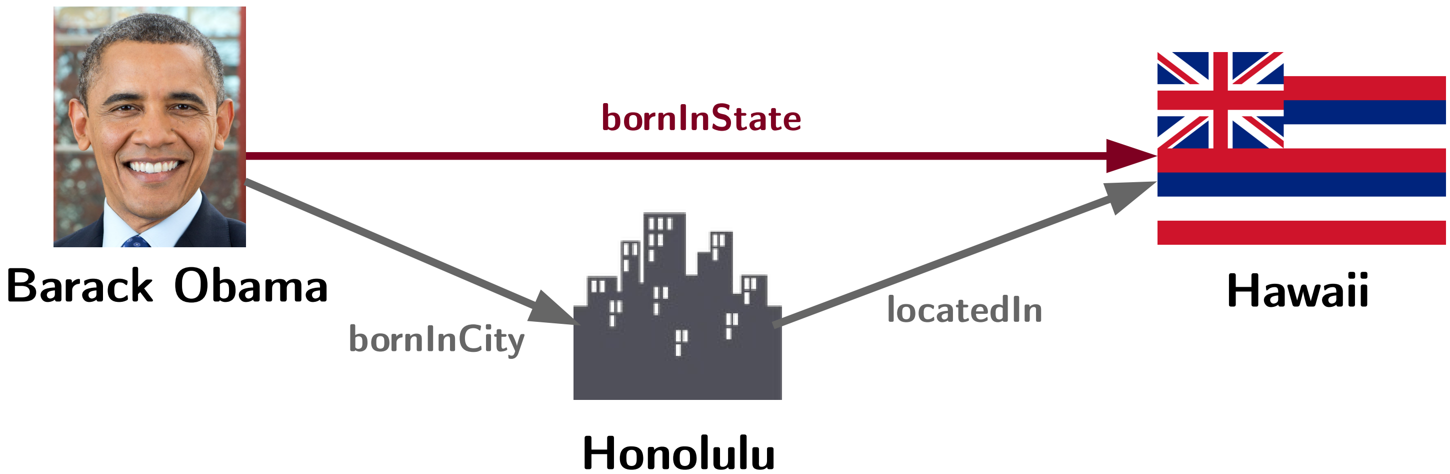
Approach

Use instance alignments to “coaleasce” the KBs.



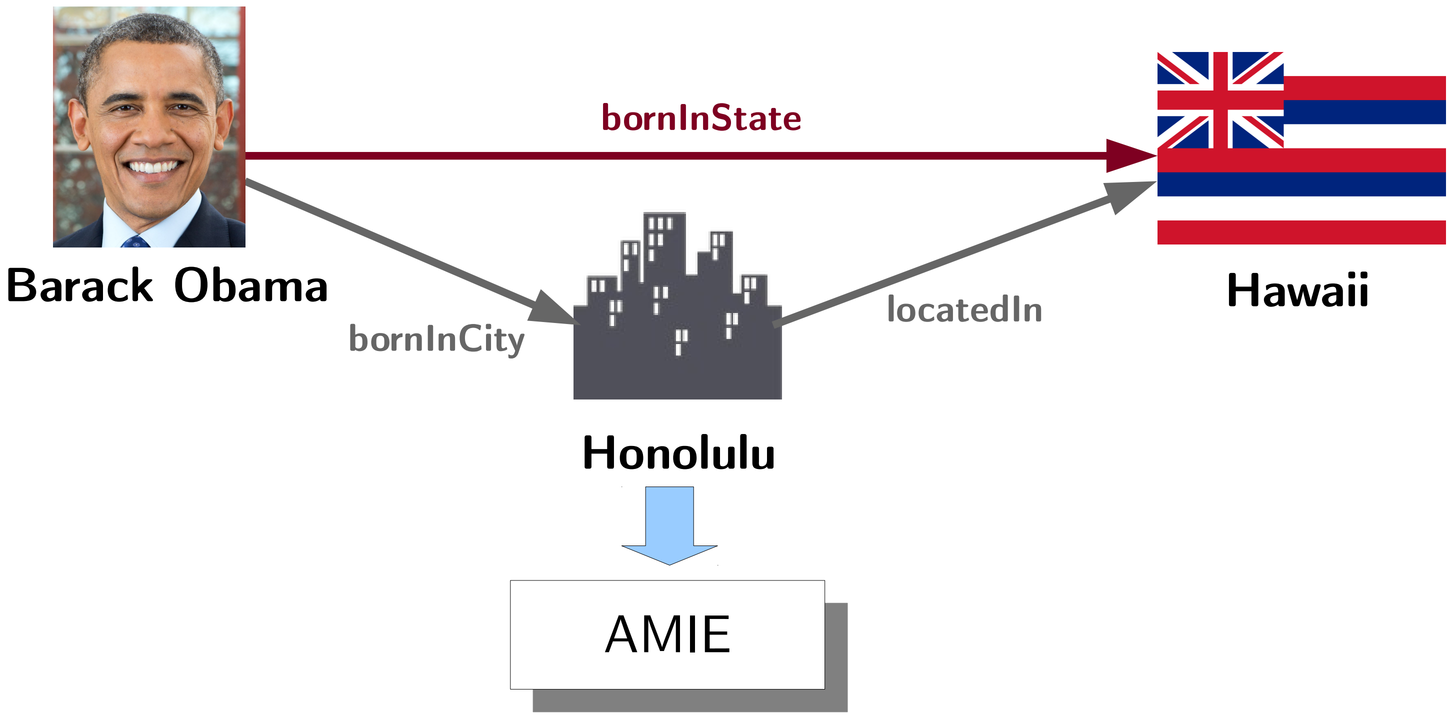
Approach

Use instance alignments to “coaleasce” the KBs.



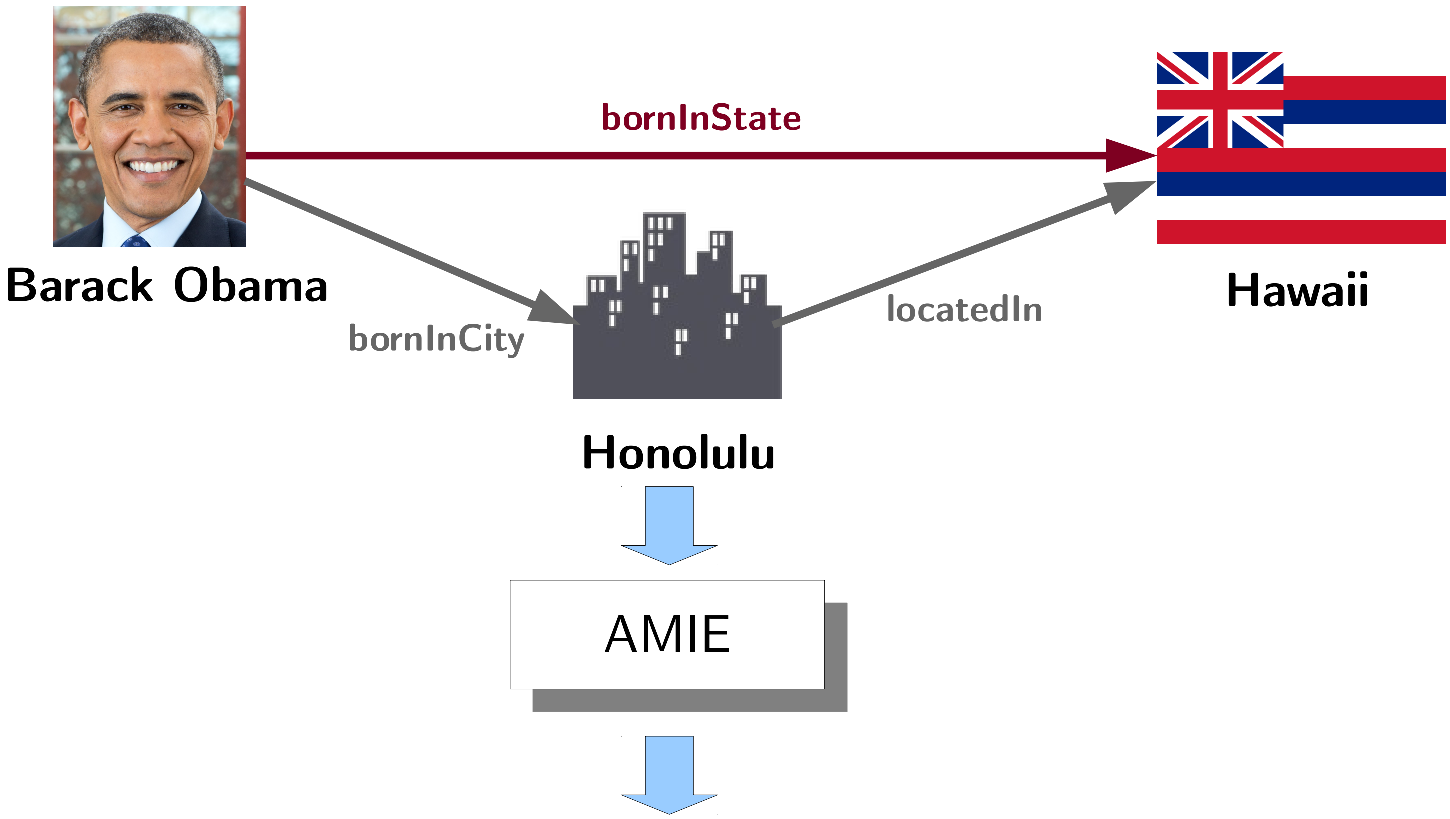
Approach

Mine alignment rules on the coalesced KB



Approach

Mine alignment rules on the coalesced KB



bornInCity(x, z), locatedIn(y, z) $\Rightarrow$ bornInState(x, y)

ROSA rules

d:artist(x, y) ⇒ y:created(x, y)	R-subsumption
d:nationality(x, y) ⇔ y:citizenOf(x, y)	R-equivalence
type(x, Athlete_d) ⇒ type(x, Person_y)	C-subsumption
y:bornIn(x, y), y:label(y, z) ⇒ i:bornIn(x, z)	2-hops translation
y:child(x, y), y:child(x, z) ⇒ f:sibling(y, z)	Triangle alignment
y:bornIn(x, y), type(x, City_y) ⇒ f:birthPlace(x, y)	Specific R-subsumption
y:locatedIn(x, Italy) ⇒ d:timeZone(x, CET)	Attribute-Value translation
type(x, Royal_f), f:gender(x, female) ⇒ type(y, Princess)	2-values translation

ROSA rules

d:artist(x, y) ⇒ y:created(x, y)	R-subsumption
d:nationality(x, y) ⇔ y:citizenOf(x, y)	R-equivalence
type(x, Athlete_d) ⇒ type(x, Person_y)	C-subsumption
y:bornIn(x, y), y:label(y, z) ⇒ i:bornIn(x, z)	2-hops translation
y:child(x, y), y:child(x, z) ⇒ f:sibling(y, z)	Triangle alignment
y:bornIn(x, y), type(x, City_y) ⇒ f:birthPlace(x, y)	Specific R-subsumption
y:locatedIn(x, Italy) ⇒ d:timeZone(x, CET)	Attribute-Value translation
type(x, Royal_f), f:gender(x, female) ⇒ type(y, Princess)	2-values translation

Complex alignments suffer from low precision and the presence of soft-rules

ROSA rules

d:artist(x, y) ⇒ y:created(x, y)	R-subsumption
d:nationality(x, y) ⇔ y:citizenOf(x, y)	R-equivalence
type(x, Athlete$_d$) ⇒ type(x, Person$_y$)	C-subsumption
y:bornIn(x, y), y:label(y, z) ⇒ i:bornIn(x, z)	2-hops translation
y:child(x, y), y:child(x, z) ⇒ f:sibling(y, z)	Triangle alignment
y:bornIn(x, y), type(x, City$_y$) ⇒ f:birthPlace(x, y)	Specific R-subsumption
y:locatedIn(x, Italy) ⇒ d:timeZone(x, CET)	Attribute-Value translation
type(x, Royal$_f$), f:gender(x, female) ⇒ type(y, Princess)	2-values translation

Luis Galárraga, Nicoleta Preda, Fabian Suchanek.
Mining Rules to Align Knowledge Bases.
AKBC 2013.

Canonicalizing Open Knowledge Bases

Open Knowledge Bases

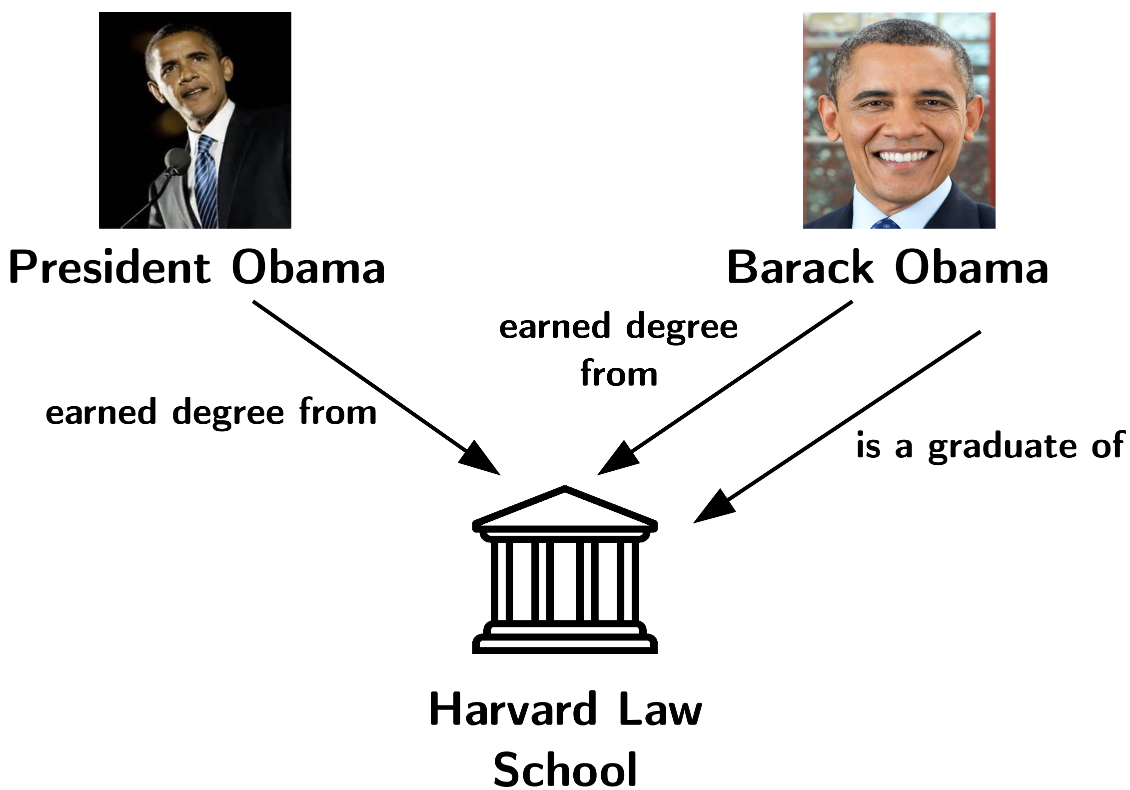
- Normally extracted from text

Open Knowledge Bases

- Normally extracted from text
  - Entities and relations are not **canonical**

Open Knowledge Bases

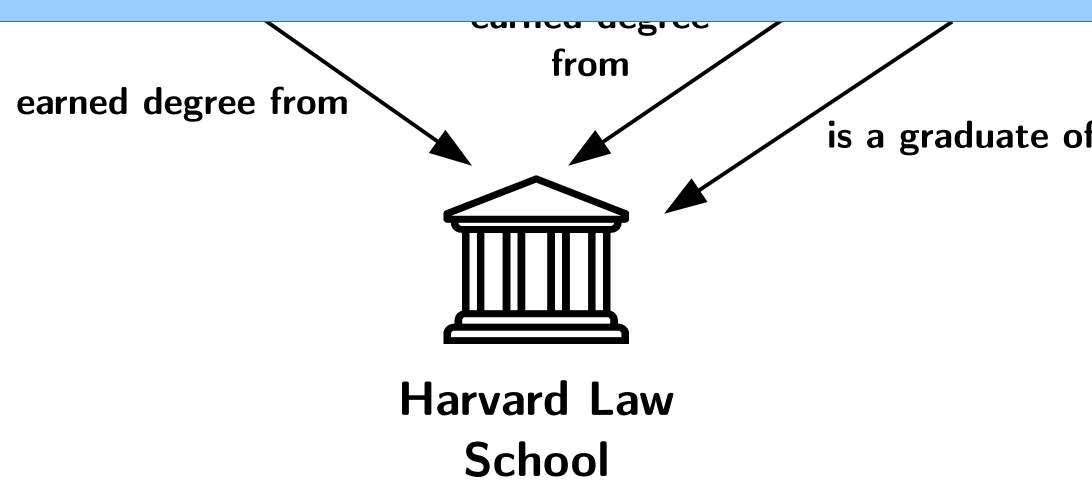
- Normally extracted from text
  - Entities and relations are not **canonical**



Open Knowledge Bases

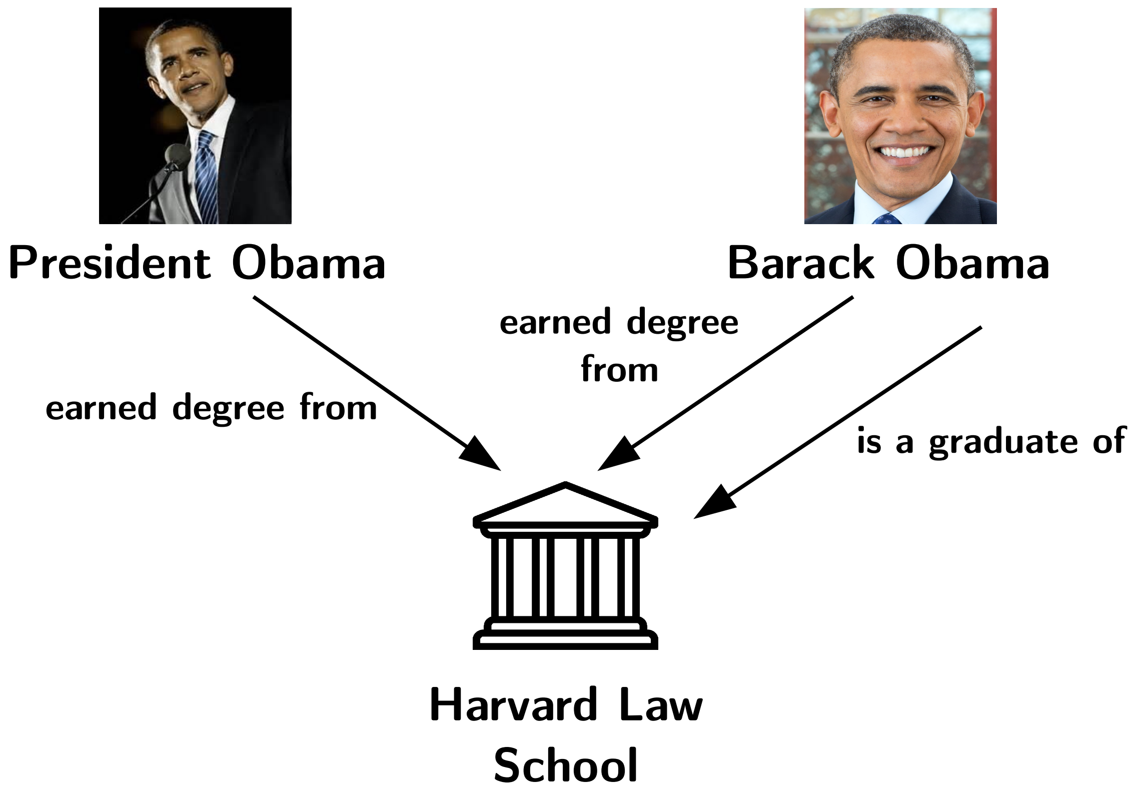
- Normally extracted from text
  - Entities and relations are not **canonical**

Goal: Take an open KB and rewrite its entities and relations in a canonical form



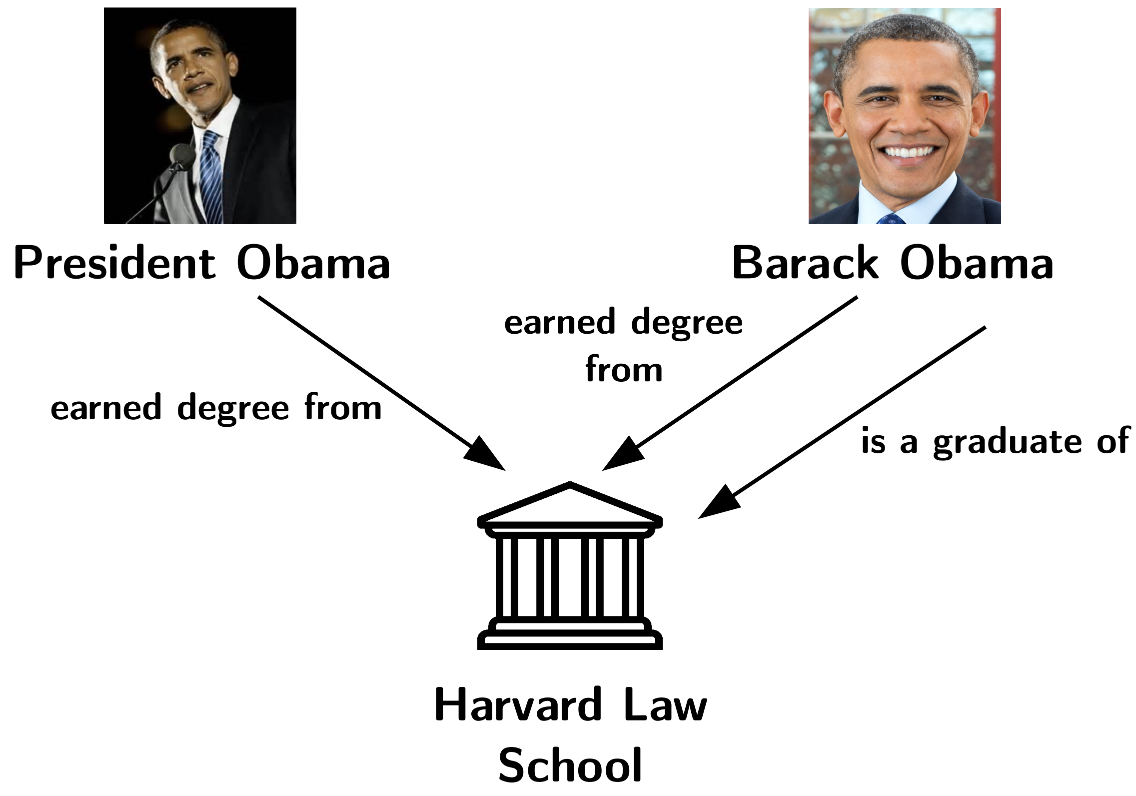
Approach

- Two stages process



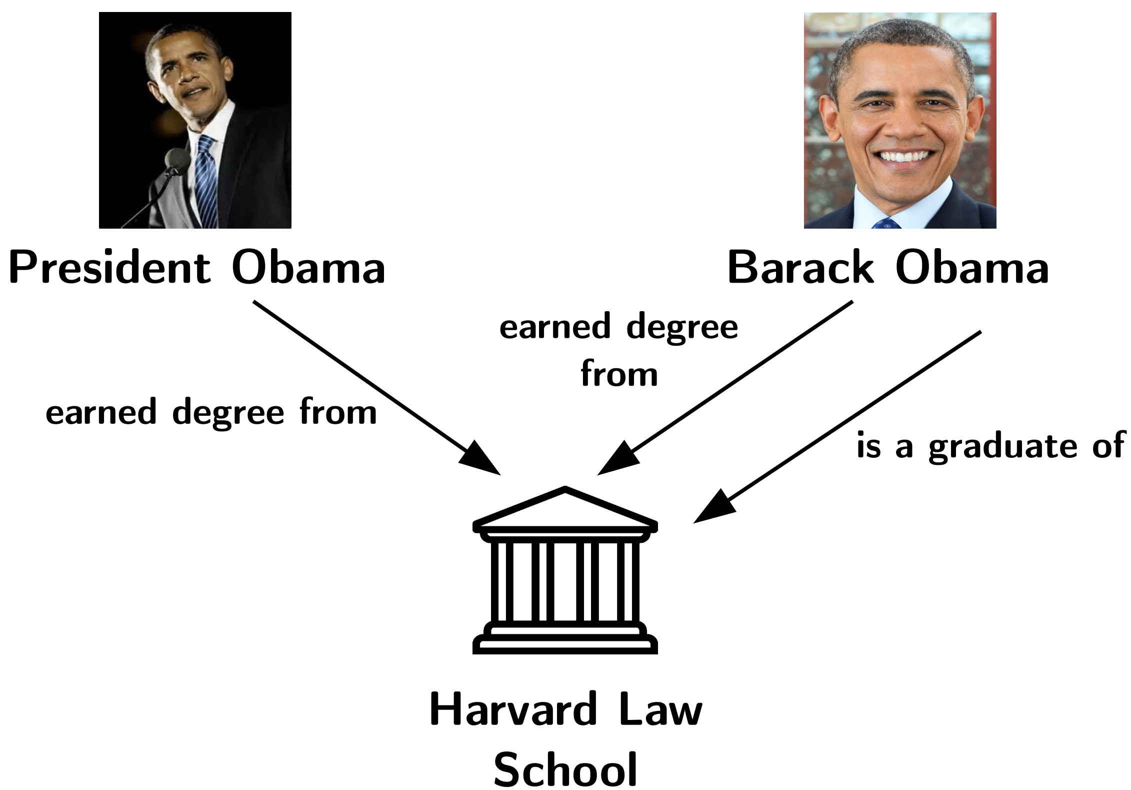
Approach

- Two stages process
  - Canonicalize the noun phrases, then the verbal phrases



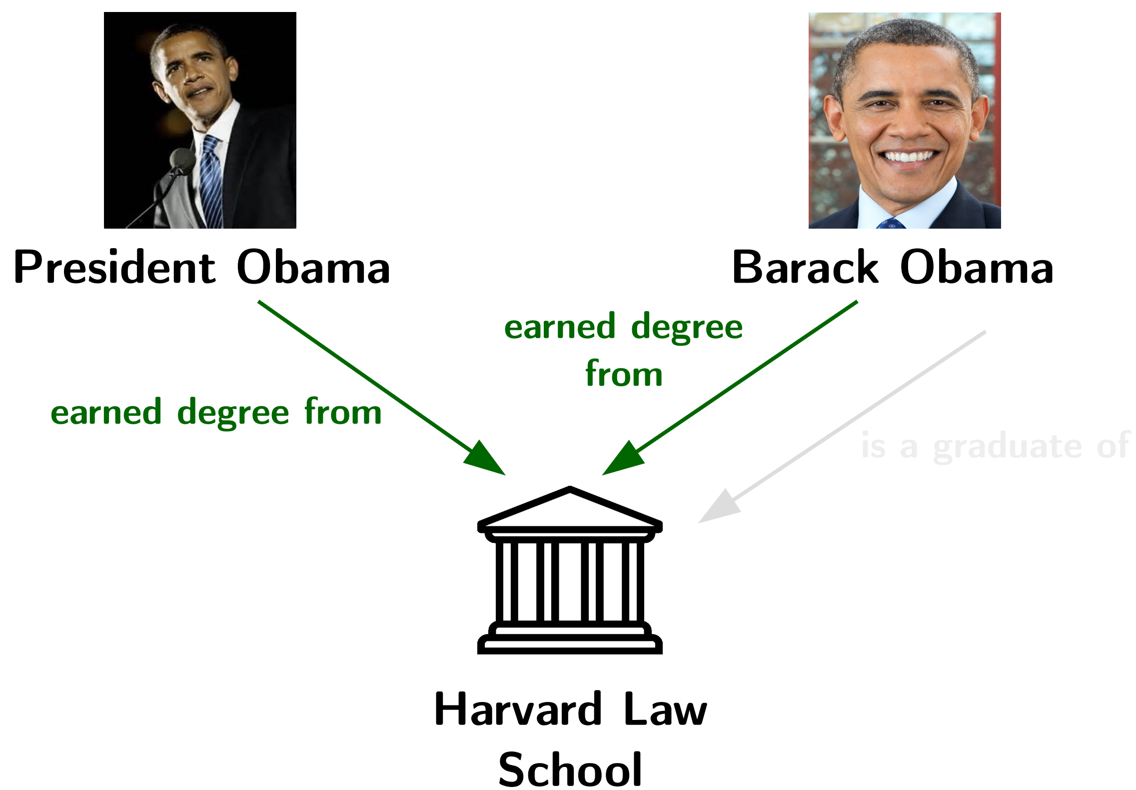
Noun phrase canonicalization

- Use a group of signals of synonymy



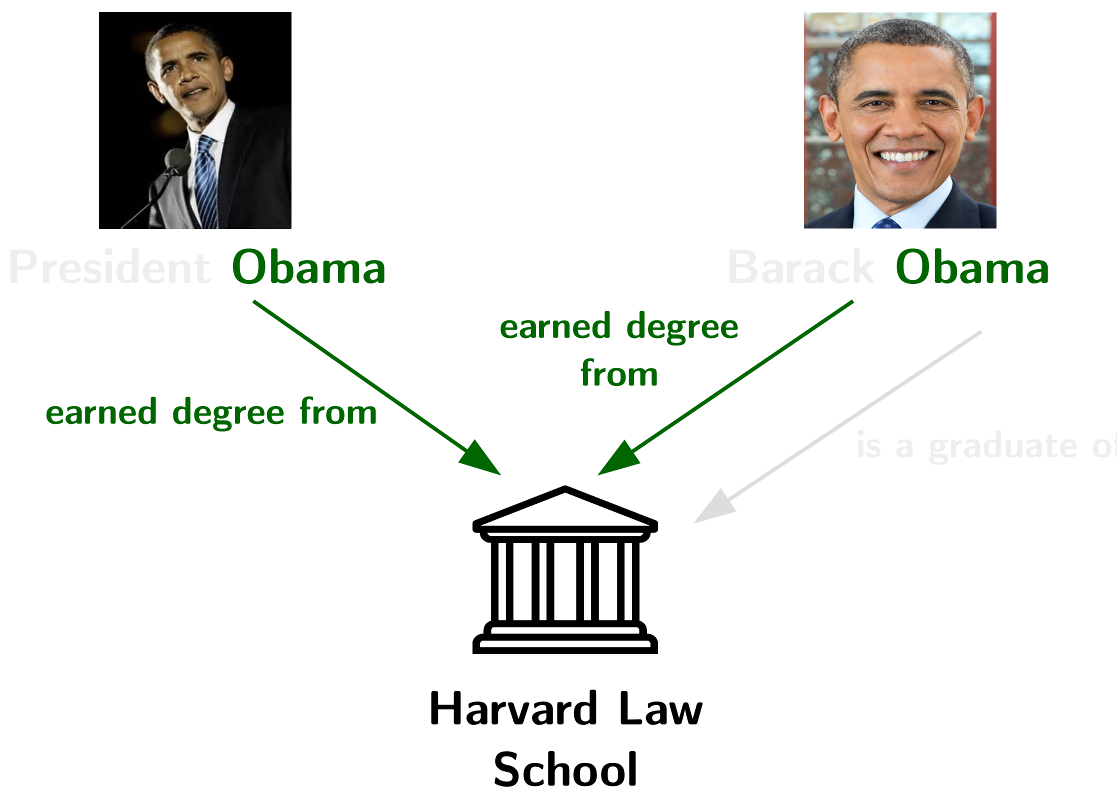
Noun phrase canonicalization

- Use a group of signals of synonymy
  - Example: attributes overlap



Noun phrase canonicalization

- Use a group of signals of synonymy
  - Example: attributes overlap, tokens overlap

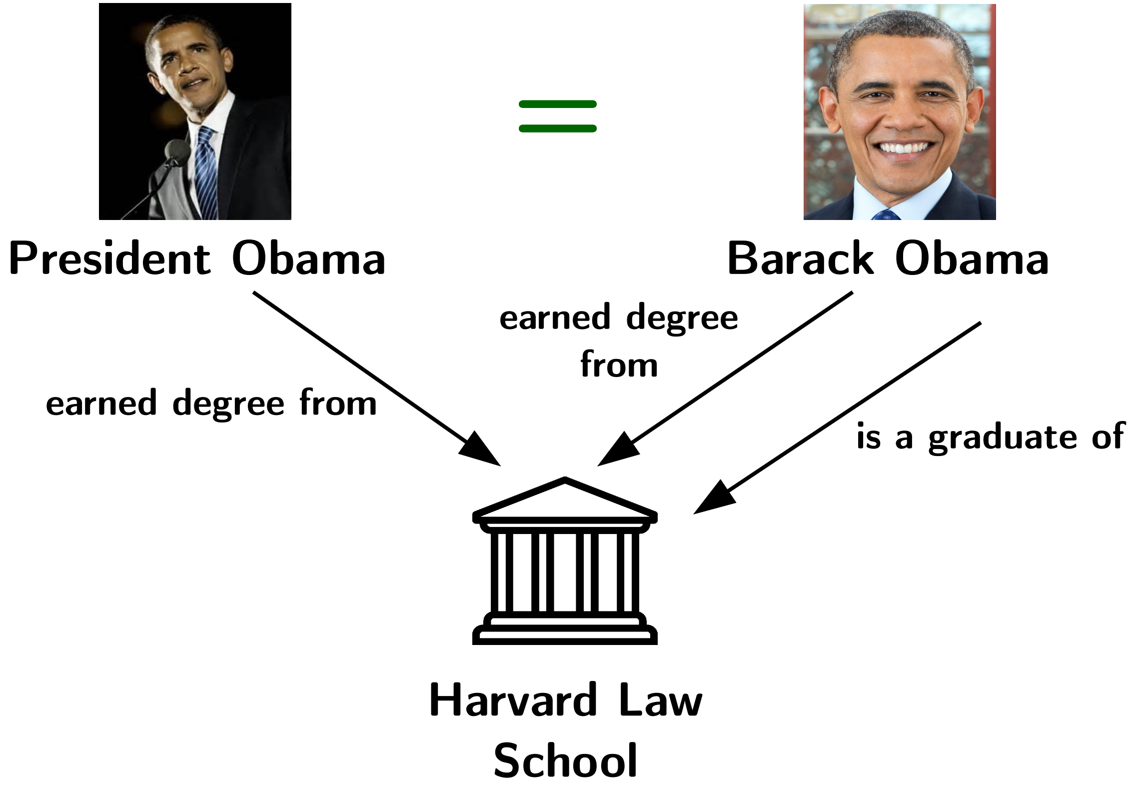


Noun phrase canonicalization

- Simple signals
  - Attribute overlap
  - String equality or similarity
  - IDF Tokens Overlap
- Source signals
  - Words overlap
  - Entities overlap
  - Types overlap
- Combined signal

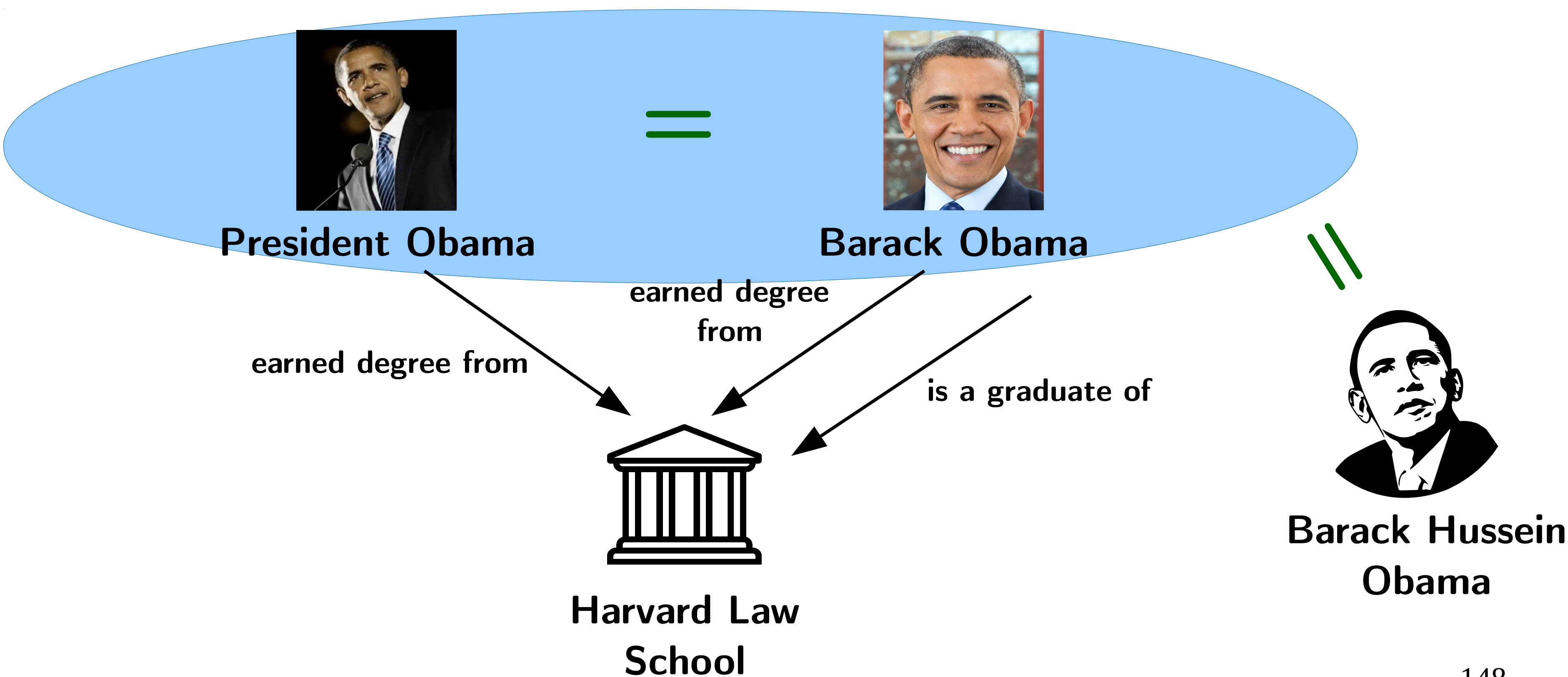
Noun phrase canonicalization

- Use signals to cluster synonym noun phrases



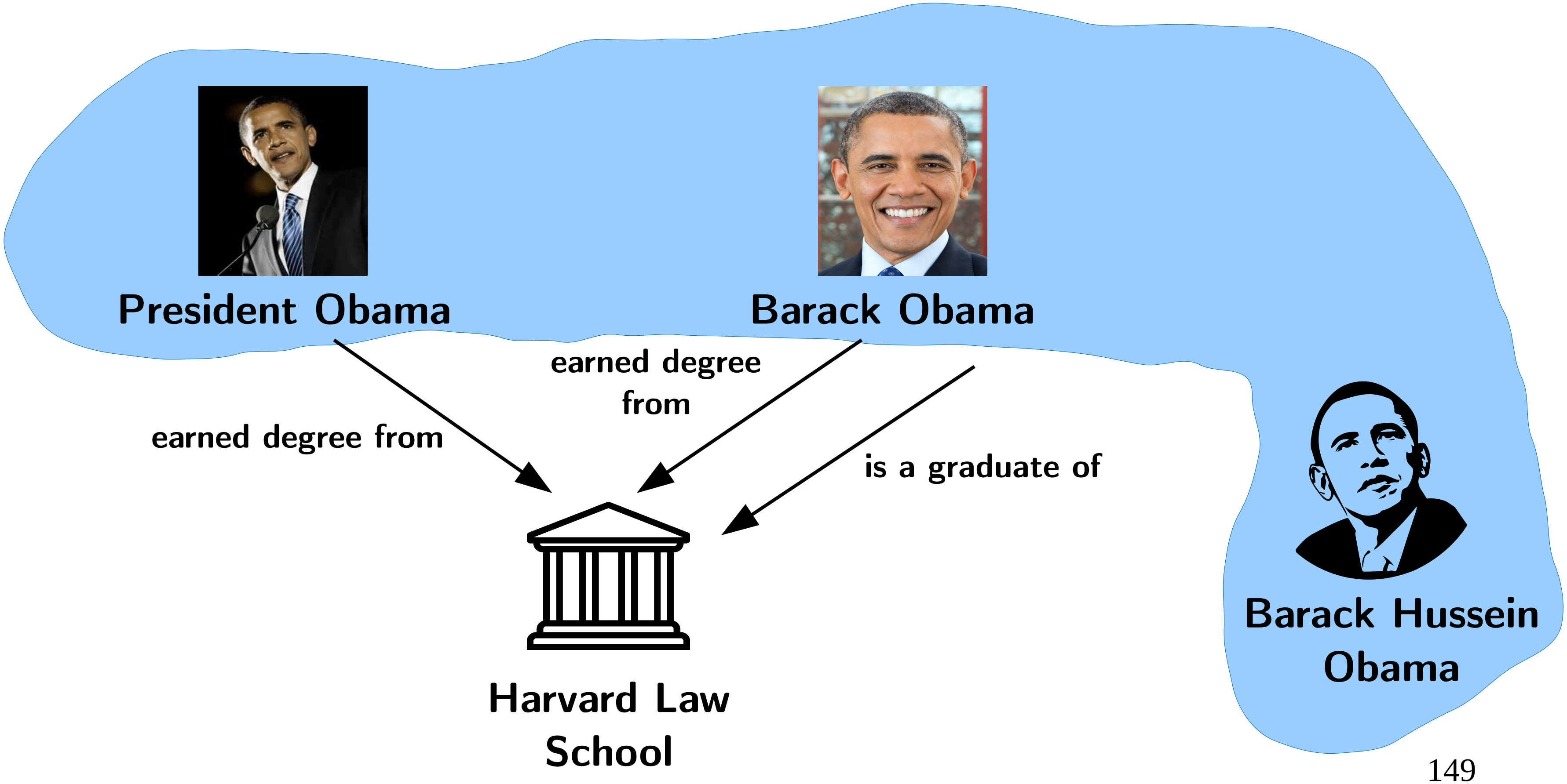
Noun phrase canonicalization

- Use signals to cluster synonym noun phrases
  - Agglomerative Clustering



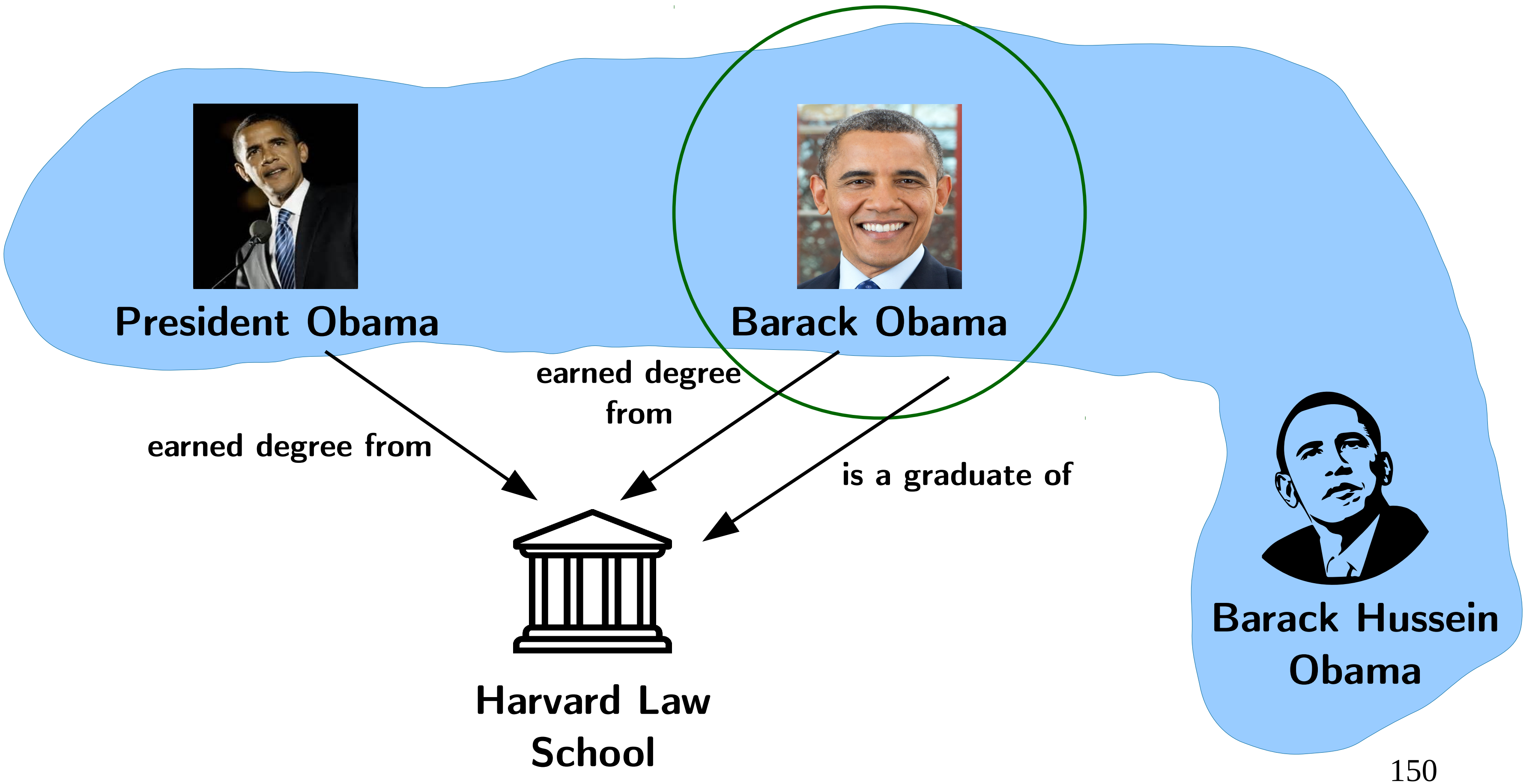
Noun phrase canonicalization

- Use signals to cluster synonym noun phrases
  - Agglomerative Clustering



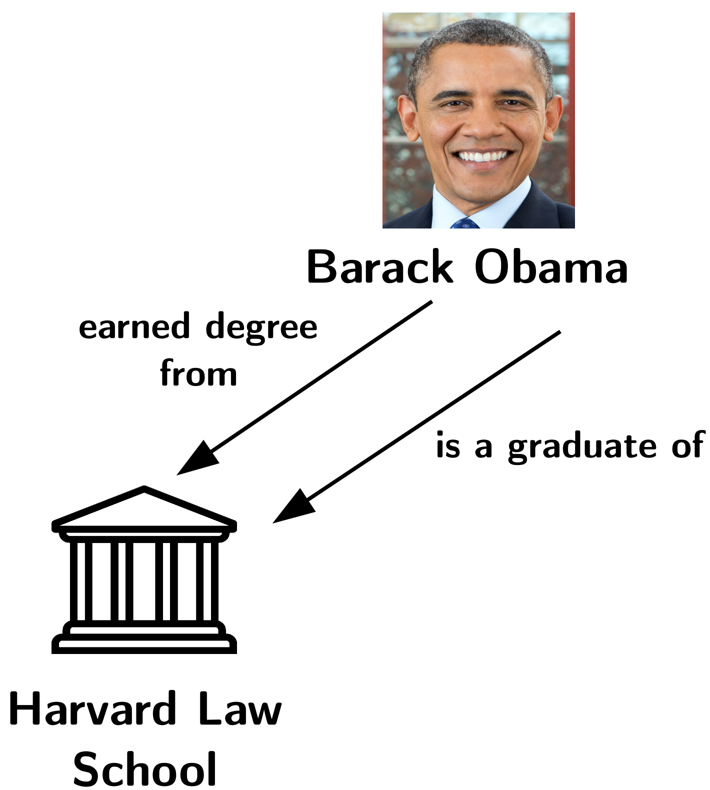
Noun phrase canonicalization

Pick one noun phrase to canonicalize all mentions



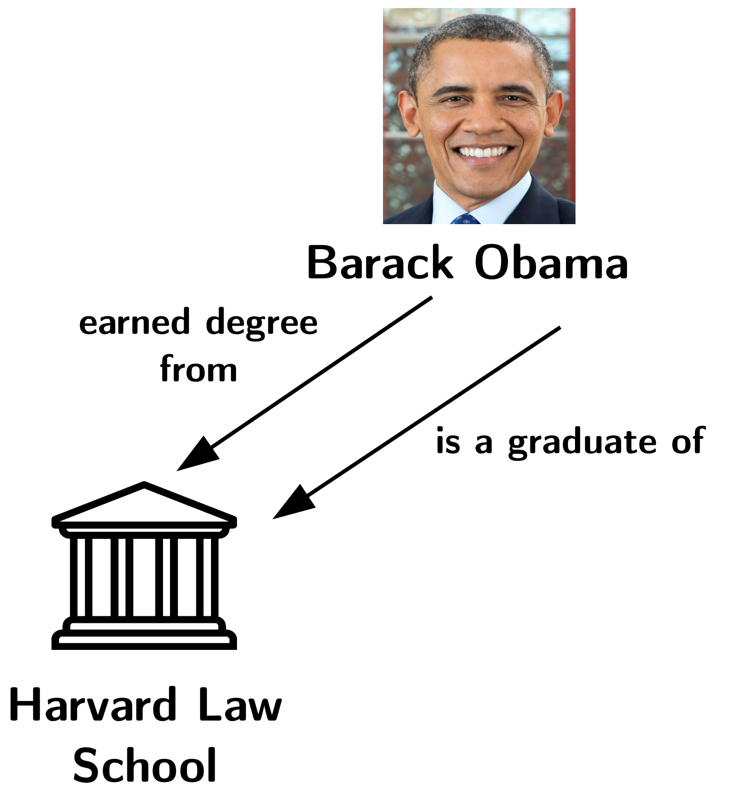
Noun phrase canonicalization

Pick one noun phrase to canonicalize all mentions



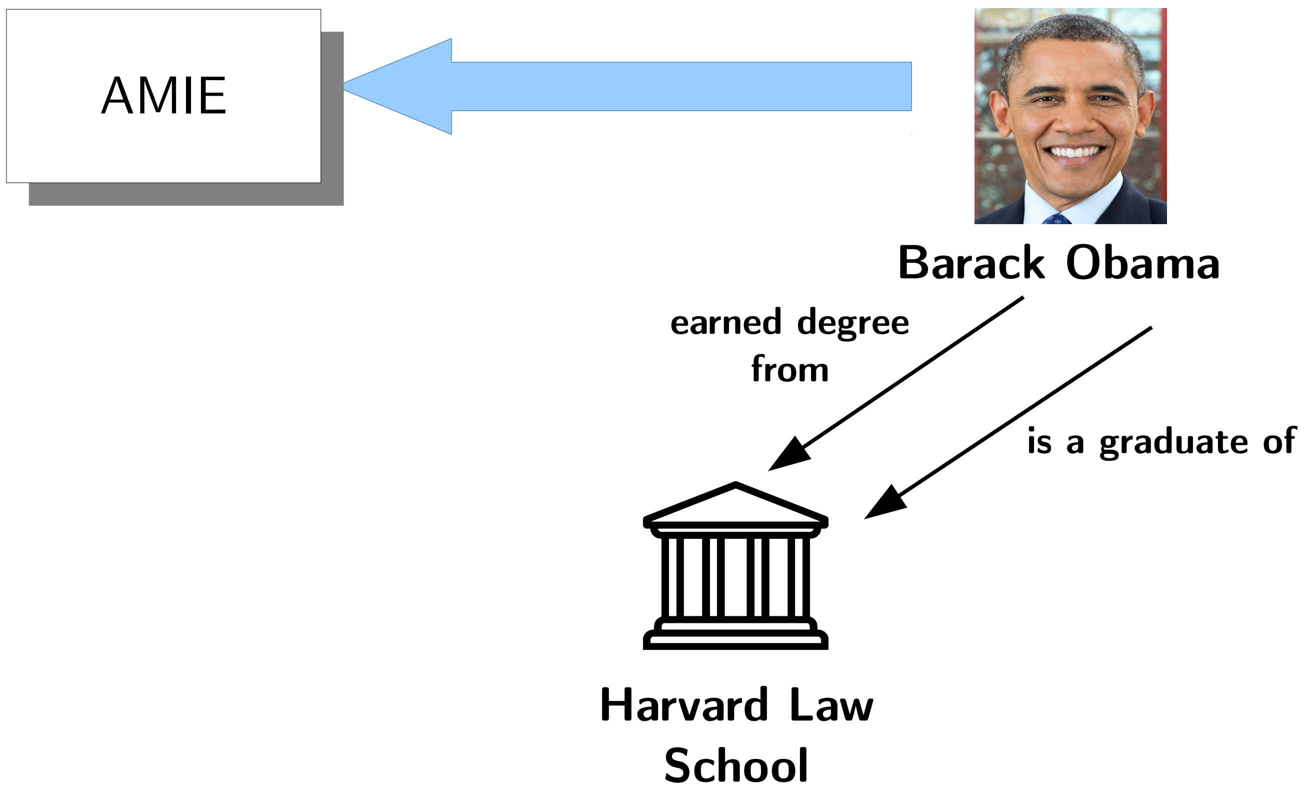
Verbal phrase clustering

Rely on canonicalization of the noun phrases



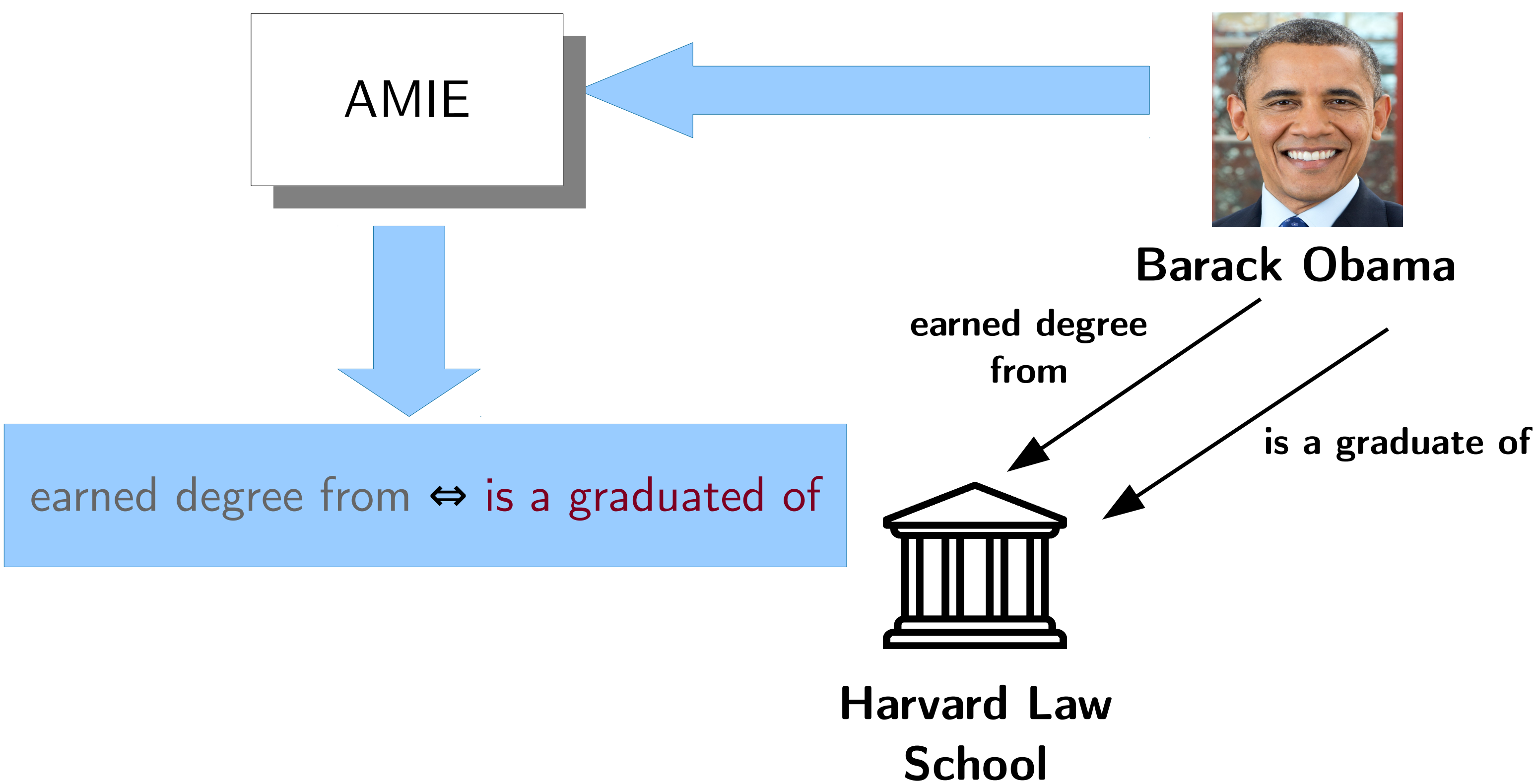
Verbal phrase clustering

Apply rule mining on the semi-canonicalized KB



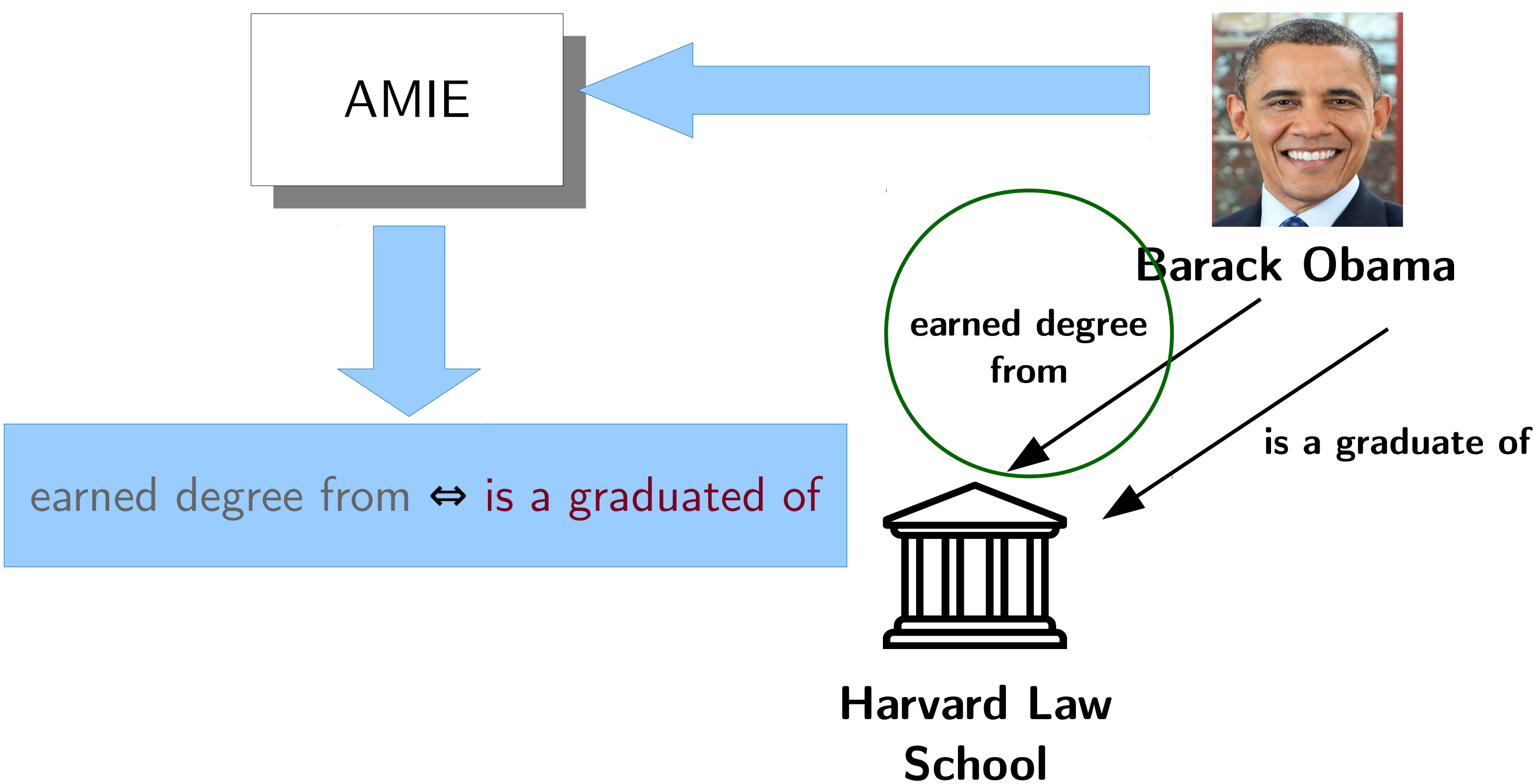
Verbal phrase clustering

Apply rule mining on the semi-canonicalized KB



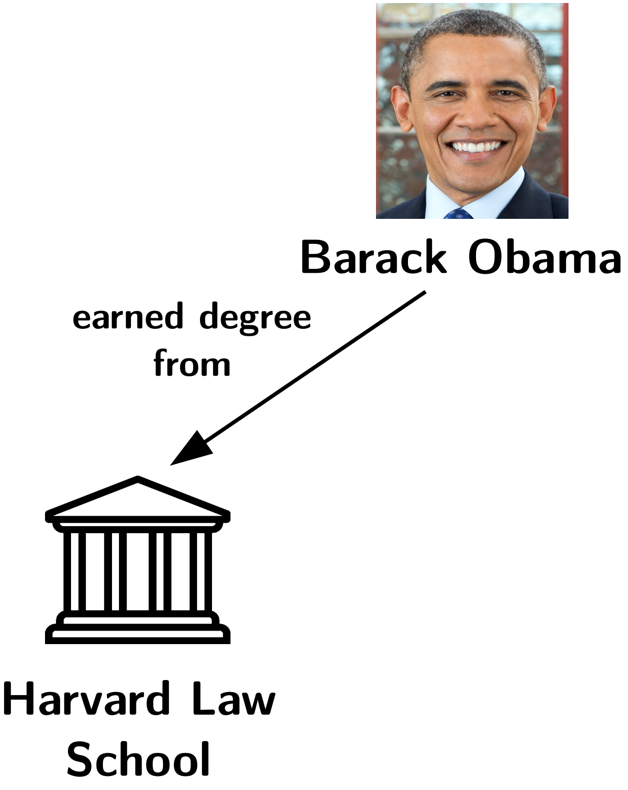
Verbal phrase clustering

Pick one verbal phrase for the canonicalization



Verbal phrase clustering

Pick one verbal phrase for the canonicalization



Experimental evaluation

Canonicalization of noun phrases

F1 measure of the signals on an open KB (with polysemy) constructed with Reverb on Clueweb09

Signal	Macro	Micro	Pairwise
Str identity	51%	84%	71%
Str. Similarity	51%	83%	67%
IDF token overlap	57%	**88%**	**79%**
Attr. Overlap	15%	28%	5%
Entity overlap	**63%**	78%	61%
Type overlap	62%	76%	56%
Word overlap	55%	76%	56%
Simple ML	55%	86%	78%
Full ML	61%	79%	46%

Canonicalization of verbal phrases

Performance of verbal phrase clustering on Reverb dataset taken from Clueweb09.

Dataset	Precision	Coverage
Reverb	94%	15%
Reverb (types)	**98%**	**21%**

Canonicalization of verbal phrases

Some clusters of verbal phrases could be linked to Freebase relations

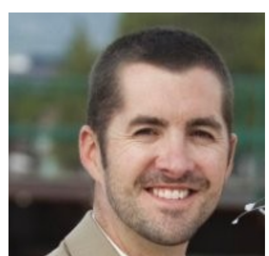
Cluster	Freebase
be spoken in, be the official language of, be the national language of	location.country.official_language
be bought, acquire	organization.organization.acquired_by

Summary

- Simple signals such as the tokens overlap are effective at identifying synonym noun phrases.
- Rule mining plus instance information can find clusters of close verbal phrases with high precision.

Luis Galárraga, Geremy Heitz, Kevin Murphy, Fabian Suchanek.
Canonicalizing Open Knowledge Bases.
In CIKM, 2014






Predicting Completeness

Predicting completeness in KBs

- KBs are highly incomplete

Predicting completeness in KBs

- KBs are highly incomplete
  - 2% of people have a father in Wikidata

Predicting completeness in KBs

- KBs are highly incomplete
  - 2% of people have a father in Wikidata
- We do not know where the incompleteness lies

Predicting completeness in KBs

- KBs are highly incomplete
  - 2% of people have a father in Wikidata
- We do not know where the incompleteness lies
  - A person without spouse in the KB could be incomplete or single

Predicting completeness in KBs

- KBs are highly incomplete
  - 2% of people have a father in Wikidata
- We do not know where the incompleteness lies
  - A person without spouse in the KB could be incomplete or single
- Problems for data producers and consumers

Predicting completeness in KBs

- KBs are highly incomplete
  - 2% of people have a father in Wikidata
- We do not know where the incompleteness lies
  - A person without spouse in the KB could be incomplete or single
- Problems for data producers and consumers
  - Consumers: no completeness guarantees for queries
  - Producers: which parts of the KB need to be populated?

Completeness

- Defined with respect to a **query q** via a complete hypothetical KB K*

Completeness

- Defined with respect to a **query q** via a complete hypothetical KB K*
  - A query q is complete in K, iff q(K*) $\subseteq$q(K)

Completeness

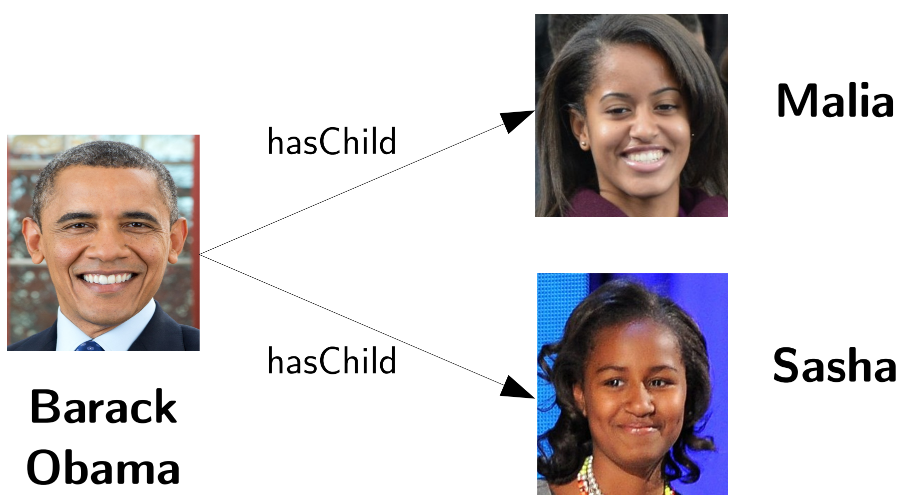
- Defined with respect to a **query q** via a complete hypothetical KB K*
  - A query q is complete in K, iff q(K*) $\subseteq$ q(K)
- We focus on queries of the form

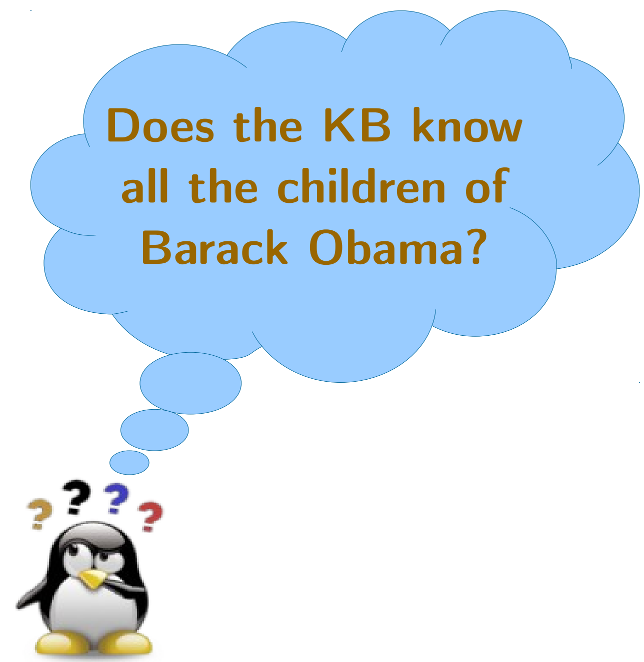
SELECT ?x { subject relation ?x }

Completeness

- Defined with respect to a **query q** via a complete hypothetical KB K*
  - A query q is complete in K, iff q(K*) ⊆ q(K)
- We focus on queries of the form

SELECT ?x { subject relation ?x }

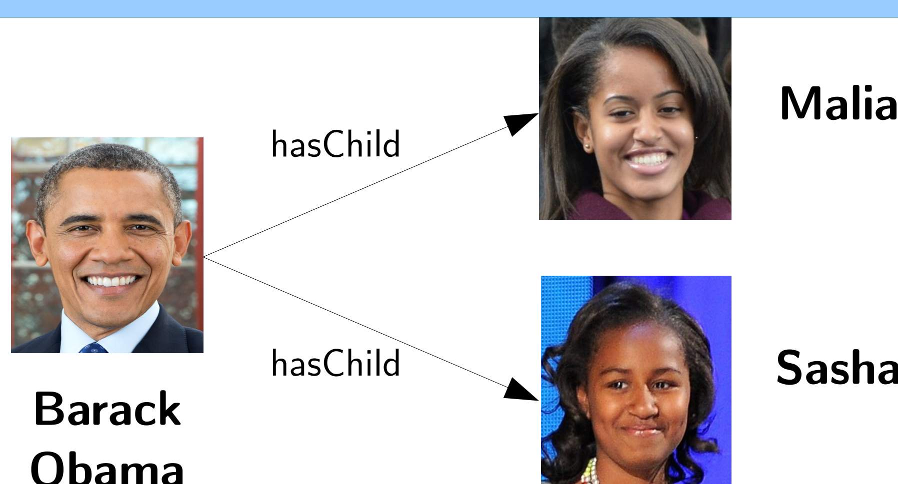




Completeness

- Defined with respect to a **query q** via a complete hypothetical KB K*

Goal: Study different signals to predict if a query of the form {o : r(s, o) } is complete in a KB



Completeness oracles

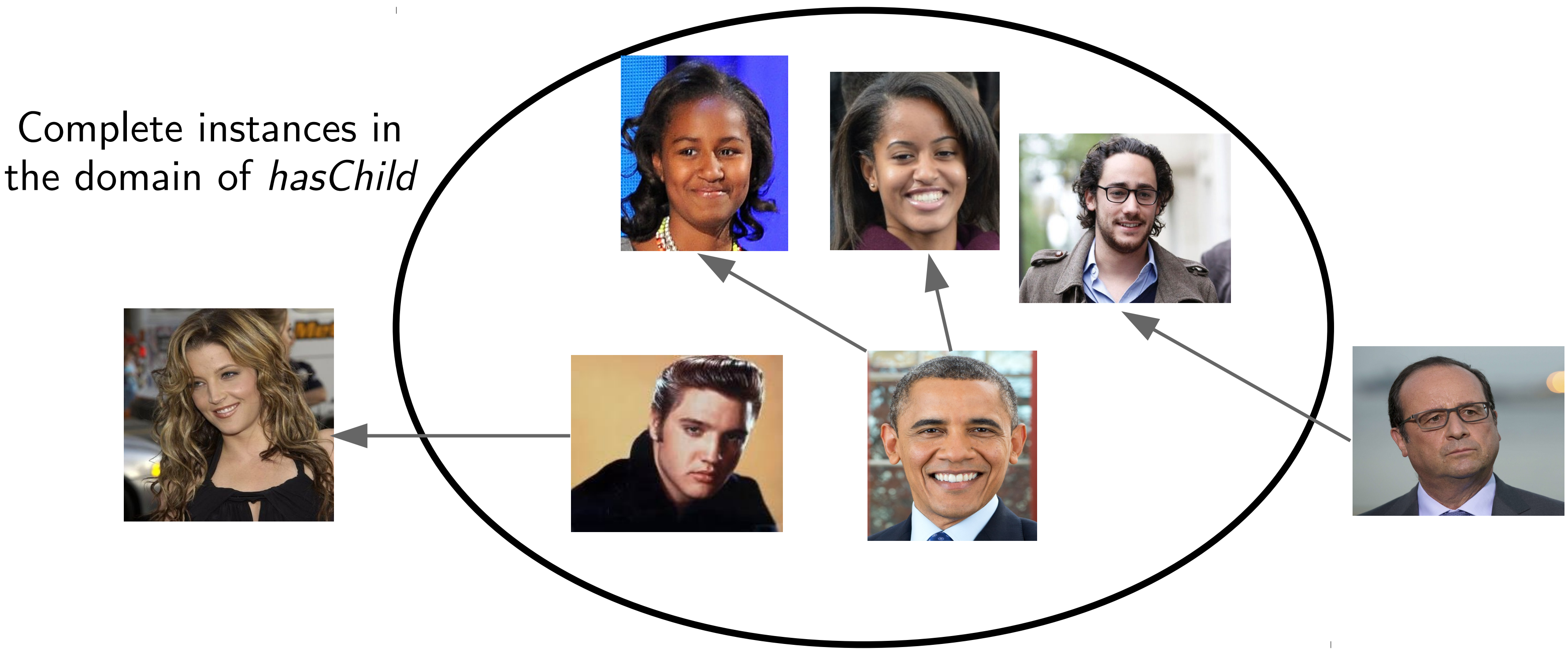
- Function that assigns a completeness value to pairs subject-relation (s, r)

Completeness oracles

- Function that assigns a completeness value to pairs subject-relation (s, r)
  - PCA oracle: (s, r) is **complete** if the KB knows at least one object *o*

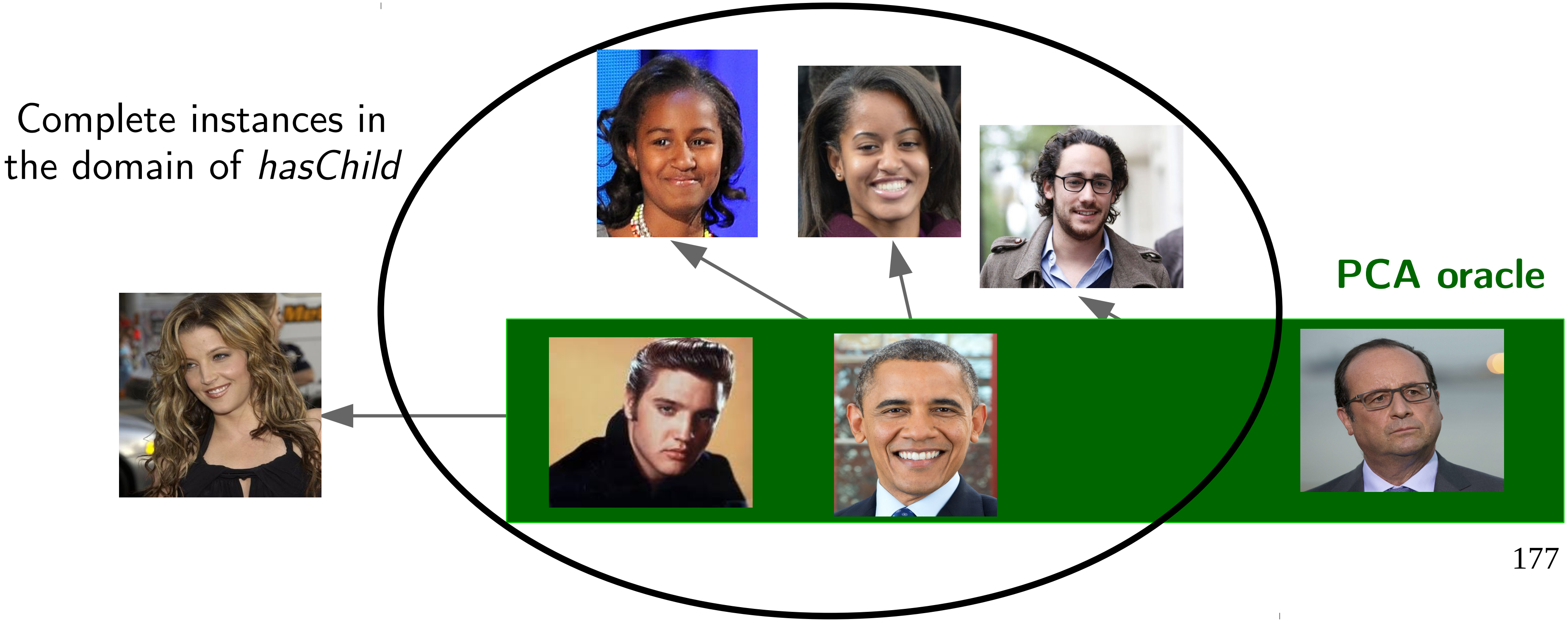
Completeness oracles

- Function that assigns a completeness value to pairs subject-relation (s, r)
  - PCA oracle: (s, r) is **complete** if the KB knows at least one object *o*



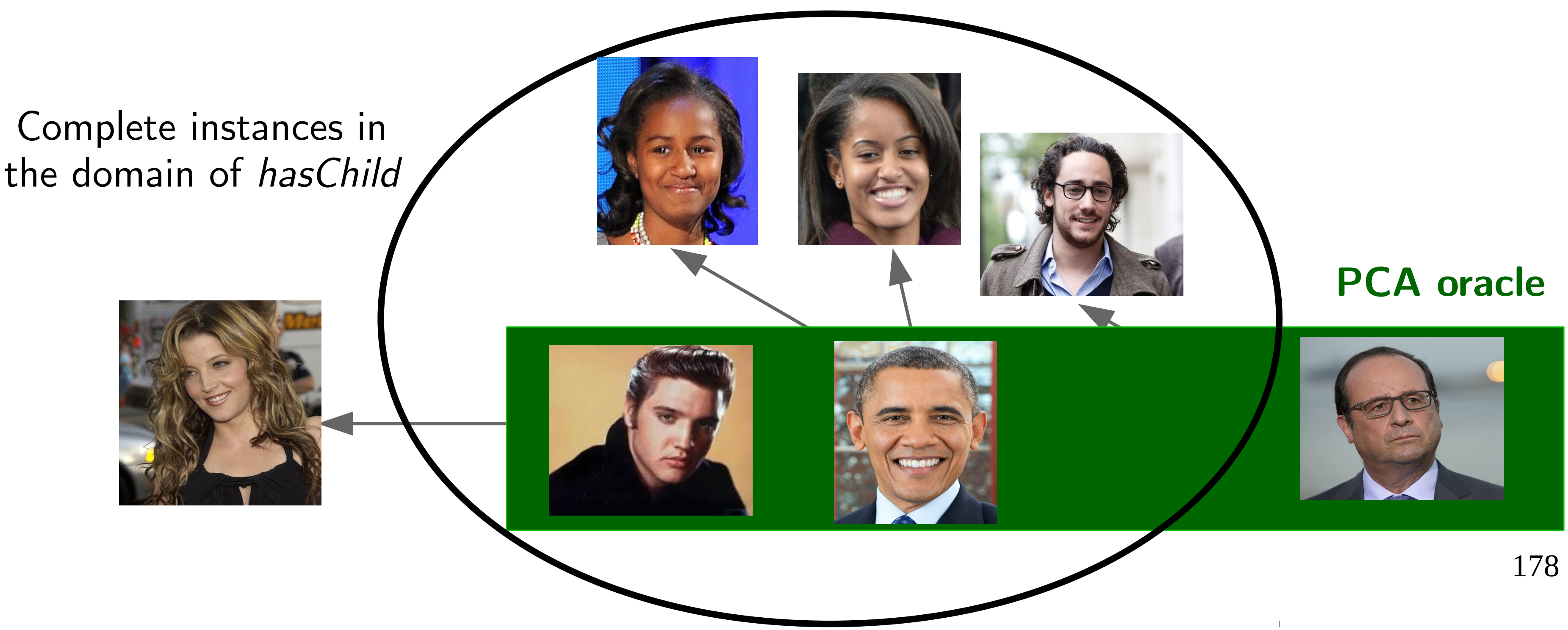
Completeness oracles

- Function that assigns a completeness value to pairs subject-relation (s, r)
  - PCA oracle: (s, r) is **complete** if the KB knows at least one object *o*



Completeness oracles

Oracles have certain precision and recall



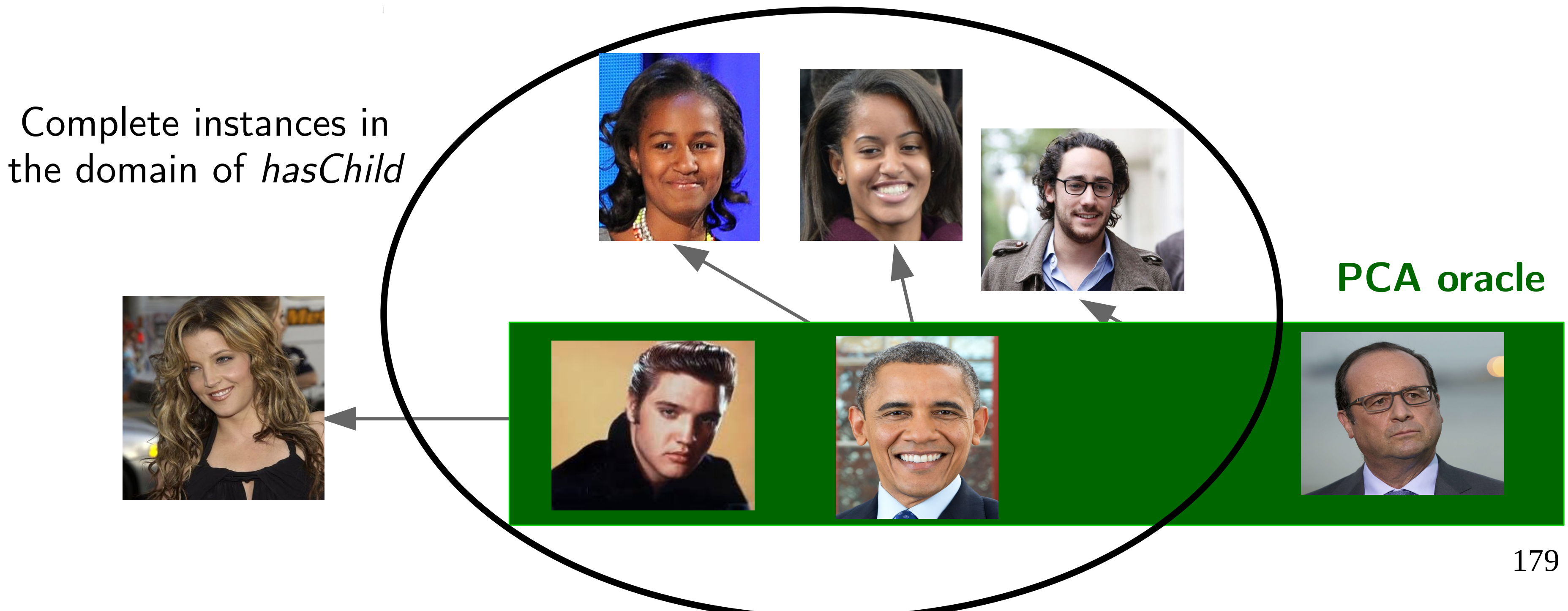
Completeness oracles

Oracles have certain precision and recall

PCA oracle

Precision = 2/3

Recall = 2/5



Completeness oracles

- CWA: $cwa(s, r) = true$
- PCA: $pca(s, r) = \exists\ o : r(s, o)$
- Cardinality: $card(s, r) = \#(o : r(s, o)) \geq k$
- Popular entities: $popularity_{pop}(s, r) = pop(s)$
- No-chg over time: $nochange_{chg}(s, r) = \sim chg(s, r)$
- Star : $star_{r_1,..,r_n}(s, r) = \forall\ i \in \{1,..,n\} : \exists\ o : r_i(s, o)$
- Class: $class_c(s, r) = type(s, c)$
- AMIE

Completeness oracles

- CWA: $cwa(s, r) = true$
- PCA: $pca(s, r) = \exists\, o : r(s, o)$
- Cardinality: $card(s, r) = \#(o : r(s, o)) \geq k$
- Popular entities: $popularity_{pop}(s, r) = pop(s)$
- No-chg over time: $nochange_{chg}(s, r) = \sim chg(s, r)$
- Star : $star_{r_1,..,r_n}(s, r) = \forall\, i \in \{1,..,n\} : \exists\, o : r_i(s, o)$
- Class: $class_c(s, r) = type(s, c)$
- AMIE

Learned oracles

Learned oracles

- Based on completeness rules

Learned oracles

- Based on completeness rules
  - Learned with AMIE from a set of completeness annotations *complete(s, r)* and *incomplete(s, r)*

Learned oracles

- Based on completeness rules
  - Learned with AMIE from a set of completeness annotations *complete(s, r)* and *incomplete(s, r)*

notype(x, Adult), type(x, Person) ⇒ complete(x, hasChild)

dateOfDeath(x, y), lessThan_1(x, placeOfDeath) ⇒ incomplete(x, placeOfDeath)

Learned oracles

- Based on completeness rules
  - Learned with AMIE from a set of completeness annotations *complete(s, r)* and *incomplete(s, r)*

    notype(x, Adult), type(x, Person) ⇒ complete(x, hasChild)

    dateOfDeath(x, y), lessThan_1(x, placeOfDeath) ⇒ incomplete(x, placeOfDeath)

  - Annotations obtained by two means:
    - Automatic: e.g., everyone must have a nationality
    - Crowd-sourcing: ask mechanical turks for more objects in the web

AMIE Oracle

- It uses learned rules to predict completeness

AMIE Oracle

- It uses learned rules to predict completeness
- In case of contradictions, predictions with higher confidence and support prevail

Experimental evaluation

Evaluating oracles

F1 measure of the oracles in YAGO3

Relation	CWA	PCA	Class	AMIE
diedIn	60%	22%	**99%**	96%
directed	40%	96%	0%	**100%**
graduatedFrom	89%	4%	**92%**	87%
hasChild	71%	1%	**78%**	**78%**
hasGender	78%	**100%**	95%	**100%**
hasParent	1%	54%	0%	**100%**
isCitizenOf	4%	98%	5%	**100%**
isConnectedTo	87%	34%	88%	**89%**
isMarriedTo	55%	7%	**57%**	46%
wasBornIn	28%	**100%**	0%	**100%**

Summary

- It is possible to predict completeness in KBs with 100% precision
  - By combining different simple oracles (signals)
- Future work:
  - More signals of completeness, completeness predictions for rule mining.

Luis Galárraga, Simon Razniewski, Antoine Amarilli, Fabian Suchanek.
Predicting completeness in Knowledge Bases.
Under Review.



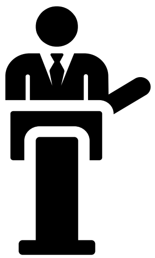

TELECOM ParisTech Institut Mines-Télécom unibz

Final conclusion

Final conclusion

- Rule Mining is about making sense out of semantic knowledge.
- Rule Mining can:
  - Produce insights about the data (AMIE)
  - Predict missing data (wikilinks)
  - Align the schemas of KBs
  - Cluster synonym verbal phrases
  - Predict completeness
- With the goal of making computers even smarter and more helpful to humans.

Credits




Icons made by Freepik from www.flaticon.com

Icon made by Business Dubai from www.flaticon.com