Knowledge Base Curation using Constraints

PHD defense

Thomas Pellissier Tanon

Télécom Paris

September 7th, 2020

Knowledge base graph structure

A knowledge base is a repository of structured information.

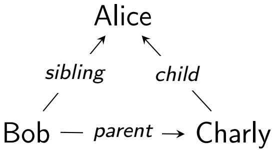


Figure: Example of a knowledge base

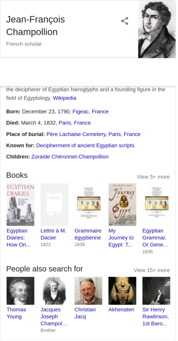
Knowledge base

Knowledge bases can be used to:

- Display key facts
- Answer questions
- Suggest
- Detect patterns

Some knowledge bases are:

- Wikidata
- YAGO
- Google Knowledge Graph
- Facebook Social Graph
- ...



Knowledge base problems

Knowledge bases are affected by two types of problem:

- Incompleteness: missing facts
- Errors: wrong facts

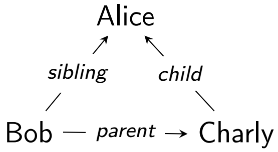


Figure: Example of knowledge base

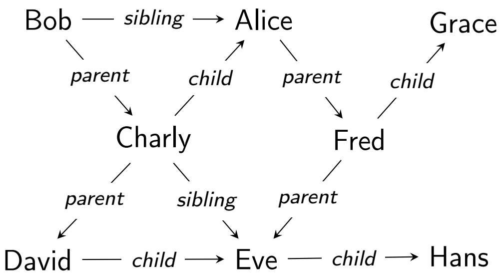
Presentation outline

1. Knowledge Base Completion
2. Knowledge Base Correction
3. Knowledge Base Creation and Querying

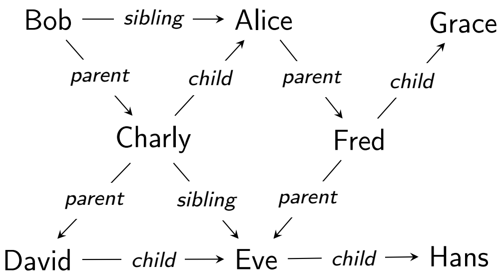
Where are we?

1. Knowledge Base Completion
2. Knowledge Base Correction
3. Knowledge Base Creation and Querying

Well known approach: rule mining

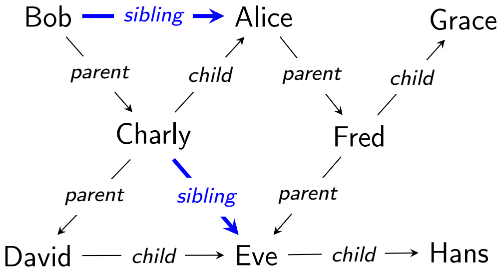


Well known approach: rule mining



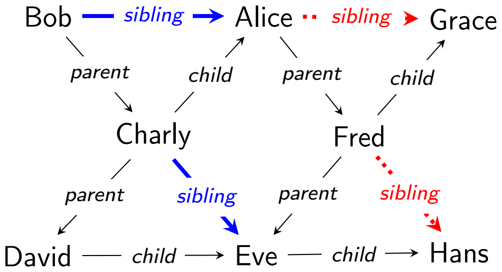
$$r_1 : \underbrace{parent(x,z) \wedge child(z,y)}_{\text{body } b(x,y)} \rightarrow \underbrace{sibling(x,y)}_{\text{head } h(x,y)}$$

Rule application



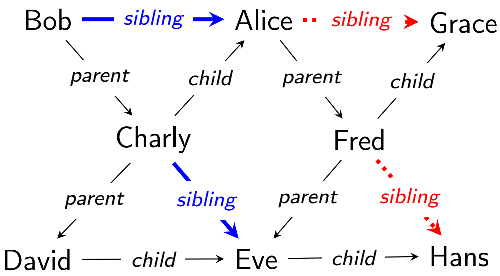
$$r_1 : \underbrace{parent(x,z) \wedge child(z,y)}_{\text{body } b(x,y)} \rightarrow \underbrace{sibling(x,y)}_{\text{head } h(x,y)}$$

Rule application



$$r_1 : \underbrace{parent(x,z) \wedge child(z,y)}_{\text{body } b(x,y)} \rightarrow \underbrace{sibling(x,y)}_{\text{head } h(x,y)}$$

Rule evaluation



Body support

$supp_b(r_1) = |b| = 4$

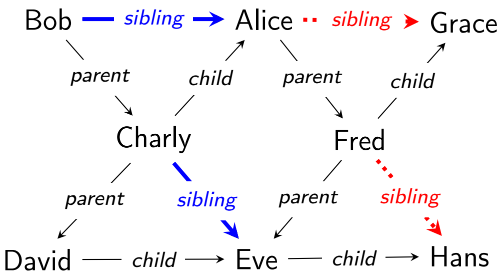
Rule support

$supp(r_1) = |b \wedge h| = 2$

$$r_1 : \underbrace{parent(x,z) \wedge child(z,y)}_{\text{body } b(x,y)} \rightarrow \underbrace{sibling(x,y)}_{\text{head } h(x,y)}$$

Closed world assumption

"I know everything"



$$r_1 : \underbrace{parent(x,z) \wedge child(z,y)}_{\text{body } b(x,y)} \rightarrow \underbrace{sibling(x,y)}_{\text{head } h(x,y)}$$

Body support

$$supp_b(r_1) = |b| = 4$$

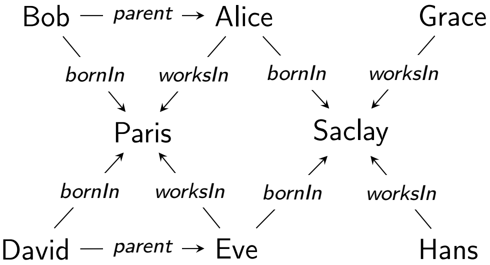
Rule support

$$supp(r_1) = |b \wedge h| = 2$$

Closed World Confidence

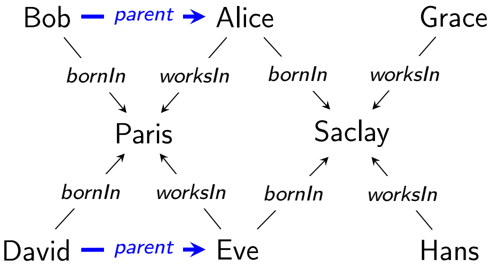
$$conf(r_1) = \frac{supp(r_1)}{supp_b(r_1)} = \frac{2}{4}$$

A bad rule



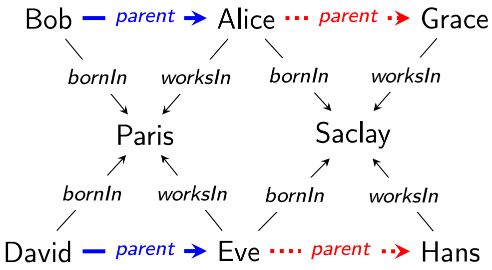
$$r_2 : bornIn(x, z) \wedge worksIn(y, z) \rightarrow parent(x, y)$$

A bad rule



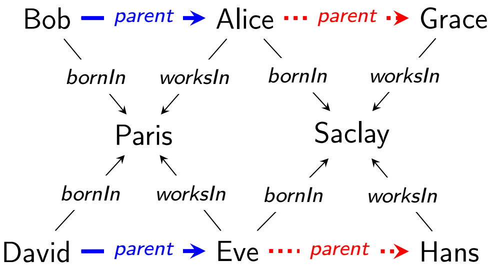
$$r_2 : bornIn(x, z) \wedge worksIn(y, z) \rightarrow parent(x, y)$$

A bad rule



$$r_2 : bornIn(x, z) \wedge worksIn(y, z) \rightarrow parent(x, y)$$

A bad rule



Body support

$supp_b(r_2) = |b| = 4$

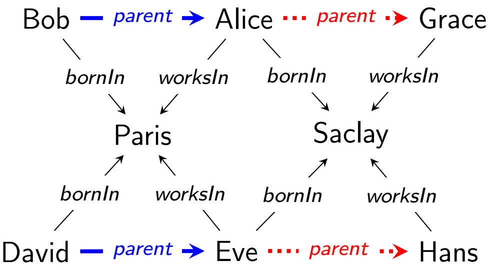
Rule support

$supp(r_2) = |b \wedge h| = 2$

$$r_2 : bornIn(x, z) \wedge worksIn(y, z) \rightarrow parent(x, y)$$

A bad rule

"I know everything"



$$r_2 : bornIn(x, z) \wedge worksIn(y, z) \rightarrow parent(x, y)$$

Body support

$$supp_b(r_2) = |b| = 4$$

Rule support

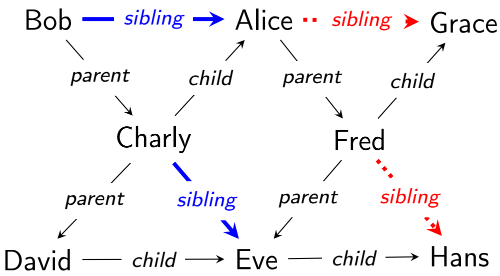
$$supp(r_2) = |b \wedge h| = 2$$

Closed World Confidence

$$conf(r_2) = \frac{supp(r_2)}{supp_b(r_2)} = \frac{2}{4}$$

Partial Completeness Assumption (PCA) (Galárraga et al.)

"If I know something, I know everything"



$$r_1 : \underbrace{parent(x,z) \wedge child(z,y)}_{\text{body } b(x,y)} \rightarrow \underbrace{sibling(x,y)}_{\text{head } h(x,y)}$$

PCA support

$$supp_{pca}(r_1) = |b \wedge h(x,*)| = 2$$

Rule support

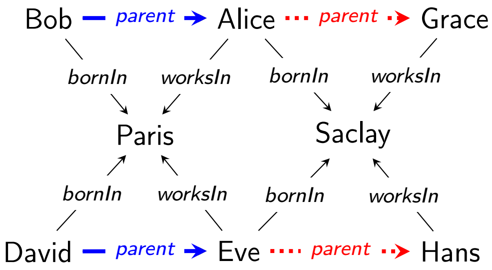
$$supp(r_1) = |b \wedge h| = 2$$

PCA confidence

$$conf_{pca}(r_1) = \frac{supp(r_1)}{supp_{pca}(r_1)} = \frac{2}{2}$$

With the bad rule

“If I know something, I know everything“



$$r_2 : bornIn(x, z) \wedge worksIn(y, z) \rightarrow parent(x, y)$$

PCA support

$$supp_{pca}(r_2) = |b \wedge h(x, *)| = 2$$

Rule support

$$supp(r_2) = |b \wedge h| = 2$$

PCA confidence

$$conf_{pca}(r_2) = \frac{supp(r_2)}{supp_{pca}(r_2)} = \frac{2}{2}$$

How to distinguish these two rules?

Paper: "Completeness-Aware Rule Learning from Knowledge Graphs" with Daria Stepanova, Simon Razniewski, Paramita Mirza and Gerhard Weikum

Full paper nominated for the best student paper award at ISWC 2017

Invited presentation at IJCAI 2018

Additional input: cardinality facts

Number of values for a given (subject, predicate)

- Retrieved from text extraction e.g. "Alice has 3 children"
- Deduced from the ontology (functional relations...)
- Learned from existing cardinalities

Cardinality fact formalization

- $num(p, s)$: Number of outgoing p-edges from s in the real world
- $miss(p, s)$: Number of outgoing p-edges from s missing from the KB

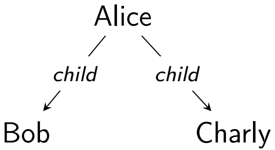
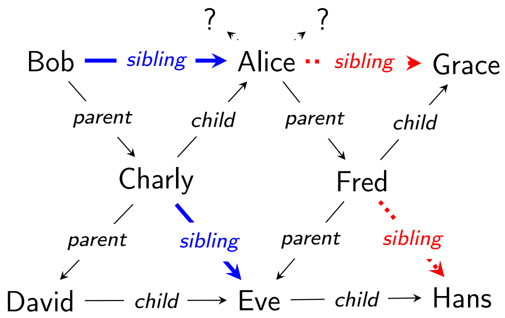


Figure: If $num(\text{child}, \text{Alice}) = 3$ then $miss(\text{child}, \text{Alice}) = 1$ in this KB

Completeness-aware confidence

"It is fine to add missing data"



With $num(\text{sibling}, \text{Alice}) = 2$

$$r_1 : \underbrace{parent(x,z) \wedge child(z,y)}_{\text{body } b(x,y)} \rightarrow \underbrace{sibling(x,y)}_{\text{head } h(x,y)}$$

Completeness support

$$\begin{aligned} supp_c(r_1) = {} & supp_b(r_1) \\ & - npi(r_1) \\ = {} & 4 - 1 = 3 \end{aligned}$$

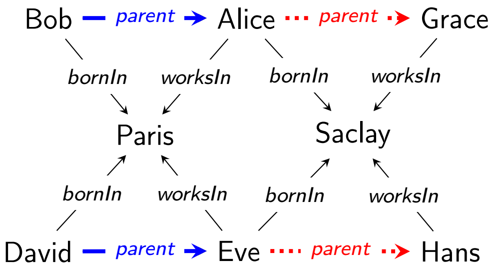
with $npi(r)$ the number of facts added to incomplete areas by r

Completeness confidence

$$conf_c(r_1) = \frac{supp(r_1)}{supp_c(r_1)} = \frac{2}{3}$$

With the bad rule

"It is fine to add missing data"



$$r_2 : bornIn(x, z) \wedge worksIn(y, z) \rightarrow parent(x, y)$$

Completeness support

$$\begin{aligned} supp_c(r_2) = \quad & supp_b(r_2) \\ & - npi(r_2) \\ = \quad & 4 - 0 = 4 \end{aligned}$$

with $npi(r)$ the number of facts added to incomplete areas by r

Completeness confidence

$$conf_c(r_2) = \frac{supp(r_2)}{supp_c(r_2)} = \frac{2}{4}$$

It generalizes the other confidence metrics

Closed world assumption

$conf(r) = conf_c(r)$ because $npi(p, s) = 0$

Partial completeness assumption

$conf_{pca}(r) = conf_c(r)$ because $npi(p, s) = 0$ in areas with facts and $npi(p, s) = supp_b(r) \mid_{p(s,*)}$ elsewhere

Evaluation: Two datasets

LUBM

- Synthetic dataset with a rich ontology (1.2 M facts)
- We use the ontology to complete the dataset
- We compute cardinalities from the complete dataset
- We remove facts randomly (the % depends on the fact predicate) to create the available dataset

WikidataPeople

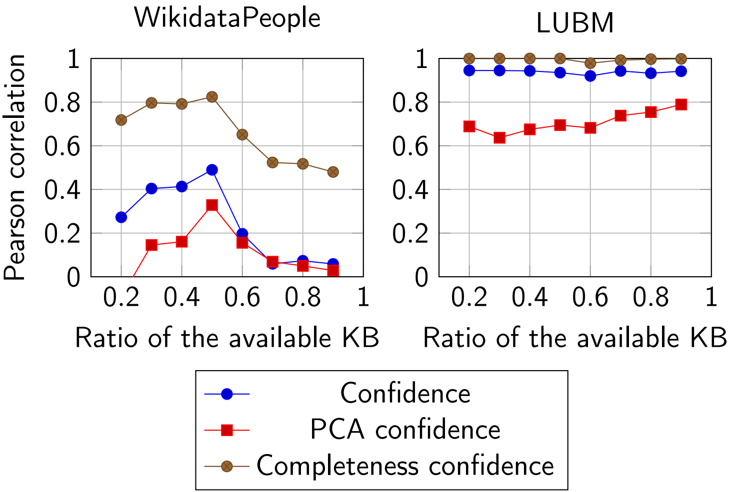
- Subset of Wikidata (2.4M facts over 9 predicates)
- We use manual rules to complete the dataset
- We compute cardinalities from the complete dataset
- We remove facts randomly to create the available dataset

Evaluation: Protocol

$$\begin{tikzcd}
\text{Available dataset} \arrow[d] & & \text{Complete dataset} \arrow[d] \\
\text{All rules } r \arrow[d] & & \text{All rules } r \arrow[d] \\
conf(r)/conf_{pca}(r)/conf_c(r) \arrow[dr] & & conf(r) \arrow[dl] \\
& \text{Pearson correlation} &
\end{tikzcd}$$

Pearson correlation: $\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$ with *cov* the covariance and σ the standard deviation

Evaluation: Results



Where are we?

1. Knowledge Base Completion
2. Knowledge Base Correction
3. Knowledge Base Creation and Querying

Constraint rules

We define constraints on the knowledge base using rules.
For example:

- "The possible genders are male, female and non-binary":
  $\Gamma_1(x, y) : \mathit{gender}(y, x) \rightarrow x \in \{\text{male}, \text{female}, \text{nonbinary}\}$
- "The value of the birth place relation should be a place":
  $\Gamma_2(x, y) : \mathit{birthPlace}(y, x) \rightarrow \mathit{type}(x, \text{Place})$

Stats

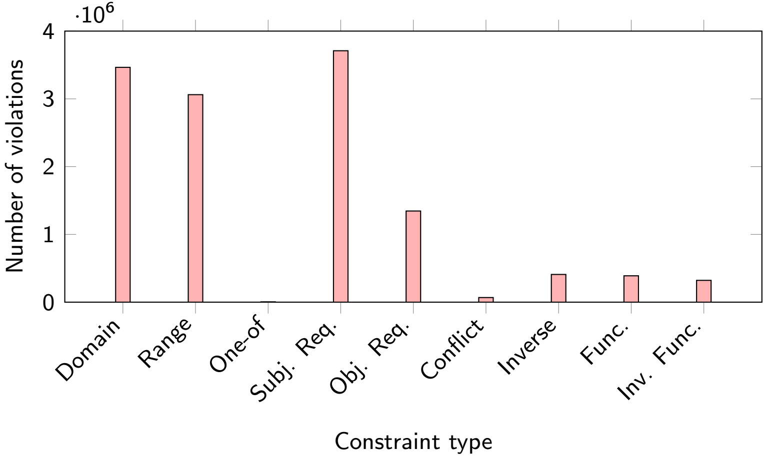


Figure: Wikidata constraint violations (July 2018)

Constraint violation

A violation of a constraint $\Gamma(\vec{x})$ is a minimal subset $\mathcal{V}$ of the KB $\mathcal{K}$ such that there exists $\vec{a}$ with $\mathcal{V} \not\models \Gamma(\vec{a})$ and $\mathcal{K} \not\models \Gamma(\vec{a})$.

Example

If $\mathit{gender}(\text{Alice}, \text{woman}) \in \mathcal{K}$
then $\mathcal{V} = \{\mathit{gender}(\text{Alice}, \text{woman})\}$ is a violation of
$\Gamma_1(x, y) : \mathit{gender}(y, x) \rightarrow x \in \{\text{male}, \text{female}, \text{nonbinary}\}$.

How to fix constraint violations?

- Sometimes it is easy:
  For example, replace the gender value "woman" by "female".

How to fix constraint violations?

- Sometimes it is easy:
  For example, replace the gender value "woman" by "female".
- but it is often hard:
  When a birth place is not a place, should we remove the bad value, insert the "place" type or a subtype "city", "hospital"...

How to automatically fix constraint violations?

Paper: "Learning How to Correct a Knowledge Base from the Edit History" with Camille Bourgaux and Fabian Suchanek

Full paper at WWW 2019

Atomic modification

An atomic modification is a tuple $(\mathcal{M}^-, \mathcal{M}^+)$ that is either:

- a fact addition: $(\emptyset, \{p(s, o)\})$
- a fact deletion: $(\{p(s, o)\}, \emptyset)$
- a fact replacement: $(\{p^-(s^-, o^-)\}, \{p^+(s^+, o^+)\})$

Example

$(\{\mathit{gender}(\text{Alice}, \text{woman})\}, \{\mathit{gender}(\text{Alice}, \text{female})\})$ is an atomic modification that replaces Alice's gender from "woman" to "female".

Solution of a constraint correction

A solution of a constraint violation $\mathcal{V}$ of $\Gamma(\vec{a})$ on $\mathcal{K}$ is an atomic modification $(\mathcal{M}^-, \mathcal{M}^+)$ such that there exists a knowledge base $\mathcal{K}' \subseteq \mathcal{K}$ with $\mathcal{V} \subseteq \mathcal{K}'$ and $(\mathcal{K}' \cup \mathcal{M}^+) \setminus \mathcal{M}^-$ satisfies $\Gamma(\vec{a})$.

Example

$(\{\mathit{gender}(\text{Alice}, \text{woman})\}, \{\mathit{gender}(\text{Alice}, \text{female})\})$ is a solution for the violation $\mathcal{V} = \{\mathit{gender}(\text{Alice}, \text{woman})\}$ of $\Gamma_1(x, y) : \mathit{gender}(y, x) \rightarrow x \in \{\text{male}, \text{female}, \text{nonbinary}\}$.

Which solution?

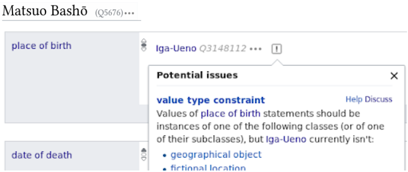
Example

Let's consider $\Gamma_2(x, y) : \mathit{birthPlace}(y, x) \rightarrow \mathit{type}(x, \text{Place})$. We have a violation $\{\mathit{birthPlace}(y, x)\}$. What is the "good" solution to improve the KB?

- Remove $\mathit{birthPlace}(y, x)$?
- Add $\mathit{type}(x, \text{Place})$?
- Add $\mathit{type}(x, C)$ with $\mathit{subClassOf}(C, \text{Place})$?

Idea: The KB edit history provides past corrections

Before



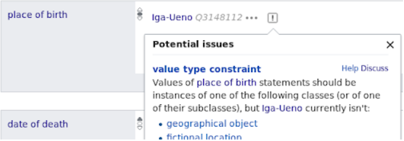
After

Edit:

(examples with Wikidata)

Extracting past corrections

To solve violations like



two solutions:

An addition like

A deletion of

We look for such edits and check if they correct a violation

There are patterns for finding good solutions

Example

Let's keep considering:

$$\Gamma_2(x, y) : \mathit{birthPlace}(y, x) \rightarrow \mathit{type}(x, \text{Place})$$

And a past correction of $\Gamma_2(\text{Matuso Basho}, \text{Iga-Ueno})$:

$$(\emptyset, \{\mathit{type}(\text{Iago-Ueno}, \text{GeoObject})\})$$

We could generalize this correction by:

$$[\Gamma_2(x, y)] \rightarrow (\emptyset, \{\mathit{type}(x, \text{GeoObject})\})$$

And refine it with:

$$[\Gamma_2(x, y)] \wedge \mathit{geoCoordinates}(x, z) \rightarrow (\emptyset, \{\mathit{type}(x, \text{GeoObject})\})$$

Correction rule mining (CorHist)

1. Generate simple rules from the past corrections
2. Specialize the rules using the KB state just before the correction has been done

Using regular rule mining and standard confidence

What about shallow and textual information?

With CorHist we do not make use of information like “Iga-Ueno is a neighborhood in Japan”.

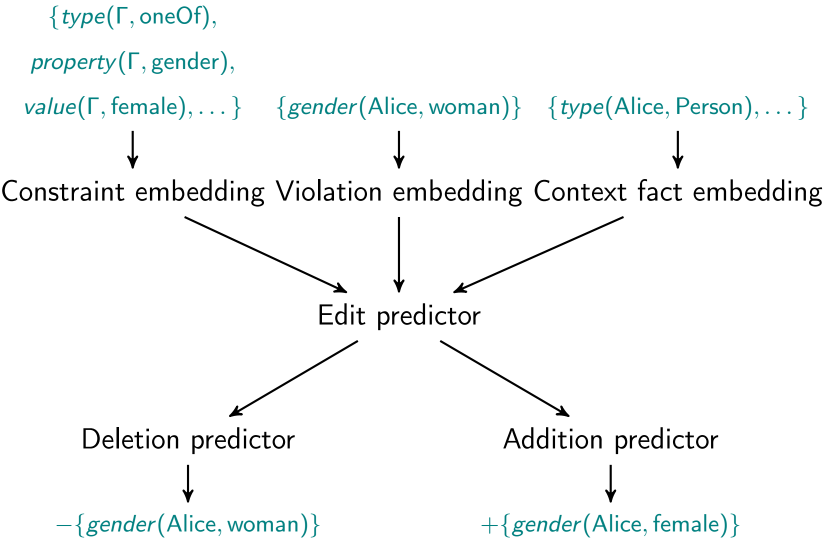
What about shallow and textual information?

With CorHist we do not make use of information like "Iga-Ueno is a neighborhood in Japan".

Idea: use a neural network that predicts the edit using:

- simple learned vector encoding of major entities and relations
- constraint description and entity facts embedding to allow generalization
- textual embedding of literal values (entity labels, literal objects...)

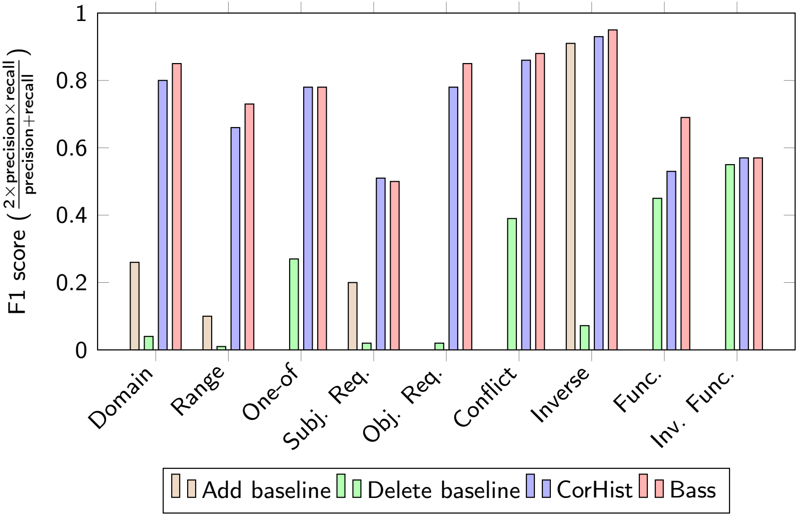
Bass


{type(Γ, oneOf),
property(Γ, gender),
value(Γ, female), . . . }
{gender(Alice, woman)}
{type(Alice, Person), . . . }
Constraint embedding
Violation embedding
Context fact embedding
Edit predictor
Deletion predictor
Addition predictor
−{gender(Alice, woman)}
+{gender(Alice, female)}

Evaluation on the past corrections (on Wikidata)

1. Extract the past corrections
2. Train CorHist and Bass on a training subset
3. Optimize CorHist hyperparameters on a cross-validation subset
4. Apply the predictions on a test subset
5. Compute precision and recall

Wikidata evaluation: Some results



Baselines: remove the violation or add the missing triple if possible

User evaluation: A Wikidata editing "game"

Francesco Belinzeri [Q57082102]

Auto | it

Francesco Belinzeri is a Italian sculptor, painter, and architect.

Violation

An entity should not have a statement for country of citizenship if it also has sex or gender with value male non-human

Possible correction

Edit statement (Q57082102, sex or gender, male non-human organism). Setting value to: male

- 64 users
- >30k actions
- >20k violations fixed on Wikidata

Where are we?

1. Knowledge Base Completion
2. Knowledge Base Correction
3. Knowledge Base Creation and Querying

Wikidata History Query Service

- Allows querying full Wikidata state at any time
- Using SPARQL queries
- Allows querying both content and edit metadata
- Used to extract the past violation corrections for CorHist

Paper: “Querying the Edit History of Wikidata ”, Thomas Pellissier Tanon and Fabian Suchanek, demo paper at ESWC 2019

YAGO 4

- New version of the YAGO knowledge base
- Based on Wikidata and schema.org
- Easy to use
- Generic build pipeline that enforces constraints
- Challenges:
  - Enforcing constraints efficiently (billions of facts)
  - Static violation repairs

Paper: “YAGO 4: a Reason-able Knowledge Base”, Thomas Pellissier Tanon, Gerhard Weikum and Fabian Suchanek, resource paper at ESWC 2020

Bash Datalog

- Translates Datalog to Bash shell code
- Allows for simple and efficient data preprocessing
- Uses relational algebra internally (my contribution)

Paper: "Bash Datalog: Answering Datalog Queries with Unix Shell Commands", Thomas Rebele, Thomas Pellissier Tanon and Fabian Suchanek, full paper at ISWC 2018 (spotlight paper)

And also

- "Property Label Stability in Wikidata", Thomas Pellissier Tanon and Lucie-Aimée Kaffee, WikiWorkshop @ WWW 2018
- "Demoing Platypus - A Multilingual Question Answering Platform for Wikidata", Thomas Pellissier Tanon, Marcos Dias de Assunção, Eddy Caron and Fabian Suchanek, demo at ESWC 2018
- "Question Answering Benchmarks for Wikidata", Dennis Diefenbach, Thomas Pellissier Tanon et al., poster at ISWC 2017

Main contributions

- Rule mining on incomplete data using cardinality information
- Automatically correcting a knowledge base using the edit history
- Contributions to knowledge base creation and knowledge base querying

Future work

- Publish the neural network approach
- Learn rules to fix text instead of fixing the knowledge base
- Learn more cardinalities and numerical values
- New (unrelated) work on SPARQL query compilation

Thank you!

- Pellissier Tanon, Weikum and Suchanek, "YAGO 4: a Reason-able Knowledge Base", full paper at ESWC 2020
- Pellissier Tanon, Bourgaux and Suchanek, "Learning How to Correct a Knowledge Base from the Edit History", full paper at WWW 2019
- Pellissier Tanon, and Suchanek, "Querying the Edit History of Wikidata", demo at ESWC 2019
- Rebele, Pellissier Tanon and Suchanek, "Bash Datalog: Answering Datalog Queries with Unix Shell Commands", full paper at ISWC 2018
- Pellissier Tanon, Stepanova, Razniewski, Mirza and Weikum, "Completeness-aware Rule Learning from Knowledge Graphs", full paper at ISWC 2017