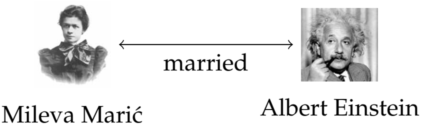
Expanding the YAGO knowledge base

Thomas Rebele

Télécom ParisTech

2018-07-05

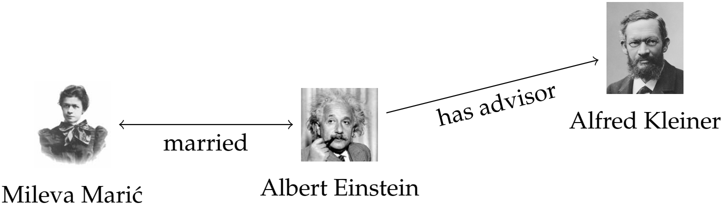
What is a knowledge base?


Mileva Marić

married

Albert Einstein

What is a knowledge base?



Mileva Marić

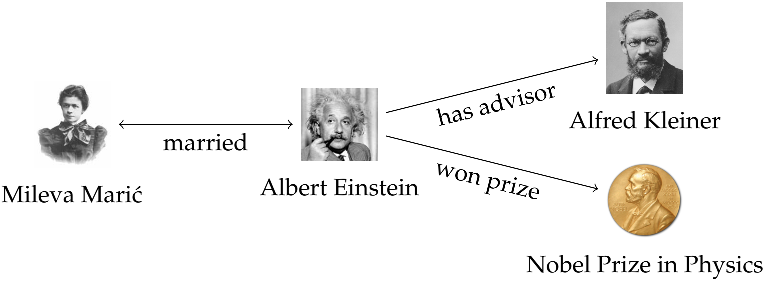
married

Albert Einstein

has advisor

Alfred Kleiner

What is a knowledge base?



Mileva Marić

married

Albert Einstein

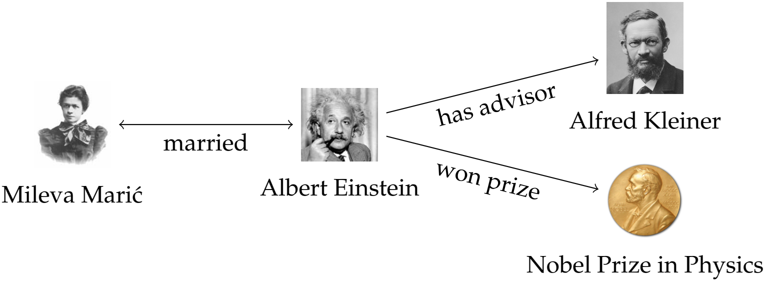
has advisor

Alfred Kleiner

won prize

Nobel Prize in Physics

What is a knowledge base?


Mileva Marić

married

Albert Einstein

has advisor

Alfred Kleiner

won prize

Nobel Prize in Physics

Applications of knowledge bases

- question answering
- semantic search
- text analysis
- machine translation

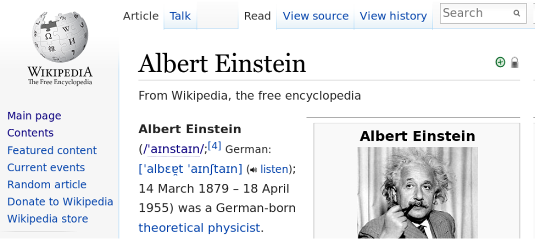
What is YAGO?

- knowledge base with 10 million entities and >210 million facts
- automatically extracted from Wikipedia, Wordnet, and Geonames
- multilingual facts from 10 languages
- focus on precision
- developed by Max-Planck Institute for Informatics and Télécom ParisTech

What is YAGO?

- I joined the project in 2015
- coordinated / contributed to the evaluation
- maintenance, participating in open source release
- development

What is YAGO?



WIKIPEDIA
The Free Encyclopedia

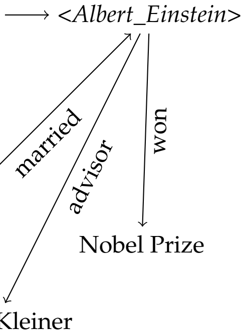
- Main page
- Contents
- Featured content
- Current events
- Random article
- Donate to Wikipedia
- Wikipedia store

Albert Einstein

From Wikipedia, the free encyclopedia

Albert Einstein (/ˈaɪnstaɪn/;[4] German: [ˈalbɛɐ̯t ˈaɪnʃtaɪn] (listen); 14 March 1879 – 18 April 1955) was a German-born theoretical physicist.

Albert Einstein



<Albert_Einstein>

married

advisor

won

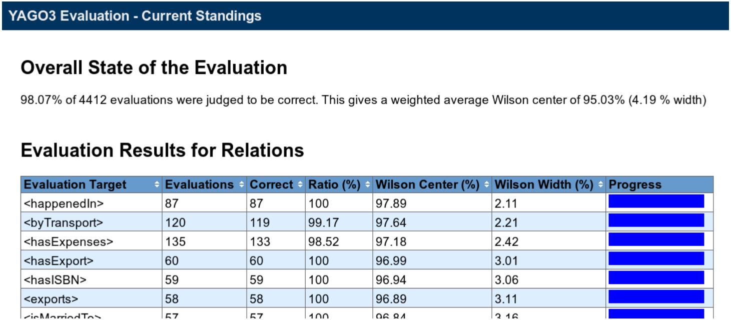
Mileva

Nobel Prize

Alfred Kleiner

yago
select knowledge

Accuracy



YAGO3 Evaluation - Current Standings

Overall State of the Evaluation

98.07% of 4412 evaluations were judged to be correct. This gives a weighted average Wilson center of 95.03% (4.19 % width)

Evaluation Results for Relations

Evaluation Target	Evaluations	Correct	Ratio (%)	Wilson Center (%)	Wilson Width (%)	Progress
<happenedIn>	87	87	100	97.89	2.11	█
<byTransport>	120	119	99.17	97.64	2.21	█
<hasExpenses>	135	133	98.52	97.18	2.42	█
<hasExport>	60	60	100	96.99	3.01	█
<hasISBN>	59	59	100	96.94	3.06	█
<exports>	58	58	100	96.89	3.11	█
<isMarriedTo>	57	57	100	96.84	3.16	█

Figure: Screenshot of evaluation result

- 2 months evaluation, 15 participants
- evaluated 4412 facts of 76 relations (with 60m total facts)
- 98% facts of the sample were correct
- Wilson center: 95%, interval width: 4.2%

Now that we have this knowledge base, what can we do with it?

Using YAGO for the Humanities: Related Work

Similar studies using Semantic Web for Digital Humanities

- [Schich et al., 2014]: about 150,000 people
- [de la Croix et al., 2015]: about 300,000 people
- [Gergaud et al., 2017]: about 1,100,000 people

These studies are only about few people. Can we do better with YAGO?

Using YAGO for the Humanities: Related Work

Similar studies using Semantic Web for Digital Humanities

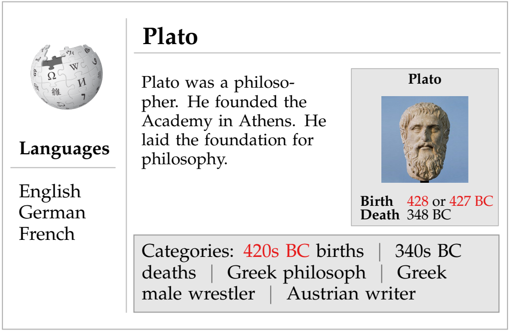
- [Schich et al., 2014]: about 150,000 people
- [de la Croix et al., 2015]: about 300,000 people
- [Gergaud et al., 2017]: about 1,100,000 people

These studies are only about few people. Can we do better with YAGO?

YAGO has 2,200,000 people, but, e.g., locations only for 700,000 people
How can we make YAGO more complete?

Using YAGO for the Humanities: Birth and Death Dates

Previous algorithm:



Languages

English
German
French

Plato

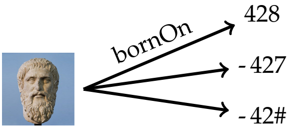
Plato was a philosopher. He founded the Academy in Athens. He laid the foundation for philosophy.

Plato

Birth 428 or 427 BC
Death 348 BC

Categories: 420s BC births | 340s BC deaths | Greek philosoph | Greek male wrestler | Austrian writer

Extracted birth dates



bornOn

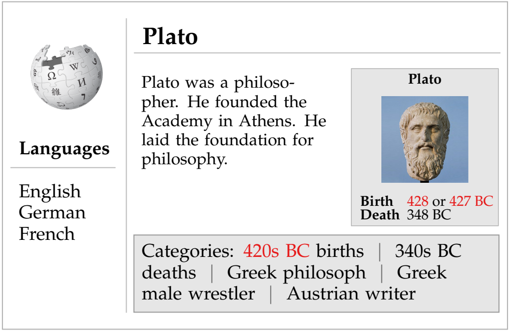
428

-427

-42#

Using YAGO for the Humanities: Birth and Death Dates

Previous algorithm:



Languages

English
German
French

Plato

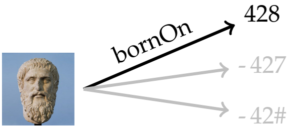
Plato was a philosopher. He founded the Academy in Athens. He laid the foundation for philosophy.

Plato

Birth 428 or 427 BC
Death 348 BC

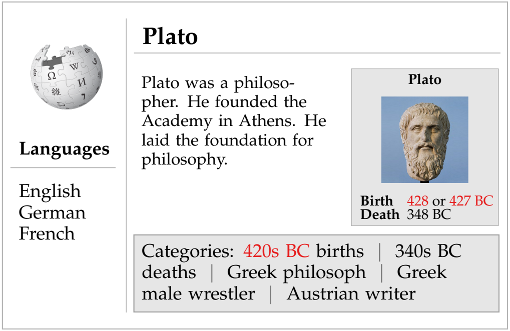
Categories: 420s BC births | 340s BC deaths | Greek philosoph | Greek male wrestler | Austrian writer

Extracted birth dates



Using YAGO for the Humanities: Birth and Death Dates

New algorithm: filtering with category dates



Languages

English
German
French

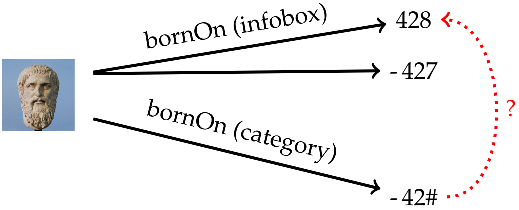
Plato

Plato was a philosopher. He founded the Academy in Athens. He laid the foundation for philosophy.

Plato

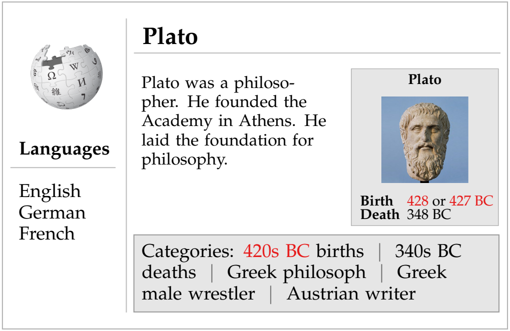
Birth 428 or 427 BC
Death 348 BC

Categories: 420s BC births | 340s BC deaths | Greek philosoph | Greek male wrestler | Austrian writer



Using YAGO for the Humanities: Birth and Death Dates

New algorithm: filtering with category dates



Languages

English
German
French

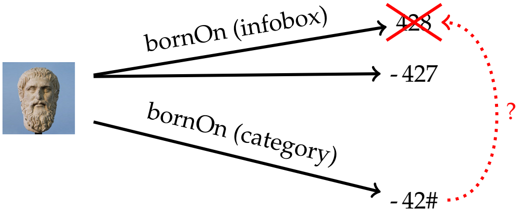
Plato

Plato was a philosopher. He founded the Academy in Athens. He laid the foundation for philosophy.

Plato

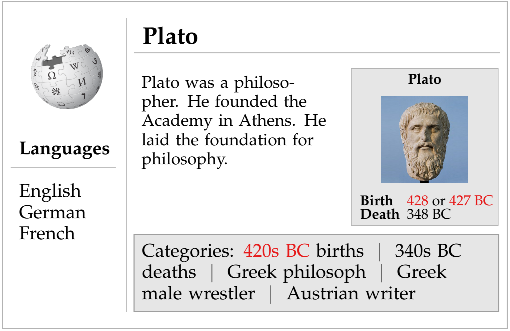
Birth 428 or 427 BC
Death 348 BC

Categories: 420s BC births | 340s BC deaths | Greek philosoph | Greek male wrestler | Austrian writer



Using YAGO for the Humanities: Birth and Death Dates

New algorithm: filtering with category dates



Plato

Plato was a philosopher. He founded the Academy in Athens. He laid the foundation for philosophy.

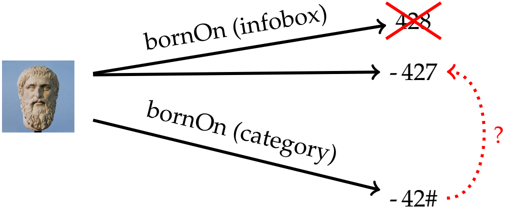
Plato

Birth 428 or 427 BC
Death 348 BC

Categories: 420s BC births | 340s BC deaths | Greek philosoph | Greek male wrestler | Austrian writer

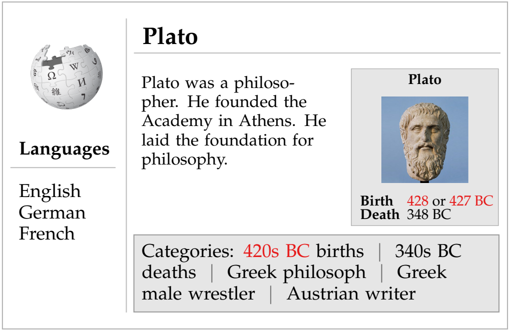
Languages

English
German
French



Using YAGO for the Humanities: Birth and Death Dates

New algorithm: filtering with category dates



Languages

English
German
French

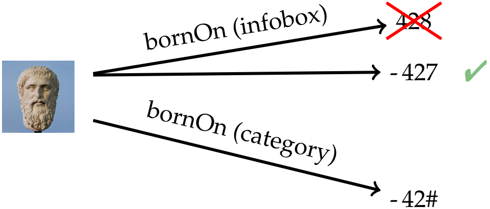
Plato

Plato was a philosopher. He founded the Academy in Athens. He laid the foundation for philosophy.

Plato

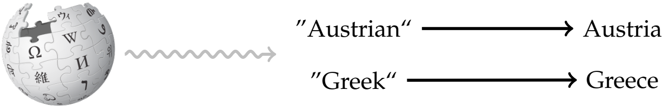
Birth 428 or 427 BC
Death 348 BC

Categories: 420s BC births | 340s BC deaths | Greek philosoph | Greek male wrestler | Austrian writer



Using YAGO for the Humanities: Place of residence

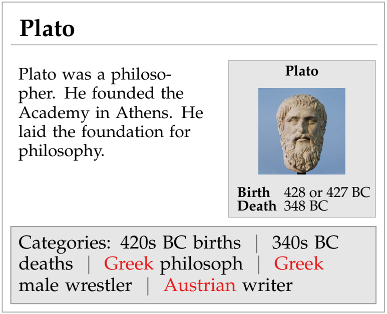
Extract mapping from demonyms / adjectives to locations



"Austrian" → Austria

"Greek" → Greece

Take most frequent location as place of residence



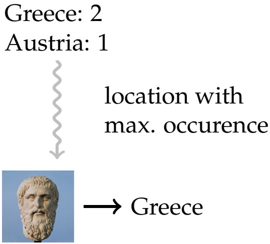
Plato

Plato was a philosopher. He founded the Academy in Athens. He laid the foundation for philosophy.

Plato

Birth 428 or 427 BC
Death 348 BC

Categories: 420s BC births | 340s BC deaths | Greek philosoph | Greek male wrestler | Austrian writer



Greece: 2
Austria: 1

location with max. occurence

→ Greece

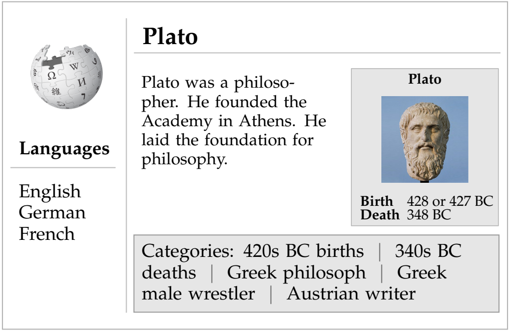
Using YAGO for the Humanities: Place of residence

Caveat: only take outermost text spans

″Holy Roman Empire“	→	*<Holy_Roman_Empire>*
″Roman Empire“	→	*<Roman_Empire>*

Using YAGO for the Humanities: Gender

Previous algorithm:



Languages

English
German
French

Plato

Plato was a philosopher. He founded the Academy in Athens. He laid the foundation for philosophy.

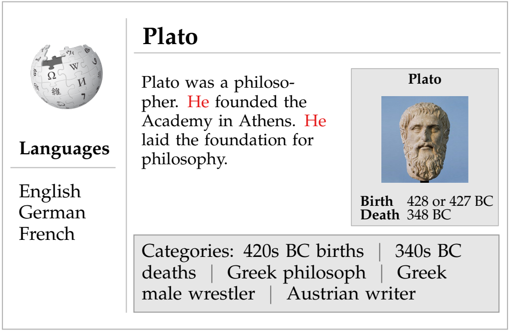
Plato

Birth	428 or 427 BC
Death	348 BC

Categories: 420s BC births | 340s BC deaths | Greek philosoph | Greek male wrestler | Austrian writer

Using YAGO for the Humanities: Gender

Previous algorithm:



Languages

English
German
French

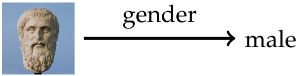
Plato

Plato was a philosopher. He founded the Academy in Athens. He laid the foundation for philosophy.

Plato	
Birth	428 or 427 BC
Death	348 BC

Categories: 420s BC births | 340s BC deaths | Greek philosoph | Greek male wrestler | Austrian writer

Extracted gender

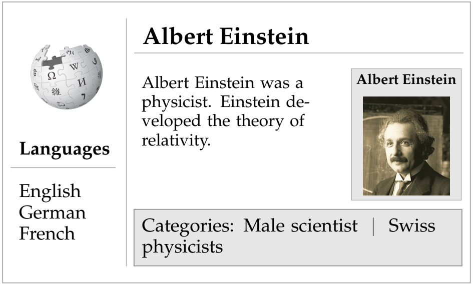


gender

male

Using YAGO for the Humanities: Gender

New algorithm:



Languages

English
German
French

Albert Einstein

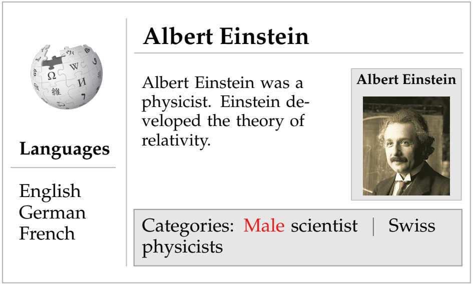
Albert Einstein was a physicist. Einstein developed the theory of relativity.

Albert Einstein

Categories: Male scientist | Swiss physicists

Using YAGO for the Humanities: Gender

New algorithm:



Languages

English
German
French

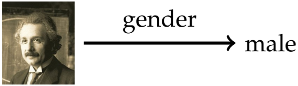
Albert Einstein

Albert Einstein was a physicist. Einstein developed the theory of relativity.

Albert Einstein

Categories: Male scientist | Swiss physicists

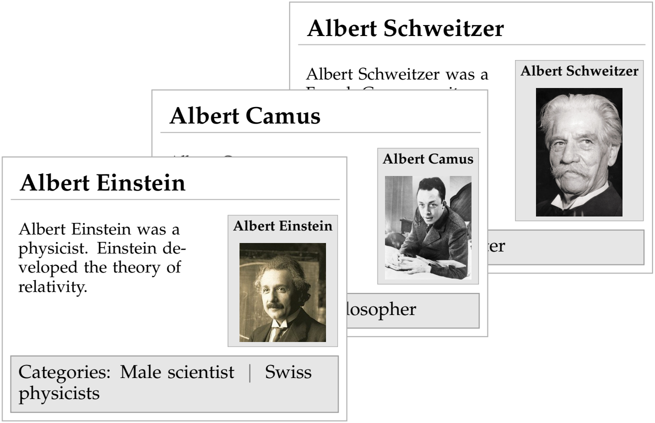
Extracted gender



gender

male

Using YAGO for the Humanities: Gender



Albert Schweitzer

Albert Schweitzer was a

er

Albert Camus

losopher

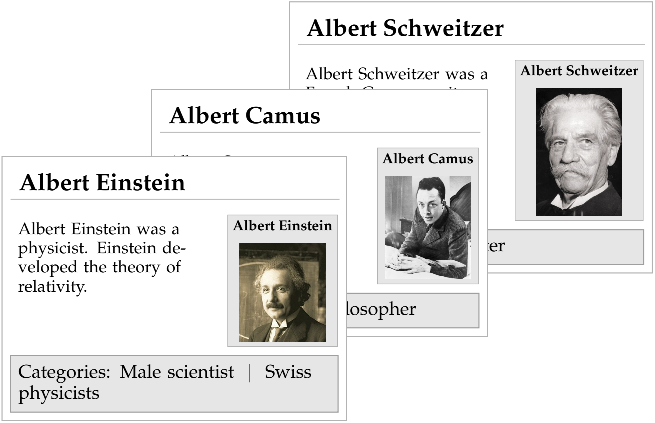
Albert Einstein

Albert Einstein was a physicist. Einstein developed the theory of relativity.

Categories: Male scientist | Swiss physicists

$$\text{"Albert"} \longrightarrow \text{male}$$

Using YAGO for the Humanities: Gender



Albert Schweitzer

Albert Schweitzer was a

Albert Schweitzer

er

Albert Camus

Albert Camus

losopher

Albert Einstein

Albert Einstein was a physicist. Einstein developed the theory of relativity.

Albert Einstein

Categories: Male scientist | Swiss physicists

"Albert" ⟶ male

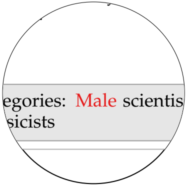
"Francesca" ⟶ female

"Kathleen" ⟶ female

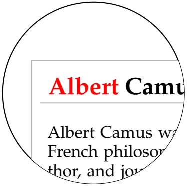
in total:
1206 first names

Using YAGO for the Humanities: Gender

Prioritize extracted facts

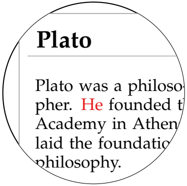

egories: Male scientis
sicists

1. **extract gender by category**


Albert Camu

Albert Camus wa
French philosop
thor, and jou

2. **extract gender by first name**


Plato

Plato was a philoso-
pher. He founded t
Academy in Athen
laid the foundatio
philosophy.

3. **extract gender by pronoun**

Using YAGO for the Humanities: Evaluation

- compare extraction process on Wikipedia dump from 2017-02-20
- extracted on 11 languages
- evaluate precision based on a sample of 100 people

Extraction	YAGO before	Recall	YAGO now	Recall	Precision	DBpedia (en)
Birth dates	1.6m	69%	1.7m	74% (+8%)	100%	0.8m
Death dates	0.7m	33%	0.8m	36% (+10%)	100%	0.3m
Place of residence	0.7m	30%	2.1m	91% (+201%)	97% (*)	0.7m
Gender	1.5m	64%	2.0m	87% (+35%)	98%	4k

Table: Coverage and precision of our methods.
Recall relative to total number of people in YAGO (2.2m).

m million k thousand

(*) 6% of anachronistic residencies (e.g., German Empire instead of Germany)

Using YAGO for the Humanities: Life expectancy over time

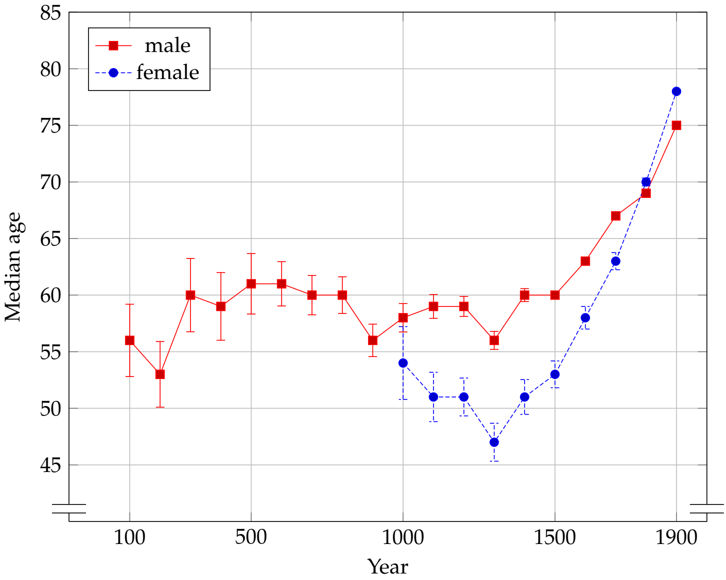


Figure: Median age over time, by year of birth

Using YAGO for the Humanities: Life expectancy over time

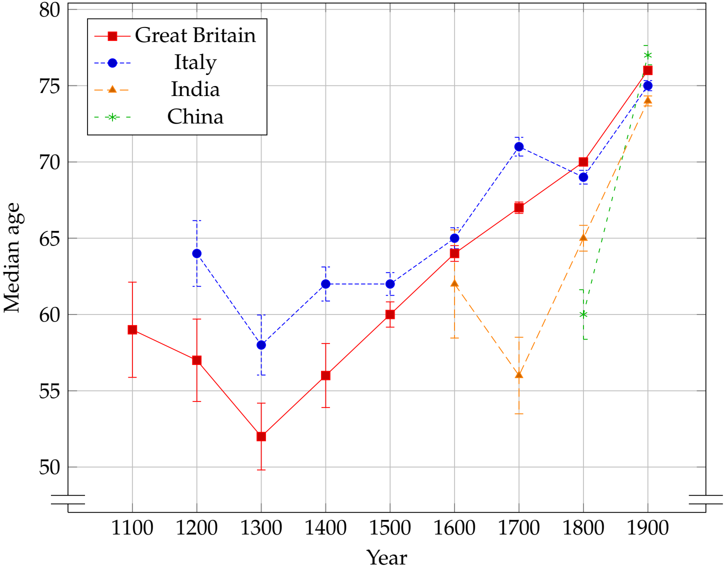


Figure: Median age over time, by year of birth

Using YAGO for the Humanities: Births per month

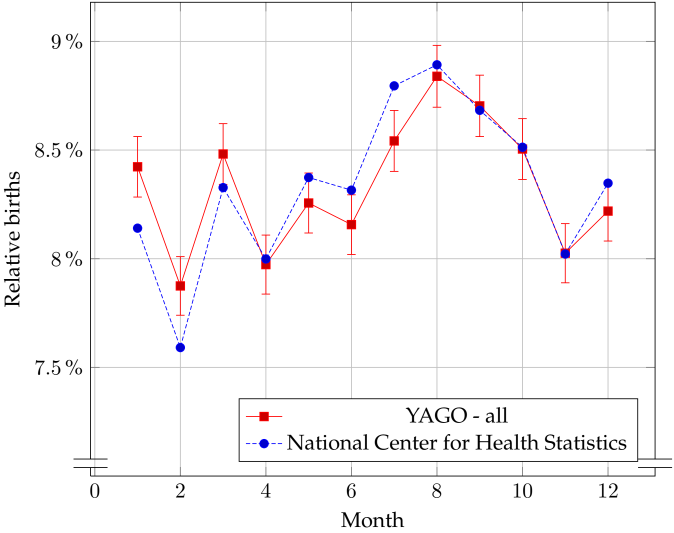


Figure: Births per month in the United States between 2003 and 2015 (with the Student's t confidence interval at $\alpha = 95\%$).

Using YAGO for the Humanities: Births per month

Possible explanation:

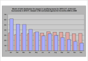
Languages

English
Euskara

Relative age effect

The relative age effect describes a bias. People born early in the selection period of sports or academia are more likely to perform well.

Relative age effect



Categories: Ageism | Epidemiology

Using YAGO for the Humanities: Births per month

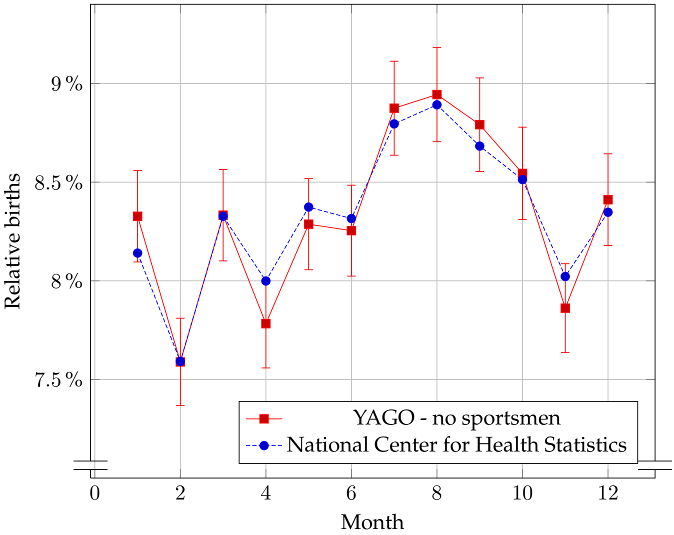


Figure: Births per month in the United States between 2003 and 2015 (with the Student's t confidence interval at $\alpha = 95\%$).

Using YAGO for the Humanities: Relative population size

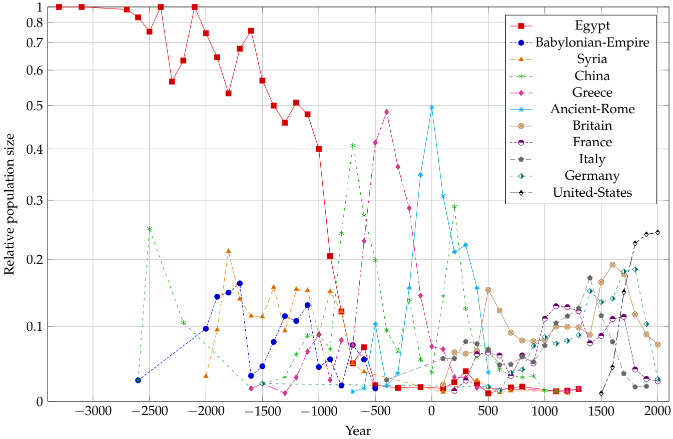
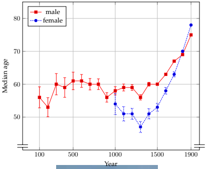


Figure: Relative population size, by century. The *y*-axis is scaled by a quadratic function.

Using YAGO for the Humanities: Summary

- extension of YAGO
  - more birth and death dates (+8%/10%, 100% precision)
  - more people with locations (+201%, 97% precison)
  - more people with genders (+35%, 98% precision)
- case studies
  - life expectancy
  - births per month
  - relative population size



Thomas Rebele

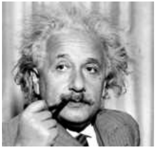
Arash Nekoei

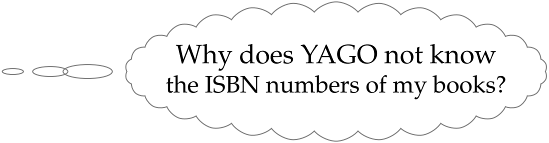
Fabian Suchanek

publication: ISWC 2017 (workshop paper)

We often had to repair regular expressions (e.g., for matching dates). Can we automate this step?

Adding Words to Regexes: Introduction

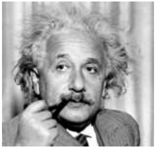


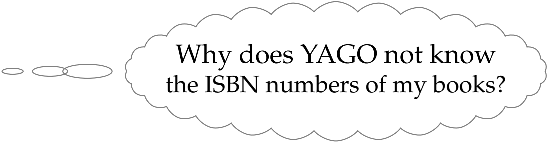


Why does YAGO not know the ISBN numbers of my books?

- we want to find ISBN numbers in Wikipedia to include it in YAGO
- we try the regex `ISBN(978|979)?\d{10}`

Adding Words to Regexes: Introduction





Why does YAGO not know
the ISBN numbers of my books?

- we want to find ISBN numbers in Wikipedia to include it in YAGO
- we try the regex `ISBN(978|979)?\d{10}`
- why does the regex not find `I978-2-1234-5680-3` ?
- how can we modify the regex automatically to match the word?

Adding Words to Regexes: Problem statement

Problem statement, first try:

Given

- a regular expression r and
- a set of strings S,

find a regular expression r' such that

- $L(r) \subseteq L(r')$
- $S \subseteq L(r')$

```
ISBN(978|979)?\d{10}
```

$\{$ `I978-2-1234-5680-3` $\}$

Adding Words to Regexes: Problem statement

Problem statement, first try:

Given

- a regular expression r and
- a set of strings S,

find a regular expression r' such that

- $L(r) \subseteq L(r')$
- $S \subseteq L(r')$

```
ISBN(978|979)?\d{10}
```

$\{$ `I978-2-1234-5680-3` $\}$

Solution:

$r' = .^*$

Adding Words to Regexes: Problem statement

Problem statement:

Given

- a regular expression r,
- a set of strings S,
- a set of negative examples E^-,

find a regular expression r' such that

- $L(r) \subseteq L(r')$
- $S \subseteq L(r')$
- $L(r') \cap E^-$ is small

```
ISBN(978|979)?\d{10}
{ I978-2-1234-5680-3 }
{ 0612345678 }
```

Adding Words to Regexes: Problem statement

Problem statement:

Given

- a regular expression r,
- a set of strings S,
- a set of negative examples E^-,

find a regular expression r' such that

- $L(r) \subseteq L(r')$
- $S \subseteq L(r')$
- $L(r') \cap E^-$ is small

`ISBN(978|979)?\d{10}`

$\{$ `I978-2-1234-5680-3` $\}$

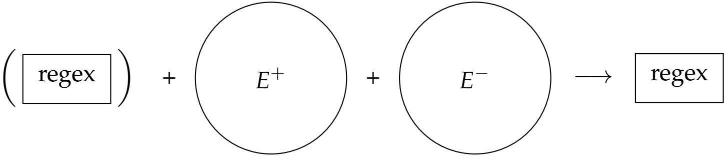
$\{$ `0612345678` $\}$

Additional goals:

- precision of $r' \geq$ or $\approx$ precision of r
- recall of $r' \geq$ recall of r
  (w.r.t. the intended meaning of the regex)

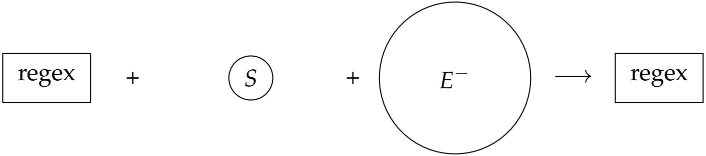
Adding Words to Regexes: What is new in our approach

Previous approaches



$$\left(\boxed{\text{regex}}\right) + E^{+} + E^{-} \longrightarrow \boxed{\text{regex}}$$

Our approach

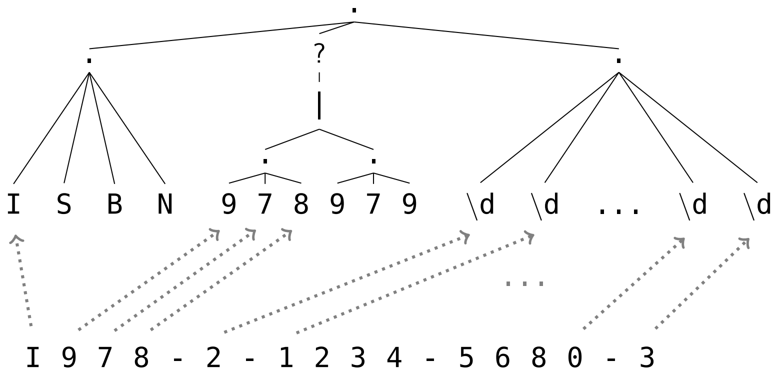


$$\boxed{\text{regex}} + S + E^{-} \longrightarrow \boxed{\text{regex}}$$

Rationale: creating a large set of positive examples is difficult

Adding Words to Regexes: Approximate regex matching

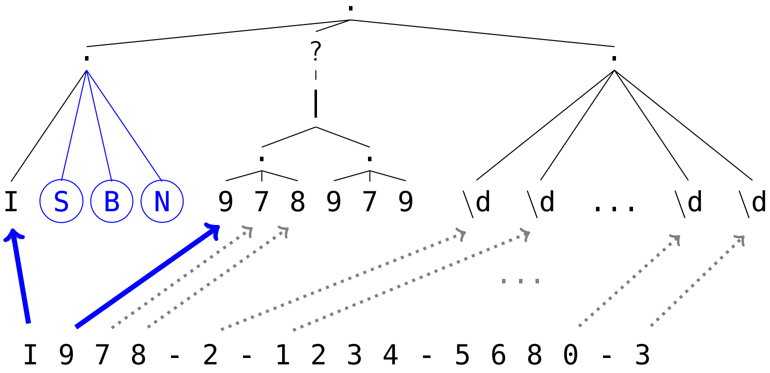
Step 1: match string and regex approximately [Myers et al. 1989]



Adding Words to Regexes: Finding the gaps

Step 2: find the gaps

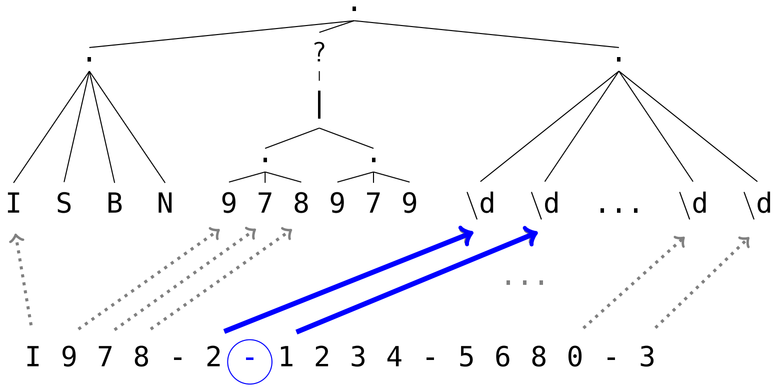
- between regex leaves



Adding Words to Regexes: Finding the gaps

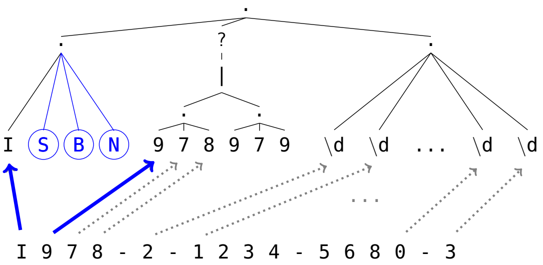
Step 2: find the gaps

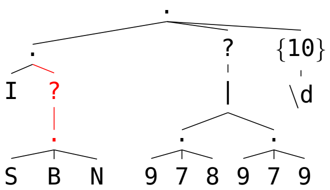
- between regex leaves
- between characters of the string



Adding Words to Regexes: Add missing parts

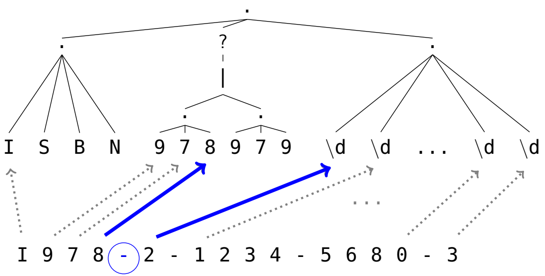
Step 3 (simple approach): adapt regex, so that it includes the missing parts

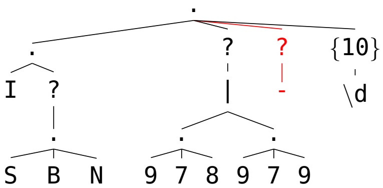




Adding Words to Regexes: Add missing parts

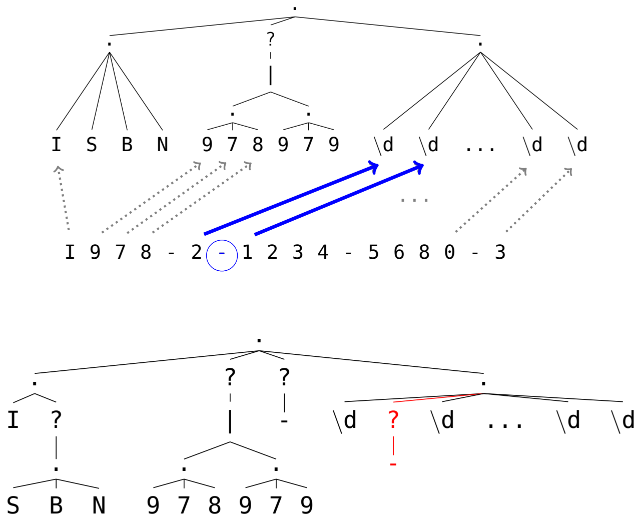
Step 3 (simple approach): adapt regex, so that it includes the missing parts





Adding Words to Regexes: Add missing parts

Step 3 (simple approach): adapt regex, so that it includes the missing parts



Adding Words to Regexes: Add missing parts

The YAGO knowledge base

Using YAGO for the Humanities

Adding Words to Regexes

Introduction

Problem statement

What is new in our approach

Approximate regex matching

Finding the gaps

Add missing parts

Feedback function

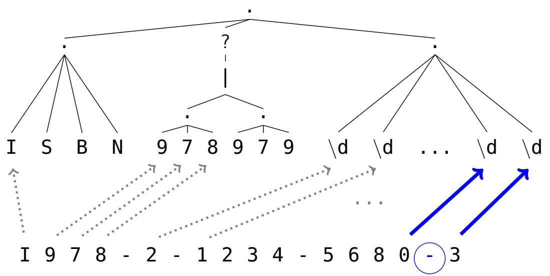
Experiments

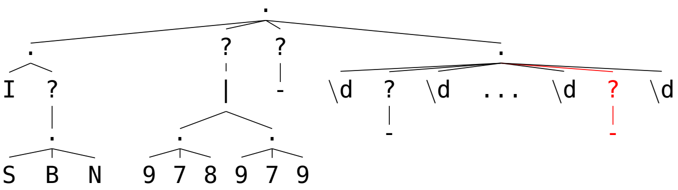
Summary

Answering Queries with Unix Shell

Conclusion

Step 3 (simple approach): adapt regex, so that it includes the missing parts

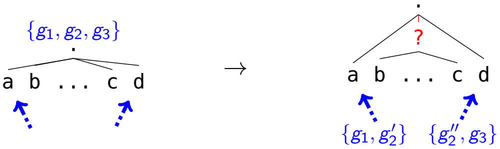




Adding Words to Regexes: Add missing parts

Step 3 (adaptive approach): adapt regex, so that it includes the missing parts

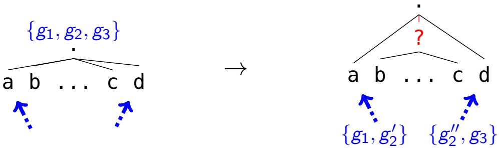
Cases:

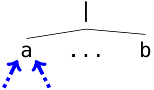


Adding Words to Regexes: Add missing parts

Step 3 (adaptive approach): adapt regex, so that it includes the missing parts

Cases:



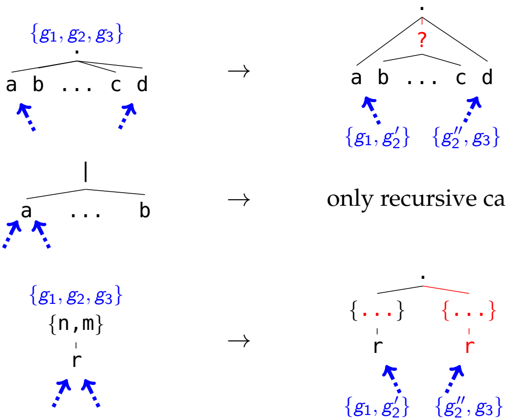


→ only recursive call

Adding Words to Regexes: Add missing parts

Step 3 (adaptive approach): adapt regex, so that it includes the missing parts

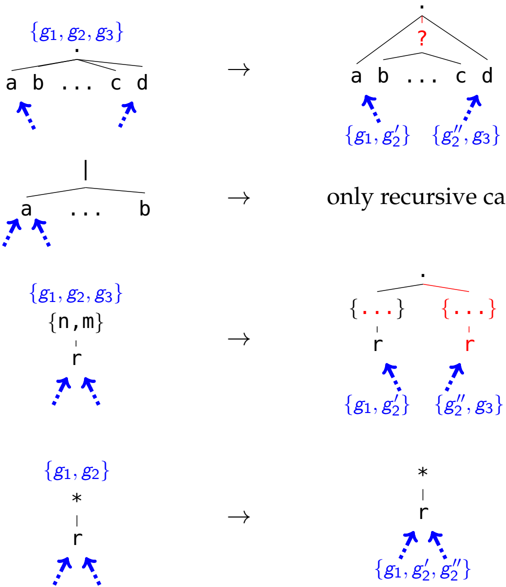
Cases:



Adding Words to Regexes: Add missing parts

Step 3 (adaptive approach): adapt regex, so that it includes the missing parts

Cases:



Adding Words to Regexes: Feedback function

Example

- now we want to find URLs
- we try regex $r =$ `http://[a-zA-Z\.]+`
- it does not find $s =$ `wikipedia.org`
- repaired regex $r' =$ `(http://)?[a-zA-Z\.]+`

- problem: r' finds all words
- precision drops

Adding Words to Regexes: Feedback function

Example

- now we want to find URLs
- we try regex $r =$ `http://[a-zA-Z\.]+`
- it does not find $s =$ `wikipedia.org`
- repaired regex $r' =$ `(http://)?[a-zA-Z\.]+`

- problem: r' finds all words
- precision drops

Solution: use feedback on set of negative examples E^-

- determine the parts of the regex that we can make optional
- we use the number of false positives, i.e.,

$$f(r') = |E^- \cap L(r')| \leq \alpha|E^- \cap L(r)|$$

- if $f(r') =$ false, add the word as disjunction instead: `http://[a-zA-Z\.]+|wikipedia.org`

Adding Words to Regexes: Feedback function

Summary of the algorithm:

1. match strings in S approximately to r
2. find gaps in the regex or in the strings
3. (adaptive:) find overlaps within the gaps
4. (simple:) add missing parts for every missing word one after the other

   (adaptive:) add missing parts and check intermediate steps with the feedback
5. (adaptive:) add a generalization of non-repaired words (similar to [Babbar et al. 2010])

Adding Words to Regexes: Experiments

Input data

- datasets:
  ReLIE [Li et al., 2008],
  Enron [Babbar et al., 2010], and
  YAGO infobox attributes
- in total 8 tasks
- in total 52 regexes

Experimental approach

- 5×2 train/test split
- missing words S are selected randomly from $E^+ \setminus L(r)$, $|S| \leq 10$
- we draw 10 different sets S

Adding Words to Regexes: Experiments

Baselines

- dis: $r|s_1|\cdots|s_n$
- star: `.*`

Competitors

- B&S: [Babbar et al., 2010] (reimplementation)
- simple
- adaptive

		baseline				adaptive		
measure	original	dis	star	B&S	simple	$\alpha = 1.0$	$\alpha = 1.1$	$\alpha = 1.20$
F1	55	55	21	40	56	**60**	**60**	**60**
recall	66	67	62	35	69	75	76	**77**
precision	64	64	14	**71**	64	63	63	63
length	56	270	2	3929	250	**76**	80	81

Table: Averaged measures for the different systems. Length is # of characters of the regex.

Adding Words to Regexes: Summary

Summary

- algorithm for adding missing words to regexes
- increases recall, while keeping precision stable
- Source code available at `https://github.com/thomasrebele/regex-repair`

Future work

- decrease dependency on E^-
- add a generalization step as postprocessing

Thomas Rebele

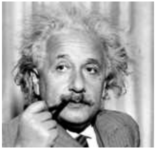
Katerina Tzompanaki

Fabian Suchanek

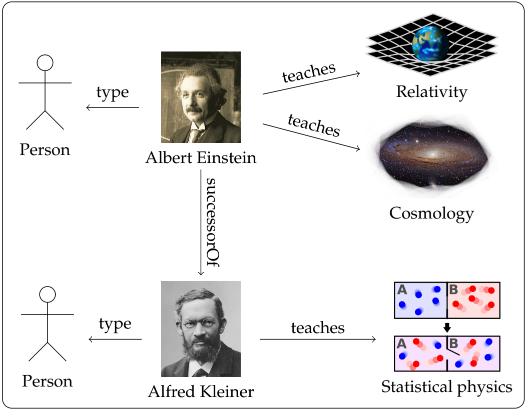
publications: ISWC 2017 (demo), PAKDD 2018 (full paper)

Now that we have all this data, how can we process it efficiently?

Answering Queries with Unix Shell: Motivation

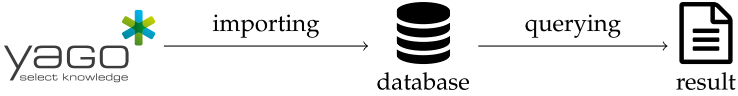


How can I find all teachers?



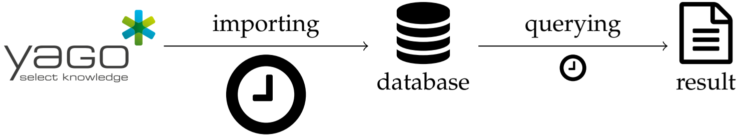
Answering Queries with Unix Shell: Motivation

Observation:



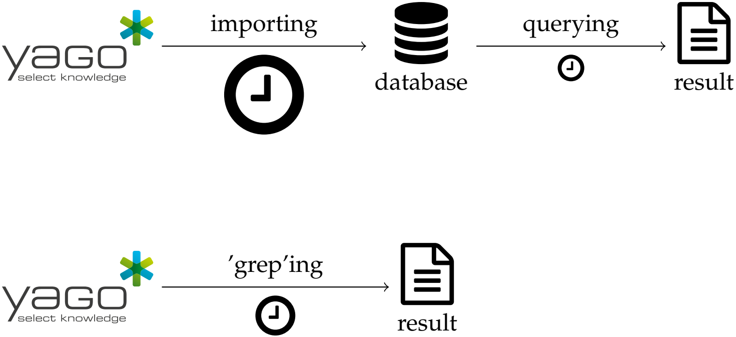
Answering Queries with Unix Shell: Motivation

Observation:

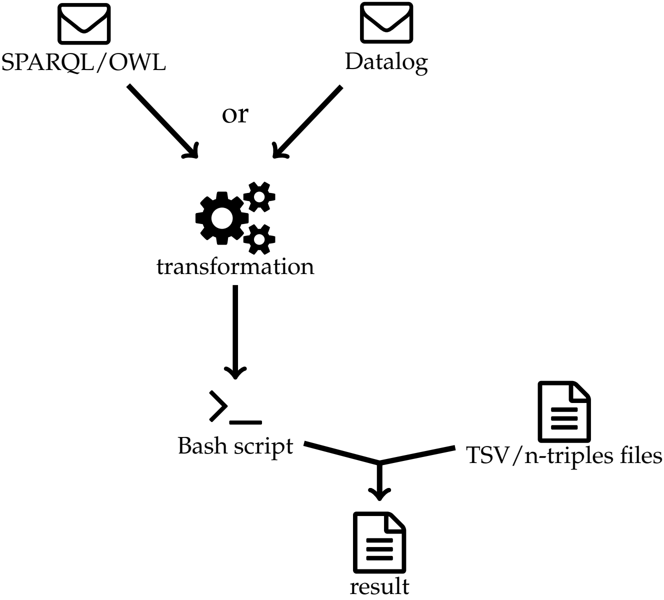


Answering Queries with Unix Shell: Motivation

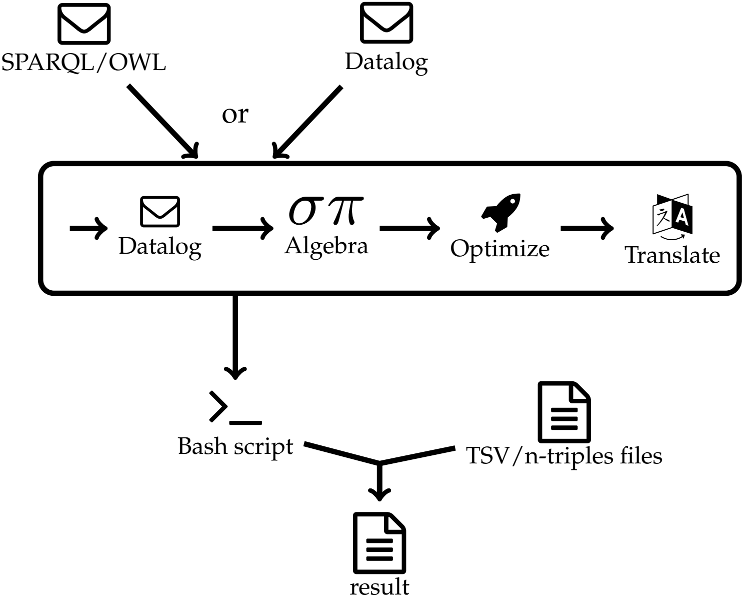
Observation:



Answering Queries with Unix Shell: System



Answering Queries with Unix Shell: System



Answering Queries with Unix Shell: Approach

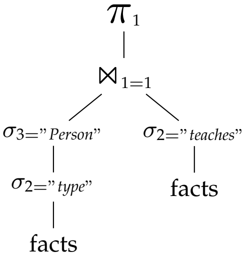
Query "Which people teach a course?" in SPARQL

```
SELECT ?X WHERE {
    ?X <type> <Person>.
    ?X <teachesCourse> ?Y.
}
```

Translating the query to Datalog

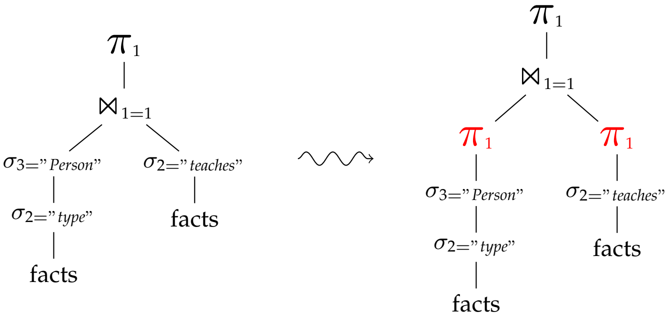
```
Person(X) :-
      facts(X, "type", "Person").
teaches(X, Y) :-
      facts(X, "teaches", Y).

Teacher(X) :- Person(X),
              teaches(X,Y).
```


π_1
$\bowtie_{1=1}$
$\sigma_{3=\text{"}Person\text{"}}$
$\sigma_{2=\text{"}teaches\text{"}}$
$\sigma_{2=\text{"}type\text{"}}$
facts
facts

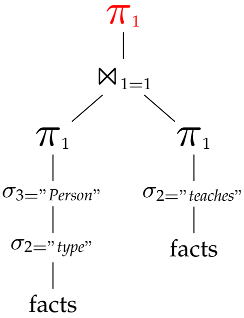
Answering Queries with Unix Shell: Approach

Optimization:


π_1
$\bowtie_{1=1}$
$\sigma_{3=\text{"}Person\text{"}}$
$\sigma_{2=\text{"}teaches\text{"}}$
$\sigma_{2=\text{"}type\text{"}}$
facts
facts
$\rightsquigarrow$
π_1
$\bowtie_{1=1}$
π_1
π_1
$\sigma_{3=\text{"}Person\text{"}}$
$\sigma_{2=\text{"}teaches\text{"}}$
$\sigma_{2=\text{"}type\text{"}}$
facts
facts

Answering Queries with Unix Shell: Approach

Algebra plan



π_1

$\bowtie_{1=1}$

π_1 π_1

$\sigma_{3=\text{"}\textit{Person}\text{"}}$ $\sigma_{2=\text{"}\textit{teaches}\text{"}}$

$\sigma_{2=\text{"}\textit{type}\text{"}}$ facts

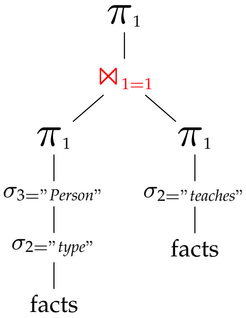
facts

Bash code

```
sort -u \
<(join -1 1 -2 1 -o 1.1 \
  <(sort -k 1 \
    <(awk ’($3 == "Person" && $2 == "type")
           { print $1 };’ facts))
  <(sort -k 1 \
    <(awk ’($2 == "teaches")
           { print $1 };’ facts))
```

Answering Queries with Unix Shell: Approach

Algebra plan



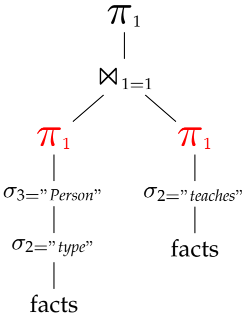
π_1
$\bowtie_{1=1}$
π_1
π_1
$\sigma_{3=\text{"}Person\text{"}}$
$\sigma_{2=\text{"}teaches\text{"}}$
$\sigma_{2=\text{"}type\text{"}}$
facts
facts

Bash code

```
sort -u \
<(join -1 1 -2 1 -o 1.1 \
  <(sort -k 1 \
    <(awk ’($3 == "Person" && $2 == "type")
           { print $1 };’ facts))
  <(sort -k 1 \
    <(awk ’($2 == "teaches")
           { print $1 };’ facts))
```

Answering Queries with Unix Shell: Approach

Algebra plan

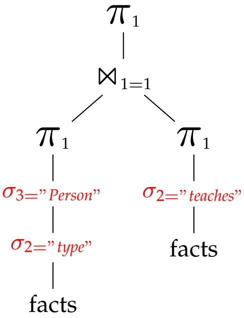

π_1
$\bowtie_{1=1}$
π_1
π_1
$\sigma_{3=\text{"}Person\text{"}}$
$\sigma_{2=\text{"}teaches\text{"}}$
$\sigma_{2=\text{"}type\text{"}}$
facts
facts

Bash code

```
sort -u \
<(join -1 1 -2 1 -o 1.1 \
  <(sort -k 1 \
    <(awk ’($3 == "Person" && $2 == "type")
          { print $1 };’ facts))
  <(sort -k 1 \
    <(awk ’($2 == "teaches")
          { print $1 };’ facts))
```

Answering Queries with Unix Shell: Approach

Algebra plan

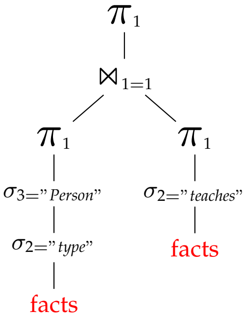

π_1
$\bowtie_{1=1}$
π_1
π_1
$\sigma_{3=\text{"}Person\text{"}}$
$\sigma_{2=\text{"}teaches\text{"}}$
$\sigma_{2=\text{"}type\text{"}}$
facts
facts

Bash code

```
sort -u \
<(join -1 1 -2 1 -o 1.1 \
  <(sort -k 1 \
    <(awk '($3 == "Person" && $2 == "type")
           { print $1 };' facts))
  <(sort -k 1 \
    <(awk '($2 == "teaches")
           { print $1 };' facts))
```

Answering Queries with Unix Shell: Approach

Algebra plan


π_1
$\bowtie_{1=1}$
π_1
π_1
$\sigma_{3=\text{"}\mathit{Person}\text{"}}$
$\sigma_{2=\text{"}\mathit{teaches}\text{"}}$
$\sigma_{2=\text{"}\mathit{type}\text{"}}$
facts
facts

Bash code

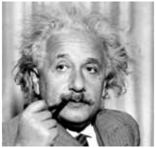
```
sort -u \
<(join -1 1 -2 1 -o 1.1 \
  <(sort -k 1 \
    <(awk ’($3 == "Person" && $2 == "type")
           { print $1 };’ facts))
  <(sort -k 1 \
    <(awk ’($2 == "teaches")
           { print $1 };’ facts))
```

Answering Queries with Unix Shell: Optimization

Optimizations

- algebraic, e.g., merge union / projects
- semi-naive evaluation
- join reordering
- remove superfluous recursive calls
- materialize repeated subplans
- read files only once
- tweak Unix commands, e.g., using `LANG=C` and MAWK

Answering Queries with Unix Shell: Optimization



How can I find all professors?

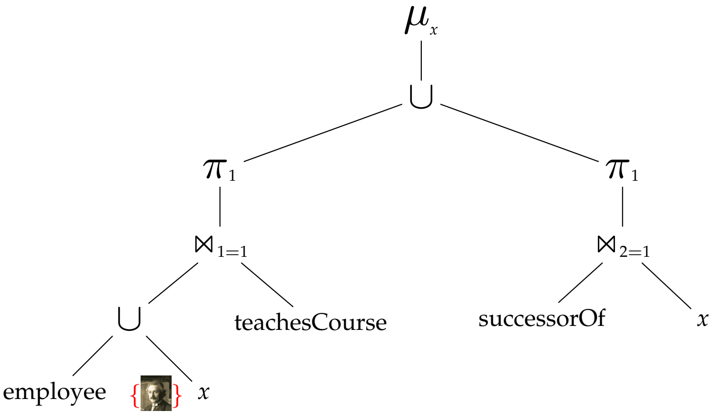
```
Professor(X) :- Person(X),
                teachesCourse(X,Y).
Professor(X) :- successorOf(X,Y),
                Professor(Y).

Person(X) :- Employee(X).
Person(X) :- Professor(X).
```

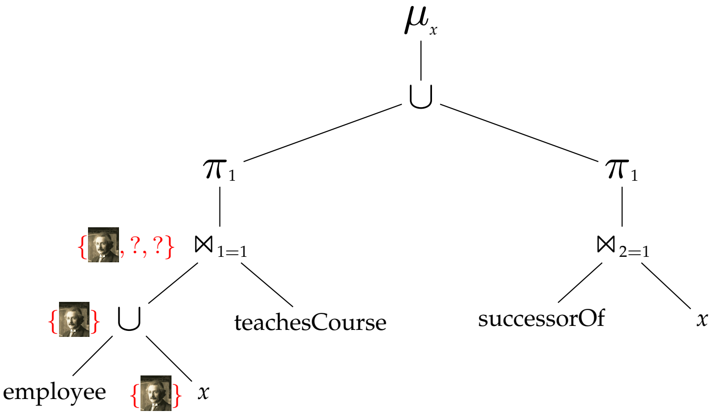
Combining the first and the last rule leads to

```
Professor(X) :- Professor(X),
                teachesCourse(X,Y).
```

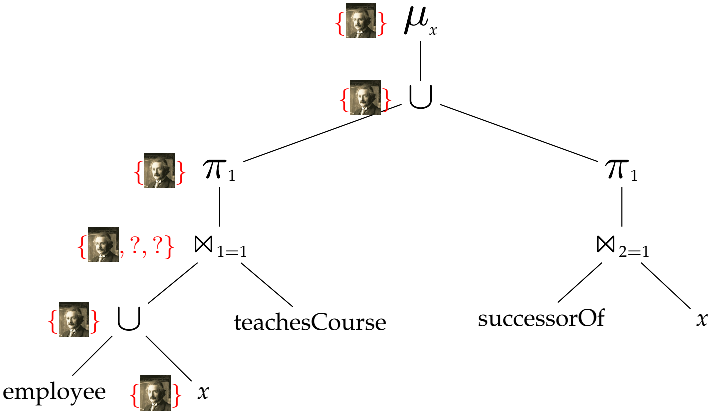
Answering Queries with Unix Shell: Optimization



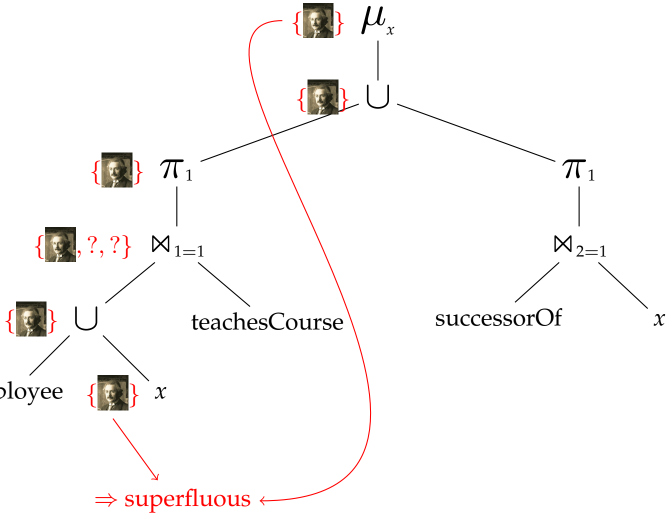
Answering Queries with Unix Shell: Optimization



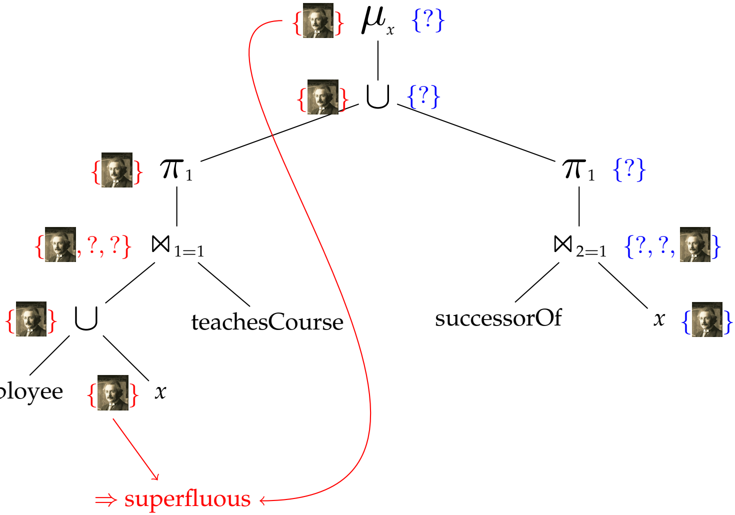
Answering Queries with Unix Shell: Optimization



Answering Queries with Unix Shell: Optimization

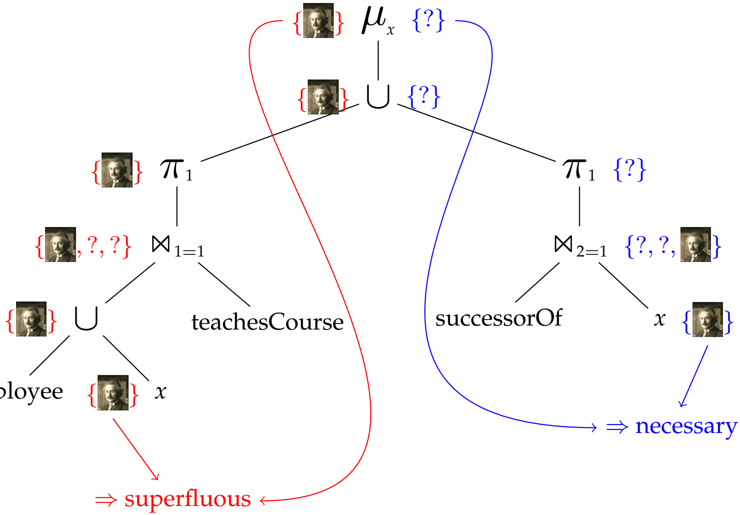


Answering Queries with Unix Shell: Optimization

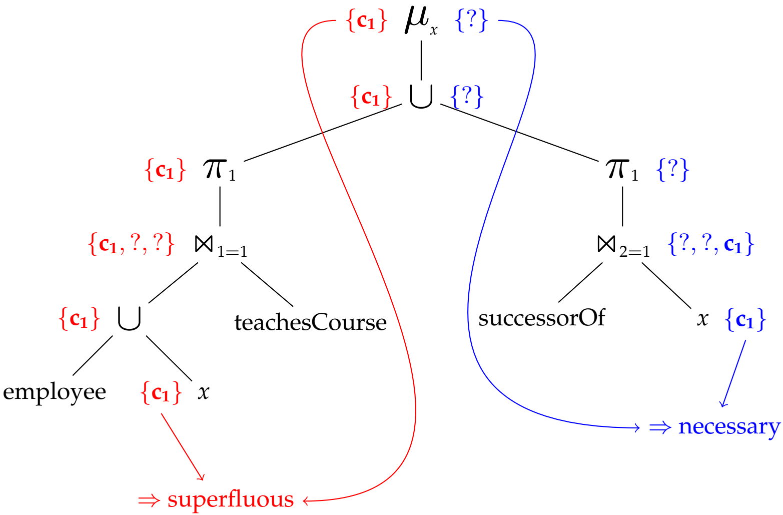


⇒ superfluous

Answering Queries with Unix Shell: Optimization

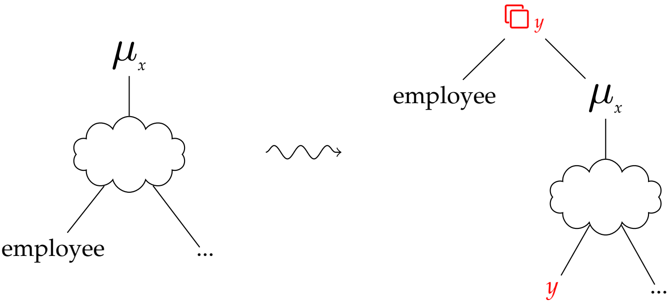


Answering Queries with Unix Shell: Optimization



Answering Queries with Unix Shell: Optimization

Assume that *employee* is an expensive subplan



Answering Queries with Unix Shell: Experiments

- Dataset: LUBM university benchmark
- 14 different queries
- competitors: Datalog-based (DLV, Souffle, RDFox),
  Triple store (Jena, Stardog, Virtuoso),
  Database management system (MonetDB, Postgres)

Number of finished queries

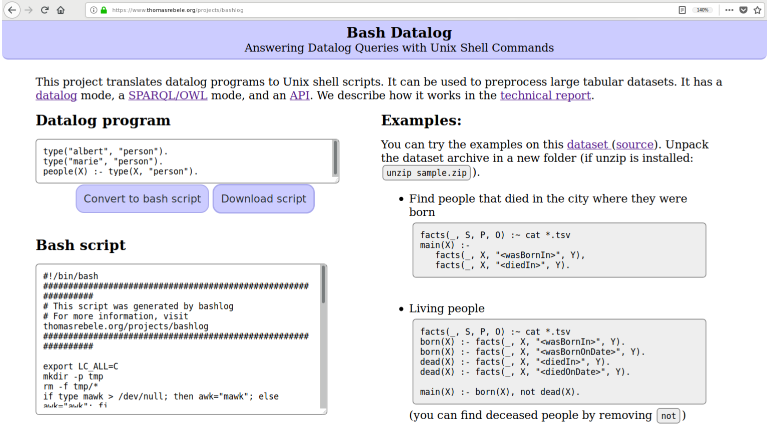
LUBM	Bash	DLV	Souffle	RDFox	Jena	Stardog	Virtuoso	MonetDB*	Postgres*
10	**14**	**14**	13	**14**	5	**14**	6	10	10
500	**14**		11	**14**		**14**	6	10	
1000	**14**		4	**14**		**14**		10	

Runtime in seconds

LUBM	Bash	DLV	Souffle	RDFox	Jena	Stardog	Virtuoso	MonetDB*	Postgres*
10	**1.6**	9.3	(21.9)	2.2	(78.7)	13.6	(11.8)	(5.2)	(20.6)
500	**83**		(310)	132		676	(1581)	(600)	
1000	**258**		(346)	278		2009		(1187)	

* = we folded the TBox into the query

Answering Queries with Unix Shell: Experiments



Bash Datalog

Answering Datalog Queries with Unix Shell Commands

This project translates datalog programs to Unix shell scripts. It can be used to preprocess large tabular datasets. It has a datalog mode, a SPARQL/OWL mode, and an API. We describe how it works in the technical report.

Datalog program

```
type("albert", "person").
type("marie", "person").
people(X) :- type(X, "person").
```

Bash script

```
#!/bin/bash
#####################################################
##########
# This script was generated by bashlog
# For more information, visit
thomasrebele.org/projects/bashlog
#####################################################
##########

export LC_ALL=C
mkdir -p tmp
rm -f tmp/*
if type mawk > /dev/null; then awk="mawk"; else
awk="awk"; fi
```

Examples:

You can try the examples on this dataset (source). Unpack the dataset archive in a new folder (if unzip is installed: `unzip sample.zip`).

- Find people that died in the city where they were born

```
facts(_, S, P, O) :~ cat *.tsv
main(X) :-
   facts(_, X, "<wasBornIn>", Y),
   facts(_, X, "<diedIn>", Y).
```

- Living people

```
facts(_, S, P, O) :~ cat *.tsv
born(X) :- facts(_, X, "<wasBornIn>", Y).
born(X) :- facts(_, X, "<wasBornOnDate>", Y).
dead(X) :- facts(_, X, "<diedIn>", Y).
dead(X) :- facts(_, X, "<diedOnDate>", Y).

main(X) :- born(X), not dead(X).
```

(you can find deceased people by removing `not`)

Figure: Screenshot of the web interface

Answering Queries with Unix Shell: Summary

Summary

- Preprocess large datasets without installing software
- Supports OWL RL subset and Datalog as query language
- Try it online at `https://www.thomasrebele.org/projects/bashlog`
- Source code available at `https://github.com/thomasrebele/bashlog`

Future work

- numerical comparisons
- aggregations (e.g., max, count)

Thomas Rebele

Thomas P. Tanon

Fabian Suchanek

publication: ISWC 2018 (full paper)

Conclusion

This thesis showed how to extend YAGO along several axes:

- Improve completeness w.r.t. people
- Automatically repairing of its regular expressions
- Preprocessing queries using only a Bash shell

- Interdisciplinary project
- Source code of all contributions is available online
- Publications in ISWC 2016, ISWC 2017, ISWC 2018, PAKDD 2018 (other publication in TPDL 2016 (demo))