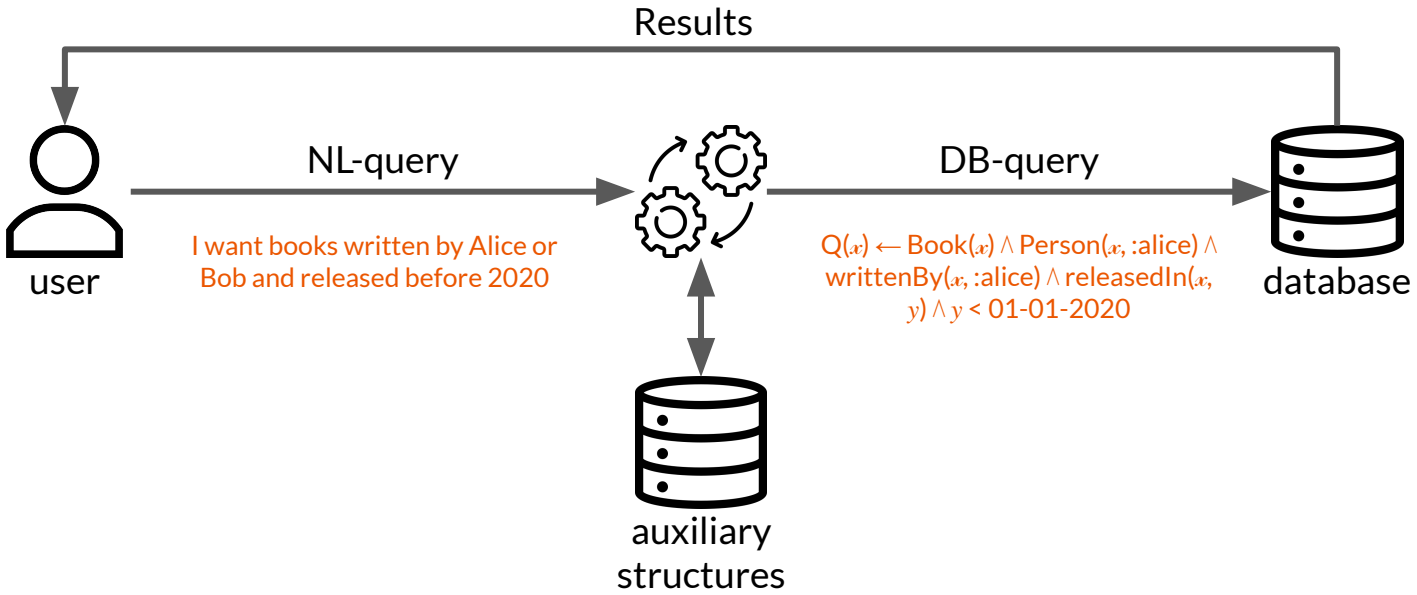
Natural Language Querying System through Entity Enrichment

Joshua Amavi[1], Mirian Halfeld-Ferrari[2] and **Nicolas Hiot**[1,2] (speaker)

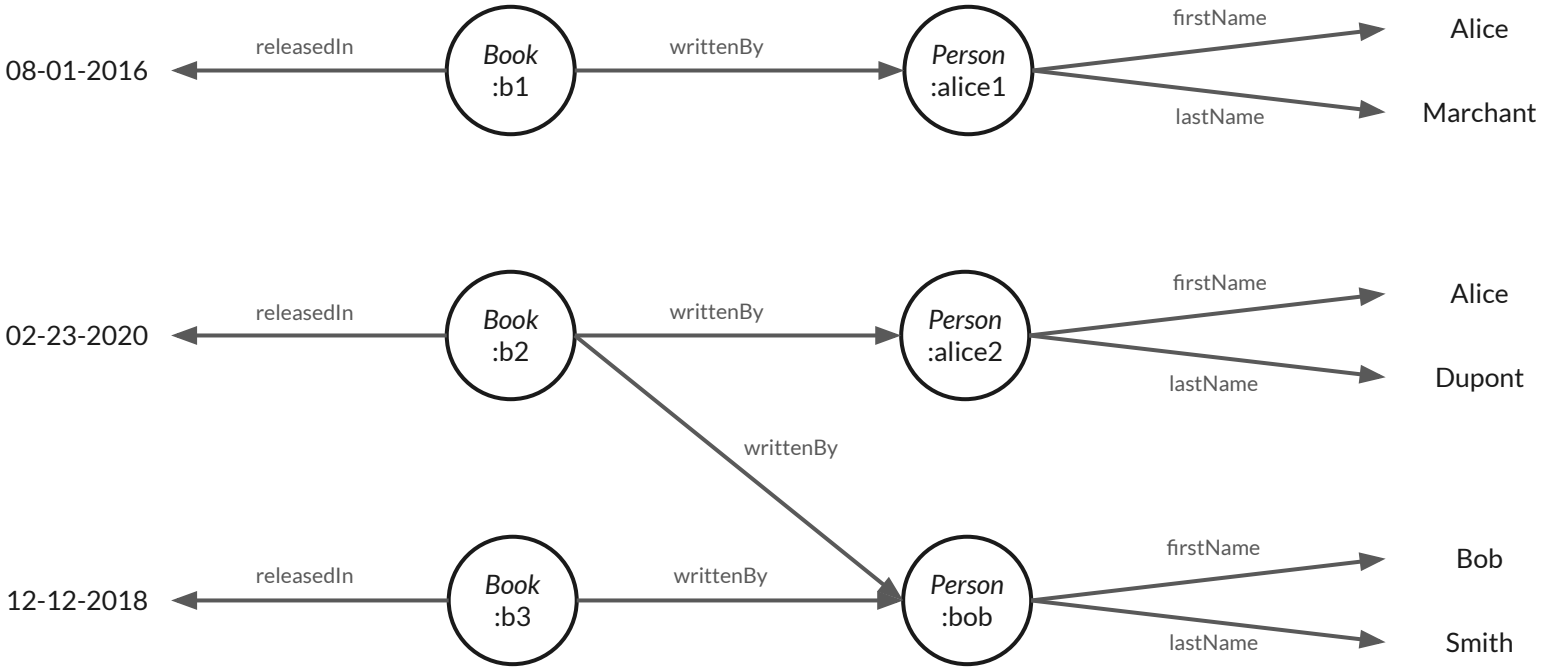
(1) Ennov
(2) University of Orléans

Natural Language Interface on database



Summary

- Entity Extraction and Enrichment
- Building DB-query from Enriched Entities
- Experimental Results
- Conclusion



Database example

I want books written by Alice or Bob and released before 2020

I want books written by Alice or Bob and released before 2020

Entity Extraction: which method?

Why we use Lexicons/Grammars approach instead of Deep Learning:

- Work well for limited knowledge
- Can be generated from the database
- Easy to update
- Use grammar for values of huge/infinite domains (dates, numbers, ...)

How to deal with ambiguity when instances share lexemes in lexicons?

Entity Extraction: how to deal with ambiguity ?

- Interaction with the user
  - Not transparent for the user
- Reduce ambiguity
  - Costly
  - Error prone
- Include the ambiguity in the query
  - Use extended entity representation to keep the value/type ambiguity

I want books *written by* Alice or Bob and *released* before 2020

Entity	Entity Value	lexT
E1	Book	Class
E2	:alice1	Person
	:alice2	Person
E3	:bob	Person

Entity	Entity Value	lexT
E4	lowerThan	Operator
E5	01-01-2020	Date
E6	*Author*	*Context*
E7	*ReleaseDate*	*Context*

I want books *written by* Alice or Bob and *released* before 2020

Entity	Entity Value	lexT
E1	Book	Class
E2	:alice1	Person
	:alice2	Person
E3	:bob	Person

Entity	Entity Value	lexT
E4	lowerThan	Operator
E5	01-01-2020	Date
E6	*Author*	*Context*
E7	*ReleaseDate*	*Context*

I want books *written by* Alice or Bob and *released* before 2020

Entity	Entity Value	lexT
E1	Book	Class
E2	:alice1	Person
	:alice2	Person
E3	:bob	Person

Entity	Entity Value	lexT
E4	lowerThan	Operator
E5	01-01-2020	Date
E6	*Author*	*Context*
E7	*ReleaseDate*	*Context*

I want books *written by* Alice or Bob and *released* before 2020

Entity	Entity Value	lexT
E1	Book	Class
E2	:alice1	Person
	:alice2	Person
E3	:bob	Person

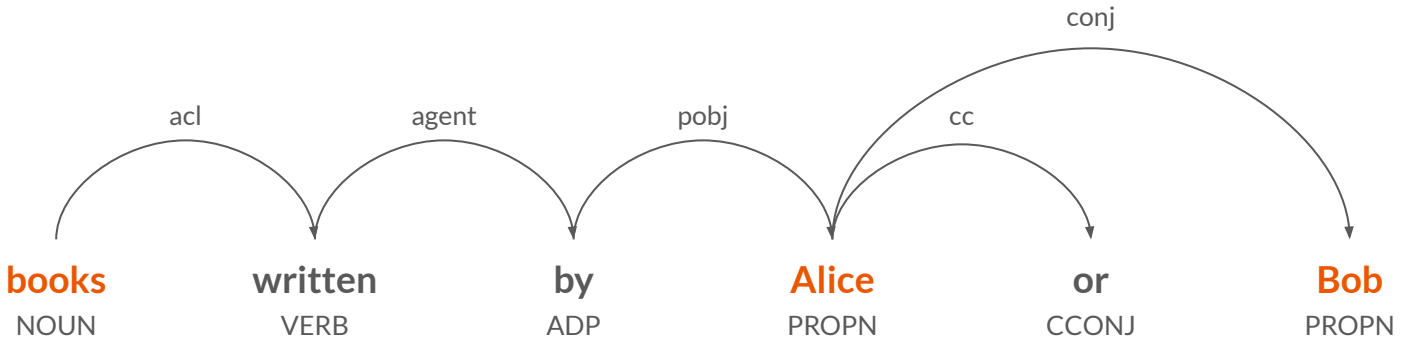
Entity	Entity Value	lexT
E4	lowerThan	Operator
E5	01-01-2020	Date
E6	*Author*	*Context*
E7	*ReleaseDate*	*Context*

I want books *written by* Alice or Bob and *released* before 2020

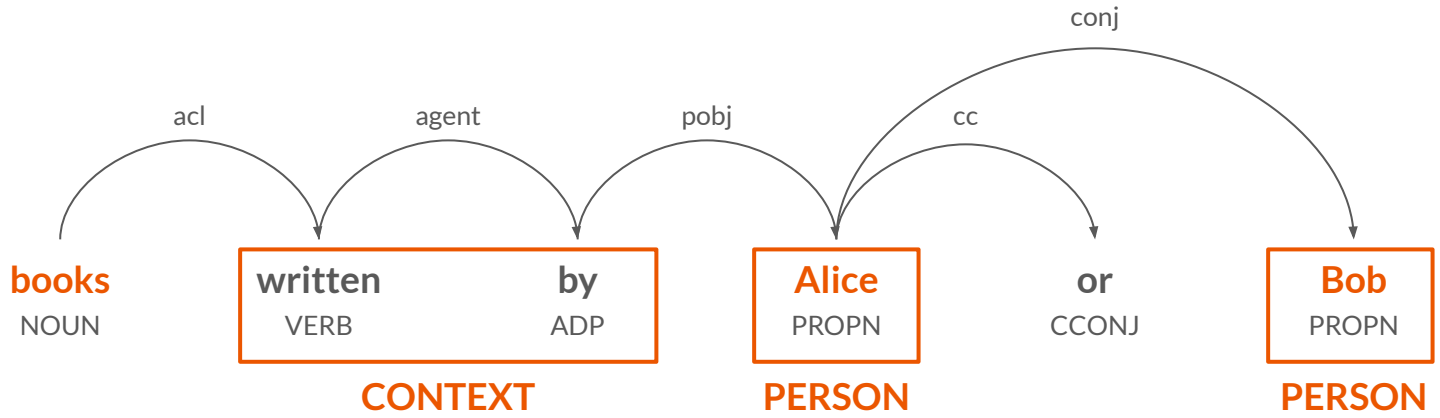
Entity	Entity Value	lexT
E1	Book	Class
E2	:alice1	Person
	:alice2	Person
E3	:bob	Person

Entity	Entity Value	lexT
E4	lowerThan	Operator
E5	01-01-2020	Date
E6	*Author*	*Context*
E7	*ReleaseDate*	*Context*

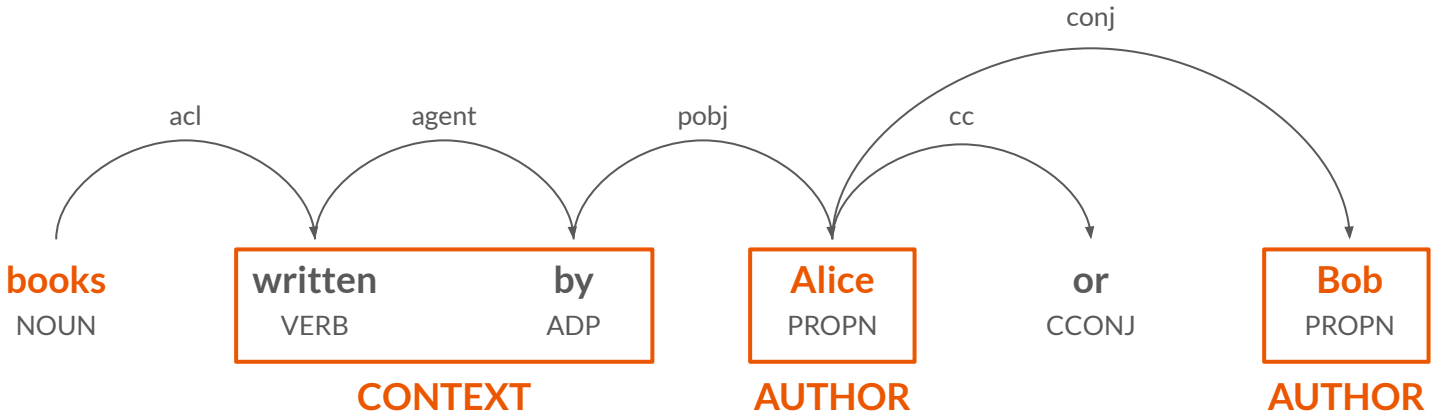
Entity Enrichment: include contexts/operators



Entity Enrichment: include contexts/operators



Entity Enrichment: include contexts/operators



I want books written by Alice or Bob and released before 2020

Entity	Entity Value	dbT	op
Ee1	Book	Class	=
Ee3	:bob	Person	=
	:bob	Author	=
Ee4	01-01-2020	ReleaseDate	<

Entity	Entity Value	dbT	op
Ee2	:alice1	Person	=
	:alice2	Person	=
	:alice1	Author	=
	:alice2	Author	=

How to deal with coordinating conjunctions? (“Alice *or* Bob”)

Entity Enrichment: conjunction management

- ***AND*** simply give different information so no need for treatment
- ***OR*** add ambiguity and need to be computed
    - Ambiguity on the value

I want books written by Alice or Bob and released before 2020

Entity	Entity Value	dbT	op
Ee1	Book	Class	=
Ee2	:alice1	Person	=
	:alice2	Person	=
	:bob	Person	=

Entity	Entity Value	dbT	op
Ee2	:alice1	Author	=
	:alice2	Author	=
	:bob	Author	=
Ee4	01-01-2020	ReleaseDate	<

Build the DB-query

Entity	Entity Value	dbT
Ee2	:alice1	Person
		Author
	:alice2	Person
		Author

- Direct translation using a mapping between dbT and predicates
- We have an ambiguity (multiple query) when an entity contains multiple values

I want books written by Alice or Bob and released before 2020

$Q1(x) \leftarrow Book(x)$
$\quad \wedge Person(x, :alice1)$
$\quad \wedge writtenBy(x, :alice1)$
$\quad \wedge releasedIn(x, y)$
$\quad \wedge y < 01\text{-}01\text{-}2020$

$Q2(x) \leftarrow Book(x)$
$\quad \wedge Person(x, :alice2)$
$\quad \wedge writtenBy(x, :alice2)$
$\quad \wedge releasedIn(x, y)$
$\quad \wedge y < 01\text{-}01\text{-}2020$

$Q3(x) \leftarrow Book(x)$
$\quad \wedge Person(x, :bob)$
$\quad \wedge writtenBy(x, :bob)$
$\quad \wedge releasedIn(x, y)$
$\quad \wedge y < 01\text{-}01\text{-}2020$

Experimentals Results

Database:
- 66 classes
- 29327 instances

Dataset:
- 113 NL-queries
- 10 classes used

dbT	precision	recall	f1-score	support
Class	1.00	0.62	0.77	82
ApplicationDate	1.00	0.62	0.76	13
ArchiveDate	0.50	0.67	0.57	3
CreationDate	0.60	0.60	0.60	5
ExpirationDate	1.00	0.50	0.67	2
Customers	1.00	0.60	0.75	5
Department	1.00	0.11	0.20	9
Sector	1.00	0.95	0.98	21
Author	0.77	0.63	0.70	38
Signatory	0.90	0.86	0.88	21
Status	1.00	0.50	0.67	6
Unit	1.00	0.27	0.43	11
…	…	…	…	…
Weighted avg.	**0.86**	**0.59**	**0.67**	**295**

Conclusion

- Information Extraction based approach: database/query language independent
- Domain independent approach
- Good precision (>80%) thanks to lexicons
- Weaker recall (<60%) because of evaluating method (ambiguity)
- Current work: evaluation method for ambiguity and deep learning entities enrichment