PADERBORN UNIVERSITY

Explainable ML on KGs

Recent Advances

Axel Ngonga

DICE

Lecture at the Data Science Summer School 2022

October 7, 2022

Section 1

Motivation

Motivation

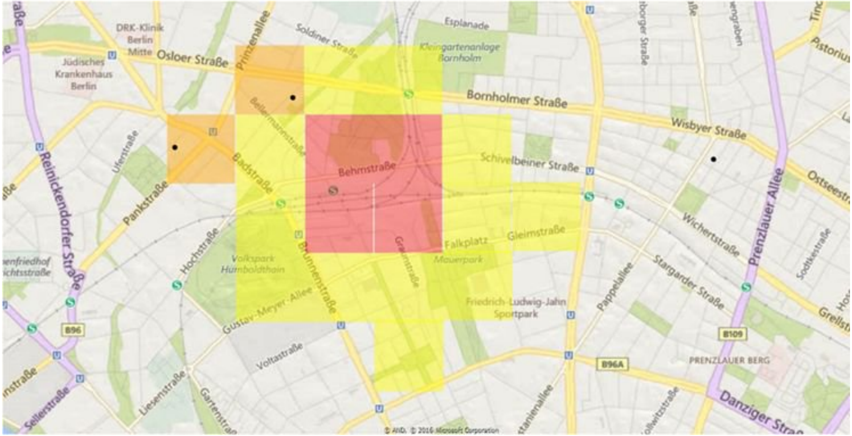
Automated Decision Making – Bail

Motivation

Automated Decision Making – Loans

Motivation

Automated Decision Making – Policing



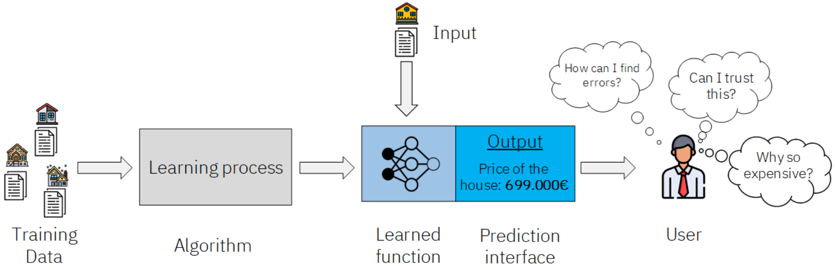
Motivation

Cooking Robot

- New cooking environment
- Unknown cookware
- Which utensil should be used to chop apples and why?

Motivation

Explainable AI



- Explain global output of machine learning model
- Explain important features
- Explain via counterfactuals

Motivation

Cooking Robot

- New cooking environment
- Unknown cookware
- Which utensil should be used to chop apples and why?
- Idea: Learn based on previous experiences or external knowledge sources

Motivation

Cooking Robot

- New cooking environment
- Unknown cookware
- Which utensil should be used to chop apples and why?
- Idea: Learn based on previous experiences or external knowledge sources

- Example: $\texttt{ChoppingDevice} \sqsubseteq \exists hasBladeLength.\{15, 16, 17\}$
- Pro: explainable, exploits background knowledge
- Contra: slow :-(

Motivation

Explainable AI

- Claim: Learning on knowledge graphs can be ante-hoc globally explainable and supports counterfactuals

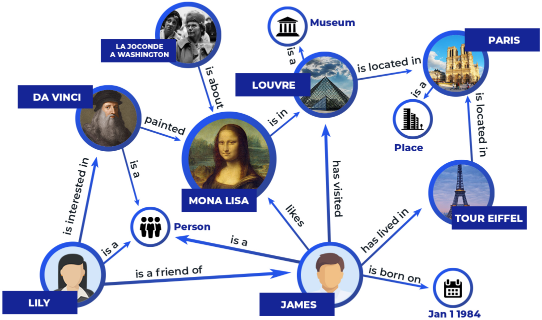
Section 2

Knowledge Graphs

Knowledge Graphs

Definition

- Focus on RDF knowledge graphs
- Formally, every RDF graph $G = (V, E)$, where
  - $V = \mathcal{R}$ is the set of all resources
  - $E \subseteq \mathcal{R} \times \mathcal{P} \times \mathcal{R}$ where $\mathcal{P}$ is the set of all predicates
  - RDF graphs are hence hypergraphs



https://towardsdatascience.com/explainable-artificial-intelligence-14944563cc79

Knowledge Graphs

$\mathcal{ALC}$ – Concepts

- $\mathcal{ALC}$ = Attributive Language with Complement
- Simplest closed DL (w.r.t. propositional logics)
- (Complex) $\mathcal{ALC}$ concepts are defined iteratively
  - Every concept name is a concept
  - $\top$ (“top”) and $\bot$ (“bottom”) are concepts

$\mathcal{ALC}$ – Concepts

- $\mathcal{ALC}$ = Attributive Language with Complement
- Simplest closed DL (w.r.t. propositional logics)
- (Complex) $\mathcal{ALC}$ concepts are defined iteratively
  - Every concept name is a concept
  - $\top$ ("top") and $\bot$ ("bottom") are concepts
- Let C and D be concepts and r be a role. The following constructs are $\mathcal{ALC}$ concepts:
  - $\neg C$ (Negation or complement)

$\mathcal{ALC}$ – Concepts

- $\mathcal{ALC}$ = Attributive Language with Complement
- Simplest closed DL (w.r.t. propositional logics)
- (Complex) $\mathcal{ALC}$ concepts are defined iteratively
  - Every concept name is a concept
  - $\top$ ("top") and $\bot$ ("bottom") are concepts
- Let C and D be concepts and r be a role. The following constructs are $\mathcal{ALC}$ concepts:
  - $\neg C$ (Negation or complement)
  - $C \sqcap D$ (Conjunction)

Knowledge Graphs

$\mathcal{ALC}$ – Concepts

- $\mathcal{ALC}$ = Attributive Language with Complement
- Simplest closed DL (w.r.t. propositional logics)
- (Complex) $\mathcal{ALC}$ concepts are defined iteratively
  - Every concept name is a concept
  - $\top$ ("top") and $\bot$ ("bottom") are concepts
- Let C and D be concepts and r be a role. The following constructs are $\mathcal{ALC}$ concepts:
  - $\neg C$ (Negation or complement)
  - $C \sqcap D$ (Conjunction)
  - $C \sqcup D$ (Disjunction, union)

$\mathcal{ALC}$ – Concepts

- $\mathcal{ALC}$ = Attributive Language with Complement
- Simplest closed DL (w.r.t. propositional logics)
- (Complex) $\mathcal{ALC}$ concepts are defined iteratively
  - Every concept name is a concept
  - $\top$ ("top") and $\bot$ ("bottom") are concepts
- Let C and D be concepts and r be a role. The following constructs are $\mathcal{ALC}$ concepts:
  - $\neg C$ (Negation or complement)
  - $C \sqcap D$ (Conjunction)
  - $C \sqcup D$ (Disjunction, union)
  - $\exists r.C$ (existential restriction)

$\mathcal{ALC}$ – Concepts

- $\mathcal{ALC}$ = Attributive Language with Complement
- Simplest closed DL (w.r.t. propositional logics)
- (Complex) $\mathcal{ALC}$ concepts are defined iteratively
  - Every concept name is a concept
  - $\top$ ("top") and $\bot$ ("bottom") are concepts
- Let C and D be concepts and r be a role. The following constructs are $\mathcal{ALC}$ concepts:
  - $\neg C$ (Negation or complement)
  - $C \sqcap D$ (Conjunction)
  - $C \sqcup D$ (Disjunction, union)
  - $\exists r.C$ (existential restriction)
  - $\forall r.C$ (universal restriction)

Knowledge Graphs

$\mathcal{ALC}$ – Examples

$$Person \sqcap \exists hasChild.\top$$

$\mathcal{ALC}$ – Examples

$Person \sqcap \exists hasChild.\top$

- Persons with at least one child

$Animal \sqcap \forall eats.Plant$

Knowledge Graphs

$\mathcal{ALC}$ – Examples

$Person \sqcap \exists hasChild.\top$

- Persons with at least one child

$Animal \sqcap \forall eats.Plant$

- Animals that only each plants

$Professor \sqcup Student$

$\mathcal{ALC}$ – Examples

$Person \sqcap \exists hasChild.\top$

- Persons with at least one child

$Animal \sqcap \forall eats.Plant$

- Animals that only each plants

$Professor \sqcup Student$

- Professors or Students

$Person \sqcap \forall bornIn.\neg \mathrm{City}$

Knowledge Graphs

$\mathcal{ALC}$ – Examples

$Person \sqcap \exists hasChild.\top$

- Persons with at least one child

$Animal \sqcap \forall eats.Plant$

- Animals that only each plants

$Professor \sqcup Student$

- Professors or Students

$Person \sqcap \forall bornIn.\neg \mathrm{City}$

- Persons not born in a city

$\mathcal{ALC}$ – Class expressions and Axioms

- Every $\mathcal{ALC}$ concept is a class expression
- Often need subsumption to learn models
  - Let R be a retrieval function
  - $C \sqsubseteq D$ iff $R(C) \subseteq R(D)$
- Example: $Person \sqcap \forall bornIn.\neg City \sqsubseteq Person$

Motivation

Let's play!

Motivation

Let's play!

- What is 3+3?

Motivation

Let's play!

- What is 3+3?
- Square root of 4?

Motivation

Let's play!

- What is 3+3?
- Square root of 4?
- What's the capital of France?

Motivation

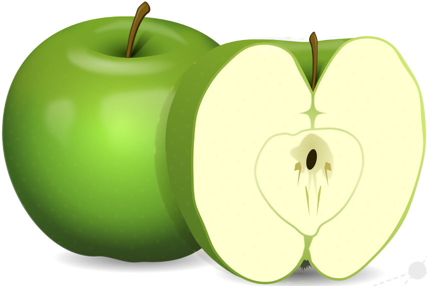
Let's play!

- What is 3+3?
- Square root of 4?
- What's the capital of France?
- Close your eyes.

Motivation

How does the brain form thoughts?

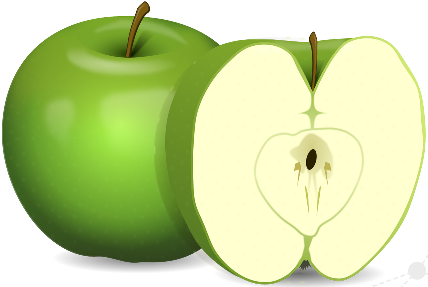
- System 1 [Kahneman, 2011]
  - Intuitive responses
  - Time-efficient
  - Unconscious



Motivation

How does the brain form thoughts?

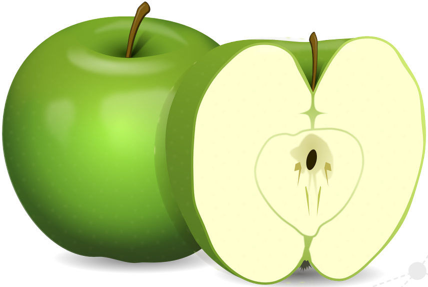
- System 1 [Kahneman, 2011]
  - Intuitive responses
  - Time-efficient
  - Unconscious
- System 2
  - Logical responses
  - Resource-intensive
  - Conscious



Motivation

How does the brain form thoughts?

- System 1 [Kahneman, 2011]
  - Intuitive responses
  - Time-efficient
  - Unconscious
- System 2
  - Logical responses
  - Resource-intensive
  - Conscious
- Both trainable and configurable



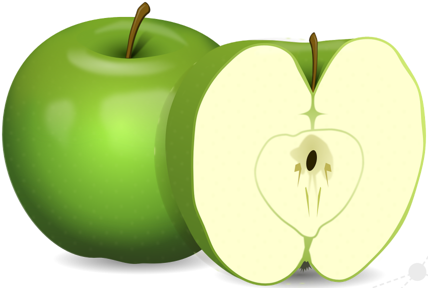
Motivation

How does the brain form thoughts?

In a nutshell

- Using multiple representations seems to be useful for humans
- Are multiple representations beneficial for structured machine learning?

- System 1 [Kahneman, 2011]
  - Intuitive responses
  - Time-efficient
  - Unconscious
- System 2
  - Logical responses
  - Resource-intensive
  - Conscious
- Both trainable and configurable



Section 3

Class Expression Learning

Class Expression Learning

Formal definition

- Supervised learning with background knowledge (adapted from [Lehmann and Hitzler, 2010])
- Given:
  - Formal logic $\mathcal{L}$, e.g. $\mathcal{ALC}$
  - Background knowledge in form of knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
  - Set of positive examples $E^+ \subseteq N_I$
  - Set of negative examples $E^- \subseteq N_I$

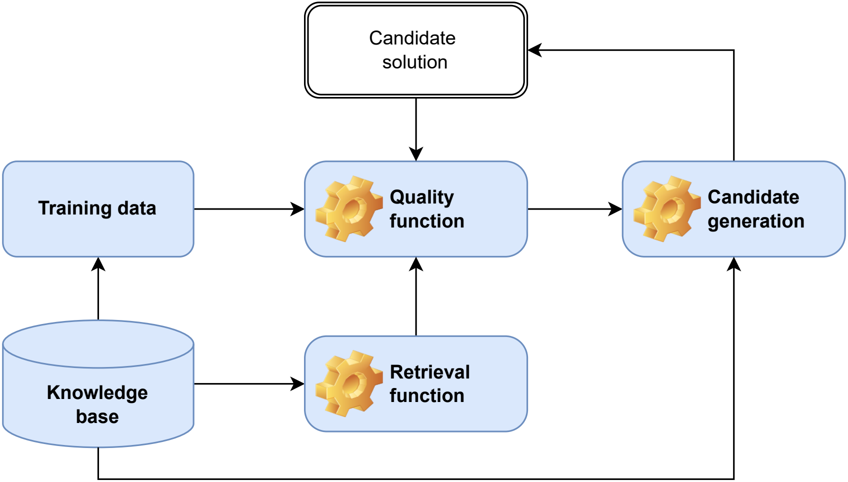
Class Expression Learning

Formal definition

- Supervised learning with background knowledge (adapted from [Lehmann and Hitzler, 2010])
- Given:
  - Formal logic $\mathcal{L}$, e.g. $\mathcal{ALC}$
  - Background knowledge in form of knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
  - Set of positive examples $E^+ \subseteq N_I$
  - Set of negative examples $E^- \subseteq N_I$
- Goal: Find at least one hypothesis $H \in \mathcal{H}$ with
  1. H is a class expression in $\mathcal{L}$, and (ideally)
  2. $\forall e^+ \in E^+ : \mathcal{K} \models H(e^+)$
  3. $\forall e^- \in E^- : \mathcal{K} \not\models H(e^-)$

Class Expression Learning

Common Approach



Class Expression Learning

Example: $\mathcal{L}$ = $\mathcal{ALC}$

- Let C and D be $\mathcal{ALC}$ concepts
- Let $r \in N_R$ be a role
- Then, the following are $\mathcal{ALC}$ concepts

Syntax	Semantics
$\top$	$\Delta^{\mathcal{I}}$
$\bot$	$\emptyset$
$C \in N_C$	$C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} : \exists y \in C^{\mathcal{I}} \text{ with } (x, y) \in r^{\mathcal{I}}\}$
$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} : (x, y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$

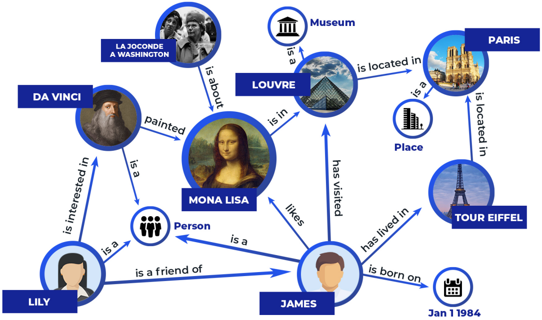
Class Expression Learning

Example: Refinement Operator

- Let $(S, \sqsubseteq)$ be a space with a quasi-ordering
- A top-down refinement operator $\rho : S \rightarrow 2^S$ is a mapping with $\rho(x) \sqsubseteq x$
- Let S be the set of all concepts in our language $\mathcal{L} = \mathcal{EL}$
- The following operator ρ is a top-down refinement operator
- $$\rho(C) = \begin{cases} C & \\ N_C \cup \{\exists r_j.\rho(C_i)\} & \text{if } C = \top \\ \rho(D) & \text{if } D \sqsubseteq C \\ C \sqcap D & \text{with } D \in N_C \\ C \sqcap \exists r.\rho(D) & \text{with } D \in N_C \end{cases}$$

Class Expression Learning

Example



[1]

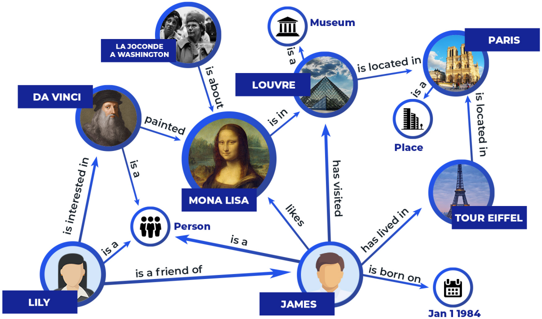
- $E^{+} = \{Louvre, TourEiffel\}$
- $E^{-} = \{Lily, James\}$

---

[1] Source: https://bit.ly/3sxCj6e

Class Expression Learning

Example



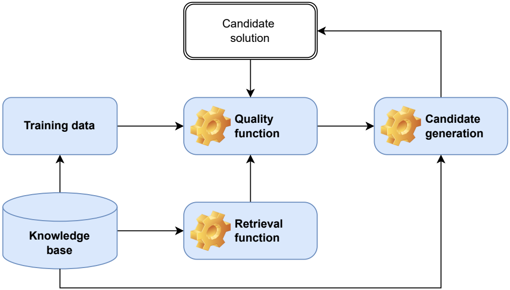
[1]

- $E^{+} = \{Louvre, TourEiffel\}$
- $E^{-} = \{Lily, James\}$
- $\mathcal{H} = \{\exists\, isLocatedIn.Place, \exists\, isLocatedIn.\{Paris\}\}$

[1] Source: https://bit.ly/3sxCj6e

Learning problem

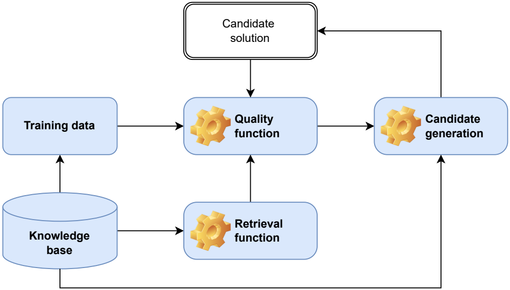
Challenges



- Retrieval is expensive

Learning problem

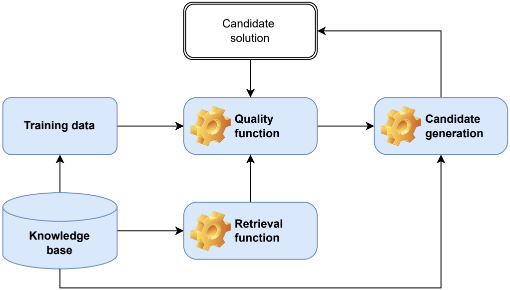
Challenges



- Retrieval is expensive $\Rightarrow$ Represent concepts in SPARQL
- Quality functions are often myopic

Learning problem

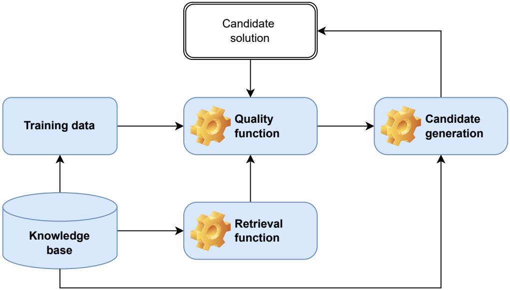
Challenges



- Retrieval is expensive $\Rightarrow$ Represent concepts in SPARQL
- Quality functions are often myopic $\Rightarrow$ Represent sets of individuals as embeddings
- Candidate generation is expensive

Learning problem

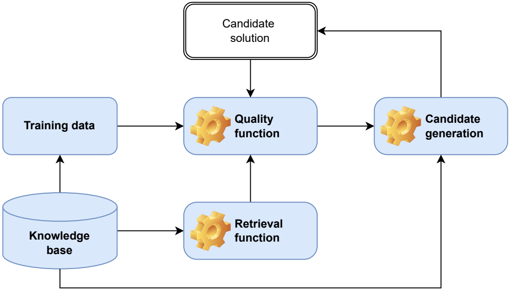
Challenges



- Retrieval is expensive ⇒ Represent concepts in SPARQL
- Quality functions are often myopic ⇒ Represent sets of individuals as embeddings
- Candidate generation is expensive ⇒ Represent individuals as graphs for priming

Learning problem

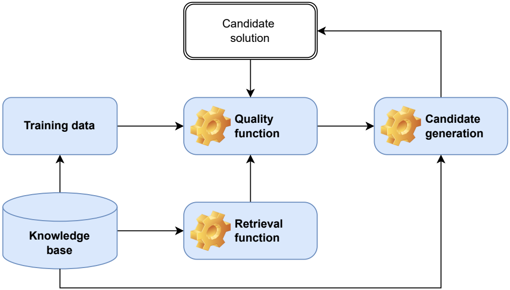
Challenges



- Retrieval is expensive $\Rightarrow$ Represent concepts in SPARQL
- Quality functions are often myopic $\Rightarrow$ Represent sets of individuals as embeddings
- Candidate generation is expensive $\Rightarrow$ Represent individuals as graphs for priming
- Search space is large

Learning problem

Challenges



- Retrieval is expensive $\Rightarrow$ Represent concepts in SPARQL
- Quality functions are often myopic $\Rightarrow$ Represent sets of individuals as embeddings
- Candidate generation is expensive $\Rightarrow$ Represent individuals as graphs for priming
- Search space is large $\Rightarrow$ Represent concepts as embeddings

Section 4

Representing Concepts as SPARQL

Representing Concepts as SPARQL

From $\mathcal{ALC}$ to SPARQL

- Assume closed world and fully materialized knowledge graph
- Retrieval in $\mathcal{ALC}$ can be realized by representing concepts as SPARQL queries [Bin et al., 2016]

Representing Concepts as SPARQL

From $\mathcal{ALC}$ to SPARQL

- Assume closed world and fully materialized knowledge graph
- Retrieval in $\mathcal{ALC}$ can be realized by representing concepts as SPARQL queries [Bin et al., 2016]

Class Expression	Graph Pattern $\mathfrak{p} = \tau(C_i, \texttt{?var})$
$A \in N_C$	`?var rdf:type A.`

Representing Concepts as SPARQL

From $\mathcal{ALC}$ to SPARQL

- Assume closed world and fully materialized knowledge graph
- Retrieval in $\mathcal{ALC}$ can be realized by representing concepts as SPARQL queries [Bin et al., 2016]

Class Expression	Graph Pattern $\mathfrak{p} = \tau(C_i, \texttt{?var})$
$A \in N_C$	`?var rdf:type A.`
$\neg C$	`{?var ?p ?o} UNION {?s ?p ?var}.` `FILTER NOT EXISTS` $\{\tau(C, \texttt{?var})\}$

Representing Concepts as SPARQL

From $\mathcal{ALC}$ to SPARQL

- Assume closed world and fully materialized knowledge graph
- Retrieval in $\mathcal{ALC}$ can be realized by representing concepts as SPARQL queries [Bin et al., 2016]

Class Expression	Graph Pattern $\mathfrak{p} = \tau(C_i, \texttt{?var})$
$A \in N_C$	`?var rdf:type A.`
$\neg C$	`{?var ?p ?o} UNION {?s ?p ?var}.` `FILTER NOT EXISTS` $\{\tau(C, \texttt{?var})\}$
$C_1 \sqcap \ldots \sqcap C_n$	$\{\tau(C_1, ?\text{var}) \ldots \tau(C_n, ?\text{var})\}$

Representing Concepts as SPARQL

From $\mathcal{ALC}$ to SPARQL

- Assume closed world and fully materialized knowledge graph
- Retrieval in $\mathcal{ALC}$ can be realized by representing concepts as SPARQL queries [Bin et al., 2016]

Class Expression	Graph Pattern $\mathfrak{p} = \tau(C_i, \texttt{?var})$
$A \in N_C$	`?var rdf:type A.`
$\neg C$	`{?var ?p ?o} UNION {?s ?p ?var}.` `FILTER NOT EXISTS` $\{\tau(C, \texttt{?var})\}$
$C_1 \sqcap \ldots \sqcap C_n$	$\{\tau(C_1, ?\text{var}) \ldots \tau(C_n, ?\text{var})\}$
$C_1 \sqcup \ldots \sqcup C_n$	$\{\tau(C_1, ?\text{var})\}$ `UNION` $\ldots$ `UNION` $\{\tau(C_n, ?\text{var})\}$

Representing Concepts as SPARQL

From $\mathcal{ALC}$ to SPARQL

- Assume closed world and fully materialized knowledge graph
- Retrieval in $\mathcal{ALC}$ can be realized by representing concepts as SPARQL queries [Bin et al., 2016]

Class Expression	Graph Pattern $\mathfrak{p} = \tau(C_i, \texttt{?var})$
$A \in N_C$	`?var rdf:type A.`
$\neg C$	`{?var ?p ?o} UNION {?s ?p ?var}.` `FILTER NOT EXISTS` $\{\tau(C, \texttt{?var})\}$
$C_1 \sqcap \ldots \sqcap C_n$	$\{\tau(C_1, ?\mathrm{var}) \ldots \tau(C_n, ?\mathrm{var})\}$
$C_1 \sqcup \ldots \sqcup C_n$	$\{\tau(C_1, ?\mathrm{var})\}$ `UNION` $\ldots$ `UNION` $\{\tau(C_n, ?\mathrm{var})\}$
$\exists\, r.C$	$\{$`?var r ?s.` $\tau(C, ?\mathrm{s})\}$

Representing Concepts as SPARQL

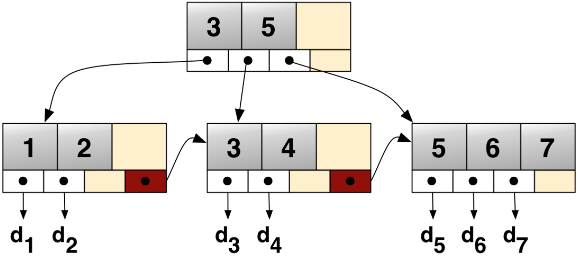
From $\mathcal{ALC}$ to SPARQL

- Assume closed world and fully materialized knowledge graph
- Retrieval in $\mathcal{ALC}$ can be realized by representing concepts as SPARQL queries [Bin et al., 2016]

Class Expression	Graph Pattern $\mathfrak{p} = \tau(C_i, \texttt{?var})$
$A \in N_C$	`?var rdf:type A.`
$\neg C$	`{?var ?p ?o} UNION {?s ?p ?var}.` `FILTER NOT EXISTS` $\{\tau(C, \texttt{?var})\}$
$C_1 \sqcap \ldots \sqcap C_n$	$\{\tau(C_1, ?var) \ldots \tau(C_n, ?var)\}$
$C_1 \sqcup \ldots \sqcup C_n$	$\{\tau(C_1, ?var)\}$ `UNION` $\ldots$ `UNION` $\{\tau(C_n, ?var)\}$
$\exists\, r.C$	$\{$`?var r ?s.` $\tau(C, ?s)\}$

$\forall\, r.C$	`{ ?var r ?s0.`
	 `{ SELECT ?var (count(?s1) AS ?cnt1)`
	 `WHERE { ?var r ?s1.` $\tau(C,$ `?s1)}`
	 `GROUP BY ?var }`
	 `{ SELECT ?var (count(?s2) AS ?cnt2)`
	 `WHERE { ?var r ?s2 .}`
	 `GROUP BY ?var }`
	 `FILTER ( ?cnt1 = ?cnt2 ) }`

- Important difference are indexing data structures
- Typical indexes include
  - Resource index, e.g., a hash table
  - Triple index, e.g., a B^+ tree

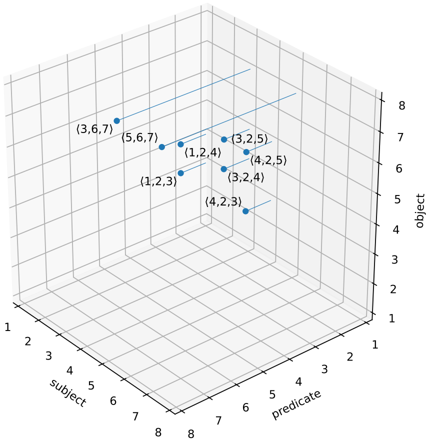


Representing Concepts as SPARQL

Tentris: Idea

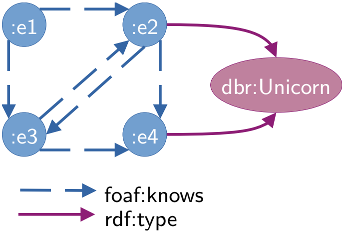
Idea [Bigerl et al., 2020]

- Exploit tensor representation to accelerate querying
- Devise data structure to accommodate rapid querying



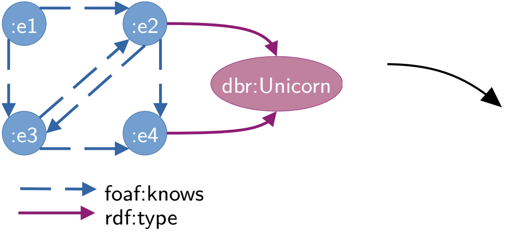
Representing Concepts as SPARQL

From RDF to Tensors



Representing Concepts as SPARQL

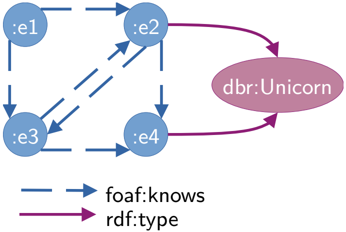
From RDF to Tensors



term	*id*(term)
:e1	1
foaf:knows	2
:e2	3
:e3	4
:e4	5
rdf:type	6
dbr:Unicorn	7
unbound	8

Representing Concepts as SPARQL

From RDF to Tensors

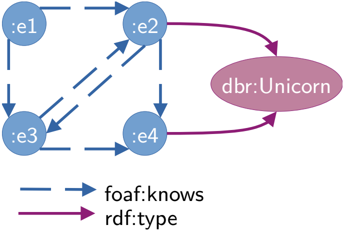


term	*id*(term)
:e1	1
foaf:knows	2
:e2	3
:e3	4
:e4	5
rdf:type	6
dbr:Unicorn	7
unbound	8

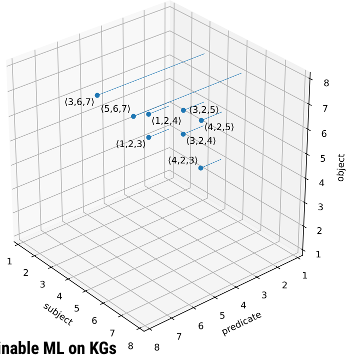
id(s)	*id*(p)	*id*(o)
1	2	3
1	2	4
3	2	4
3	2	5
4	2	3
4	2	5
3	6	7
5	6	7

Representing Concepts as SPARQL

From RDF to Tensors



term	*id*(term)
:e1	1
foaf:knows	2
:e2	3
:e3	4
:e4	5
rdf:type	6
dbr:Unicorn	7
unbound	8



Representing Concepts as SPARQL

Tentris: Data Model

- Consider order-n tensors $T : \mathbf{K} = \mathbf{K}_1 \times \cdots \times \mathbf{K}_n \to V$

Representing Concepts as SPARQL

TENTRIS: Data Model

- Consider order-n tensors $T : \mathbf{K} = \mathbf{K}_1 \times \cdots \times \mathbf{K}_n \to V$
  - $\mathbf{K}_1 = \cdots = \mathbf{K}_n \subset \mathbb{N}$

Representing Concepts as SPARQL

Tentris: Data Model

- Consider order-n tensors $T : \mathbf{K} = \mathbf{K}_1 \times \cdots \times \mathbf{K}_n \to V$
  - $\mathbf{K}_1 = \cdots = \mathbf{K}_n \subset \mathbb{N}$
  - $\mathbb{B}$ or $\mathbb{N}$ as co-domain

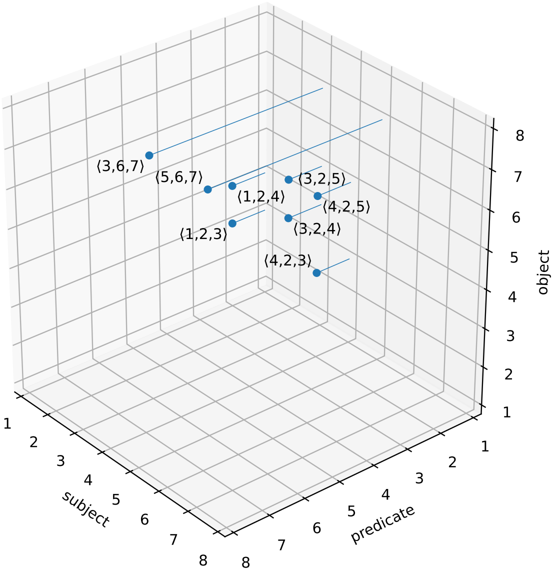
Representing Concepts as SPARQL

TENTRIS: Data Model

- Consider order-n tensors $T : \mathbf{K} = \mathbf{K}_1 \times \cdots \times \mathbf{K}_n \to V$
  - $\mathbf{K}_1 = \cdots = \mathbf{K}_n \subset \mathbb{N}$
  - $\mathbb{B}$ or $\mathbb{N}$ as co-domain
- $\mathbf{k} \in \mathbf{K}$ is a key with key parts $\langle \mathbf{k}_1, \ldots, \mathbf{k}_n \rangle$
- Values v in a tensor are accessed in array style, e.g., $T[\mathbf{k}] = v$

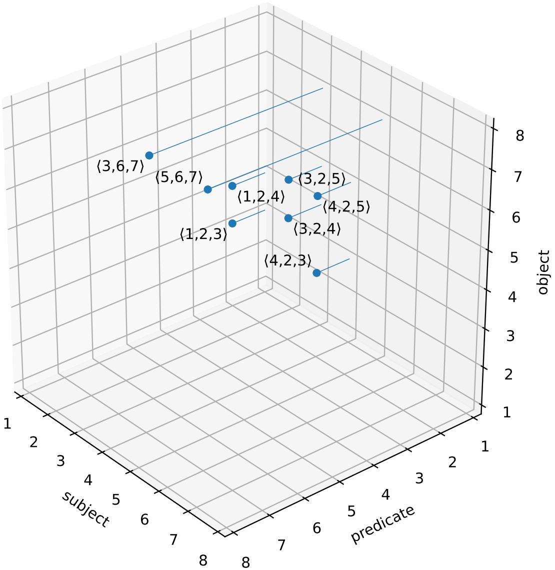
Representing Concepts as SPARQL

Tentris: Data Model



Representing Concepts as SPARQL

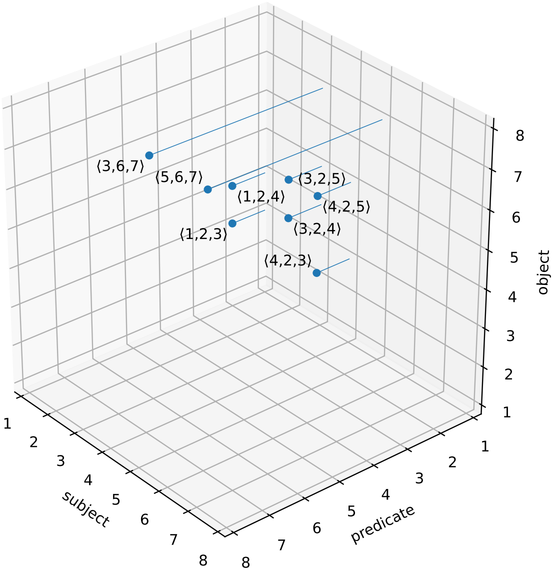
Tentris: Data Model



- $\mathbf{K} = \mathbb{N}^3$
- $V = \mathbb{B}$

Representing Concepts as SPARQL

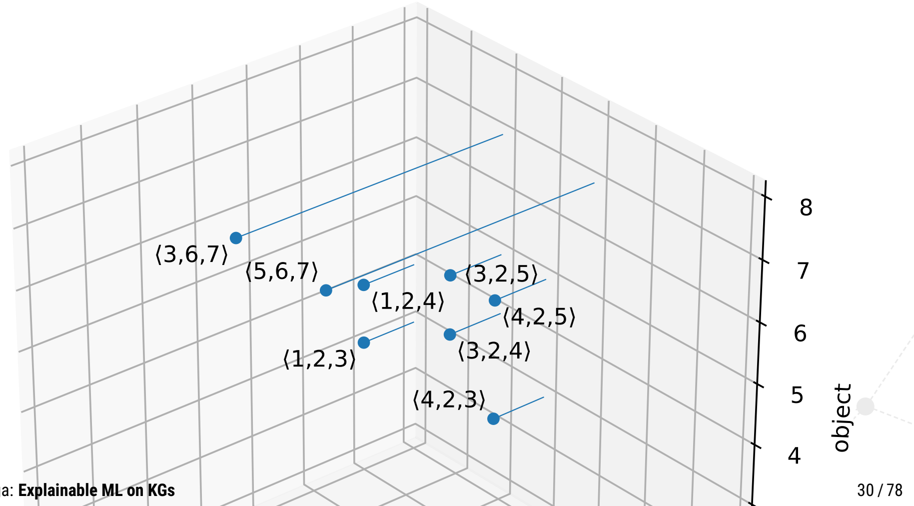
TENTRIS: Data Model



- $\mathbf{K} = \mathbb{N}^3$
- $V = \mathbb{B}$
- $T[\langle 3,6,7\rangle] = 1$
- $T[\langle 3,6,3\rangle] = 0$

TENTRIS: Data Model

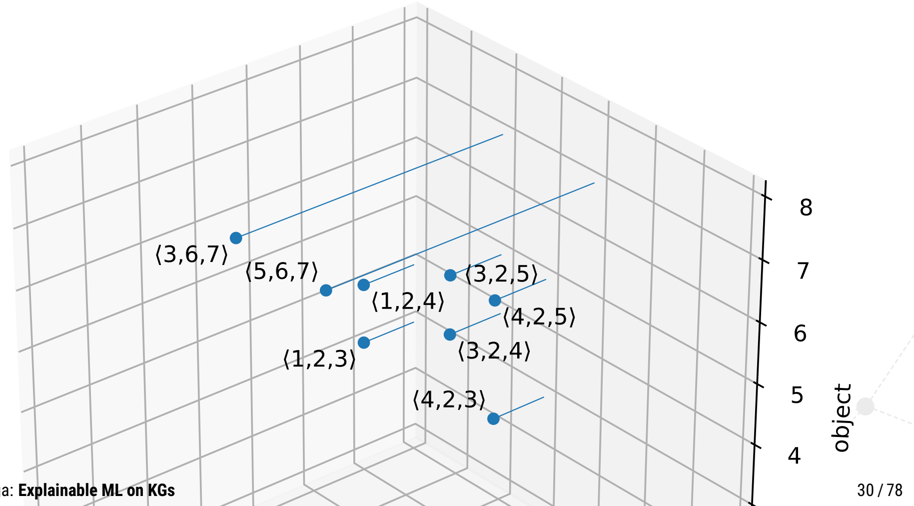
- Slicing selects portion of T, e.g., $T^{(1)} := T[1, 2, :]$ is order-1 tensor



Representing Concepts as SPARQL

Tentris: Data Model

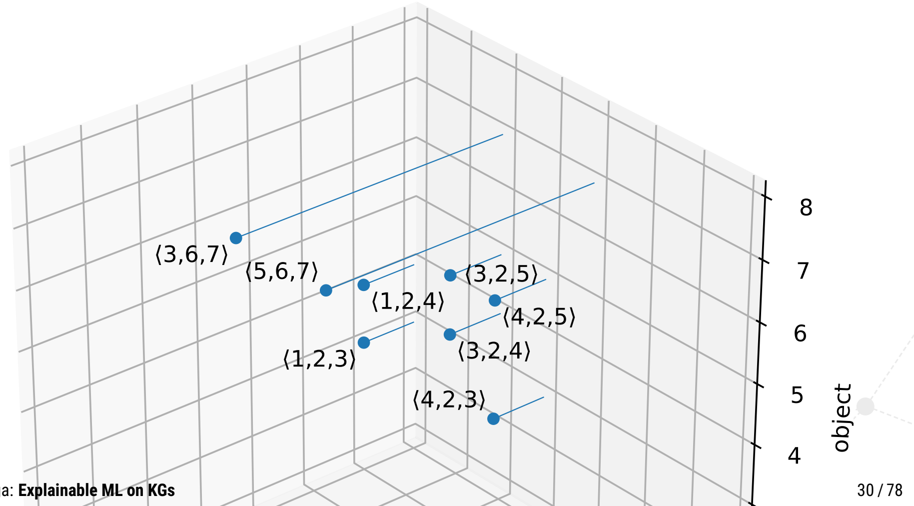
- Slicing selects portion of T, e.g., $T^{(1)} := T[1, 2, :]$ is order-1 tensor
- For our example, $T[1, 2, :] = [0, 0, 1, 1, 0, 0, 0, 0]$



Representing Concepts as SPARQL

TENTRIS: Data Model

- Slicing selects portion of T, e.g., $T^{(1)} := T[1,2,:]$ is order-1 tensor
- For our example, $T[1,2,:] = [0,0,1,1,0,0,0,0]$
- Slices can be joined via Einstein summation [Barr, 1989]



PADERBORN UNIVERSITY

Representing Concepts as SPARQL

DICE

Tentris–Einstein Summation

```
SELECT ?f WHERE {
  :e1 foaf:knows ?f .
  ?f  foaf:knows ?u .
  ?u  rdf:type dbr:Unicorn
}
```

Representing Concepts as SPARQL

TENTRIS–Einstein Summation

```
SELECT ?f WHERE {
    :e1 foaf:knows ?f .
    ?f  foaf:knows ?u .
    ?u  rdf:type dbr:Unicorn
}
```

$T[1, 2, :]$ $T[:, 2, :]$ $T[:, 6, 7]$

Representing Concepts as SPARQL

Tentris–Einstein Summation

```
SELECT ?f WHERE {
    :e1 foaf:knows ?f .
    ?f  foaf:knows ?u .
    ?u  rdf:type dbr:Unicorn
}
```

$T[1, 2, :]$ $\qquad$ $T[:, 2, :]$ $\qquad$ $T[:, 6, 7]$

$$R_f \leftarrow T[1, 2, :]_f \times T[:, 2, :]_{f,u} \times T[:, 6, 7]_u$$

Representing Concepts as SPARQL

TENTRIS: Querying

- Triple pattern is mapped to

$$\mathbf{k}_i^{(Q)} := \begin{cases} :, & \text{if } Q_i \in U, \\ id(Q_i), & \text{otherwise.} \end{cases}$$

Representing Concepts as SPARQL

TENTRIS: Querying

- Triple pattern is mapped to

$$\mathbf{k}_i^{(Q)} := \begin{cases} :, & \text{if } Q_i \in U, \\ id(Q_i), & \text{otherwise.} \end{cases}$$

- BGP $B = \{B^{(1)}, \ldots, B^{(r)}\}$ is given by

$$T'_{\langle l \in U \rangle} \leftarrow \bigtimes_i T[\mathbf{k}^{B^{(i)}}]_{\langle l \in B^{(i)} | l \in U \rangle}$$

Representing Concepts as SPARQL

TENTRIS: Querying

- Triple pattern is mapped to

$$\mathbf{k}_i^{(Q)} := \begin{cases} :, & \text{if } Q_i \in U, \\ id(Q_i), & \text{otherwise.} \end{cases}$$

- BGP $B = \{B^{(1)}, \ldots, B^{(r)}\}$ is given by

$$T'_{\langle l \in U \rangle} \leftarrow \bigtimes_i T[\mathbf{k}^{B^{(i)}}]_{\langle l \in B^{(i)} | l \in U \rangle}$$

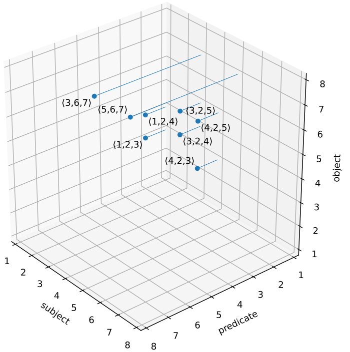
- The projection $\Pi_{U'}(B(g))$ with $U' \subseteq U$ is given by

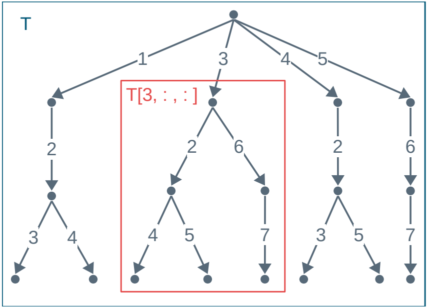
$$T''_{\langle l \in U' \rangle} \leftarrow \bigtimes_i T[\mathbf{k}^{B^{(i)}}]_{\langle l \in B^{(i)} | l \in U \rangle}$$

Representing Concepts as SPARQL

Tentris: Hypertrie

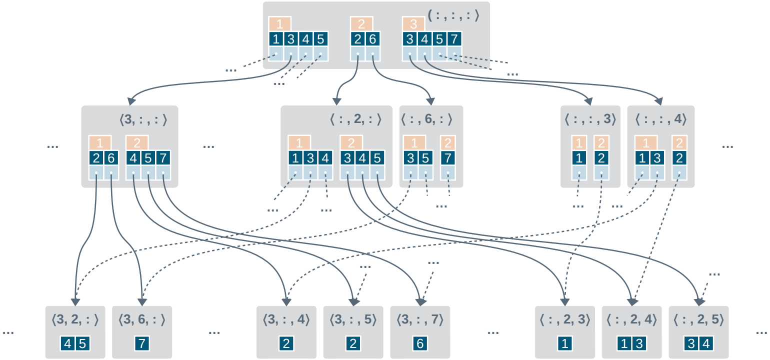
- Query for any tensor slice efficiently
- Allow for efficient querying





Representing Concepts as SPARQL

TENTRIS: Hypertrie



- Query for any tensor slice efficiently
- Storage bound is reduced from $\mathcal{O}(d! \cdot d \cdot z(h))$ for all collation orders to $\mathcal{O}(2^{d-1} \cdot d \cdot z(h))$

Representing Concepts as SPARQL

TENTRIS: Evaluation – Setup

- Evaluation via HTTP and CLI
- Timeout = 180 s
- Benchmark runtime = 60 min
- Comparison with
  - Virtuoso 7.2.5,
  - Fuseki 3.5.0,
  - Blazegraph v2.0, and
  - GraphDB Lite v.8.3.1

Representing Concepts as SPARQL

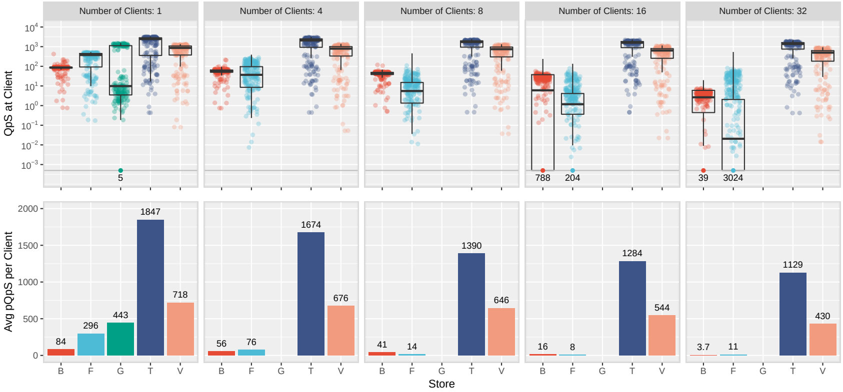
TENTRIS: Evaluation – Setup

- Evaluation via HTTP and CLI
- Timeout = 180 s
- Benchmark runtime = 60 min
- Comparison with
  - Virtuoso 7.2.5,
  - Fuseki 3.5.0,
  - Blazegraph v2.0, and
  - GraphDB Lite v.8.3.1

Dataset	#Q	#TP	#R	#D	avg JVD
SWDF	203	1.74 (1 - 9)	5.5 k (1 - 304 k)	124 (61%)	0.75 (0 - 4)
DBpedia	557	1.84 (1 - 14)	13.2 k (0 - 843 k)	222 (40%)	1.19 (0 - 4)
WatDiv	45	6.51 (2 - 10)	3.7 k (0 - 34 k)	2 (4%)	2.61 (2 - 9)

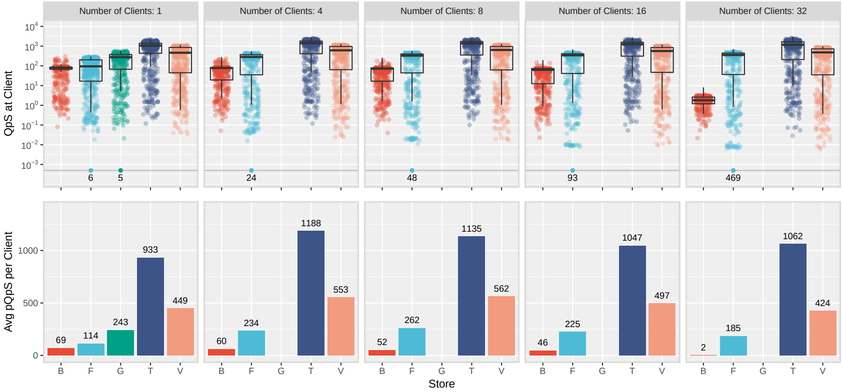
Representing Concepts as SPARQL

Tentris: Evaluation – SWDF



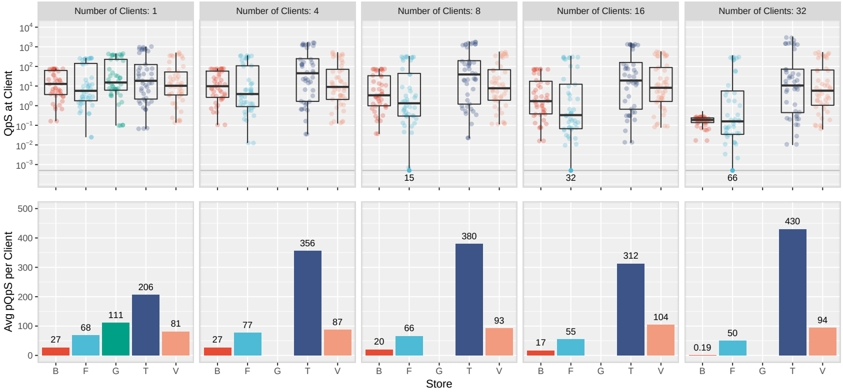
Representing Concepts as SPARQL

TENTRIS: Evaluation – DBpedia



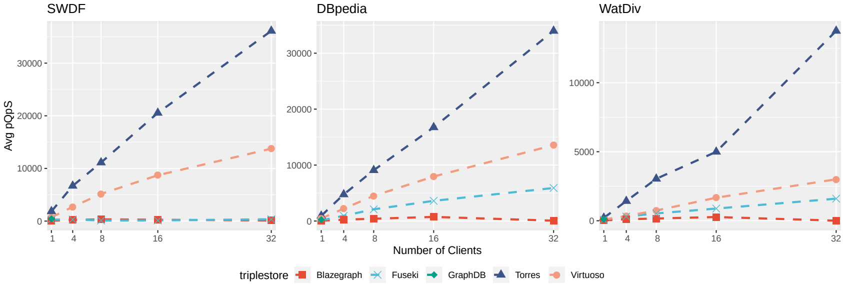
Representing Concepts as SPARQL

TENTRIS: Evaluation – WatDiv



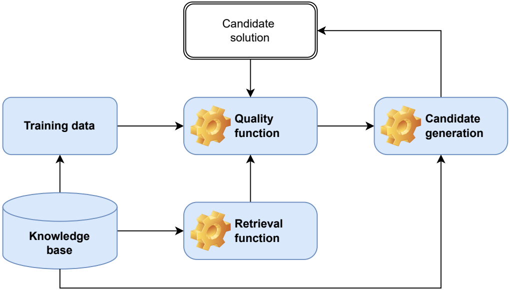
Representing Concepts as SPARQL

Tentris: Evaluation – Speedup



Learning problem

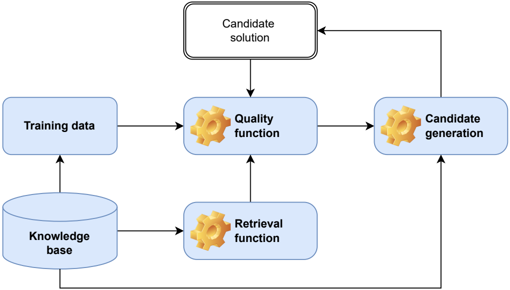
Challenges



✓ Retrieval is expensive ⇒ Represent concepts in SPARQL

Learning problem

Challenges



- ✓ Retrieval is expensive $\Rightarrow$ Represent concepts in SPARQL
- ► Quality functions are often myopic $\Rightarrow$ Exploit representation as embeddings
- ► Candidate generation is expensive $\Rightarrow$ Exploit subgraphs for priming
- ► Search space is large $\Rightarrow$ Embed concept representations

Section 5

Improving Quality Functions

Improving Quality Functions

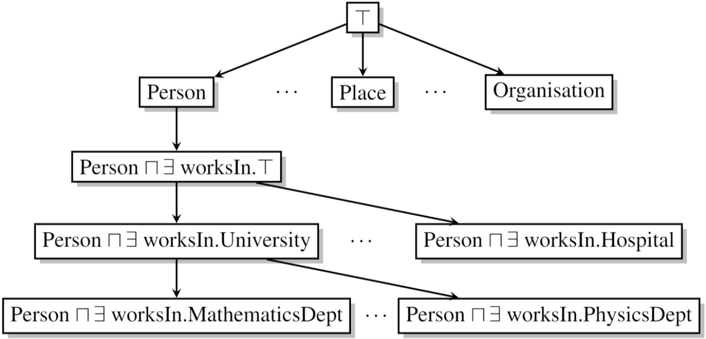
Refinement Operators

- Implement informed search in space $\mathcal{S}$ of all concepts with partial ordering $\sqsubseteq$
- Refinement operator $\rho : \mathcal{S} \rightarrow 2^{\mathcal{S}}$ with
  - $\forall x \in \rho(s) : x \sqsubseteq s$ (downward)
  - $\forall x \in \rho(s) : s \sqsubseteq x$ (upward)

Improving Quality Functions

Refinement Operators

- Implement informed search in space $\mathcal{S}$ of all concepts with partial ordering $\sqsubseteq$
- Refinement operator $\rho : \mathcal{S} \rightarrow 2^{\mathcal{S}}$ with
  - $\forall x \in \rho(s) : x \sqsubseteq s$ (downward)
  - $\forall x \in \rho(s) : s \sqsubseteq x$ (upward)



Improving Quality Functions

Quality Functions – OCEL

- Let $R(C)$ be the set of instances of C
- Let C' be the parent concept of C in the search tree

Improving Quality Functions

Quality Functions – OCEL

- Let $R(C)$ be the set of instances of C
- Let C' be the parent concept of C in the search tree
- Accuracy and accuracy gain of a concept C are defined as

$$\mathrm{acc}(C) = 1 - \frac{|E^+ \setminus R(C)| + |R(C) \cap E^-|}{|E|}$$

$$\mathrm{acc\_gain}(C) = \mathrm{acc}(C) - \mathrm{acc}(C')$$

Improving Quality Functions

Quality Functions – OCEL

- Let $R(C)$ be the set of instances of C
- Let C' be the parent concept of C in the search tree
- Accuracy and accuracy gain of a concept C are defined as

$$\mathrm{acc}(C) = 1 - \frac{|E^+ \setminus R(C)| + |R(C) \cap E^-|}{|E|}$$

$$\mathrm{acc\_gain}(C) = \mathrm{acc}(C) - \mathrm{acc}(C')$$

- The score is given by

$$\mathrm{score}(C) = \mathrm{acc}(C) + \alpha \cdot \mathrm{acc\_gain}(C) - \beta \cdot |C| \quad (\alpha, \beta \geq 0),$$

where $\alpha = 0.5$ and $\beta = 0.02$ are typical default values.

Improving Quality Functions

Quality Functions – CELOE

- Accuracy metric acc_c for CELOE:

$$\mathrm{acc}_c(C,t) = \frac{1}{t+1} \cdot \left(t \cdot \frac{|E^+ \cap R(C)|}{|E^+|} + \sqrt{\frac{|E^+ \cap R(C)|}{|R(C)|}} \right)$$

$$\mathrm{acc\_gain}_c(C) = \mathrm{acc}_c(C,t) - \mathrm{acc}_c(C',t)$$

Improving Quality Functions

Quality Functions – CELOE

- Accuracy metric acc_c for CELOE:

$$\mathrm{acc}_c(C,t) = \frac{1}{t+1} \cdot \left(t \cdot \frac{|E^+ \cap R(C)|}{|E^+|} + \sqrt{\frac{|E^+ \cap R(C)|}{|R(C)|}} \right)$$

$$\mathrm{acc\_gain}_c(C) = \mathrm{acc}_c(C,t) - \mathrm{acc}_c(C',t)$$

- $\mathrm{score}(C) = \mathrm{acc}_c(C,t) + \alpha \cdot \mathrm{acc\_gain}_c(C) - \beta \cdot |C| \quad (\alpha, \beta \geq 0)$ where typical values are $\alpha = 0.3$ and $\beta = 0.05$.

Improving Quality Functions

Quality Functions – CELOE

- Accuracy metric acc_c for CELOE:

$$\mathrm{acc}_c(C, t) = \frac{1}{t+1} \cdot \left(t \cdot \frac{|E^+ \cap R(C)|}{|E^+|} + \sqrt{\frac{|E^+ \cap R(C)|}{|R(C)|}} \right)$$

$$\mathrm{acc\_gain}_c(C) = \mathrm{acc}_c(C, t) - \mathrm{acc}_c(C', t)$$

- $\mathrm{score}(C) = \mathrm{acc}_c(C, t) + \alpha \cdot \mathrm{acc\_gain}_c(C) - \beta \cdot |C| \quad (\alpha, \beta \geq 0)$ where typical values are $\alpha = 0.3$ and $\beta = 0.05$.

Problem: Myopia

- Current metrics do not consider future accuracy of concepts

Improving Quality Functions

Quality Functions – CELOE

- Accuracy metric acc_c for CELOE:

$$\mathrm{acc}_c(C,t) = \frac{1}{t+1} \cdot \left(t \cdot \frac{|E^+ \cap R(C)|}{|E^+|} + \sqrt{\frac{|E^+ \cap R(C)|}{|R(C)|}} \right)$$

$$\mathrm{acc\_gain}_c(C) = \mathrm{acc}_c(C,t) - \mathrm{acc}_c(C',t)$$

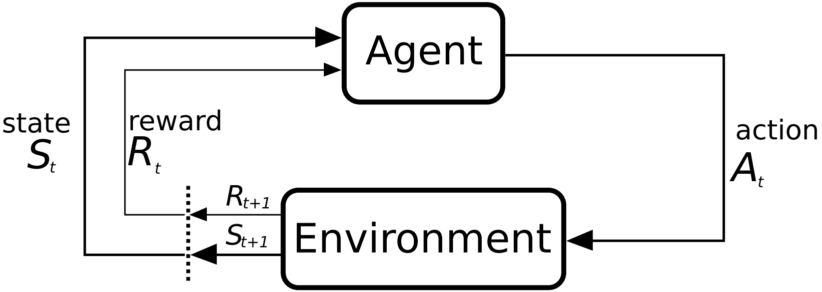
- $\mathrm{score}(C) = \mathrm{acc}_c(C,t) + \alpha \cdot \mathrm{acc\_gain}_c(C) - \beta \cdot |C| \quad (\alpha, \beta \geq 0)$
  where typical values are $\alpha = 0.3$ and $\beta = 0.05$.

Problem: Myopia

- Current metrics do not consider future accuracy of concepts
- Optimize for cumulative discounted future rewards
  [Demir and Ngonga Ngomo, 2021]

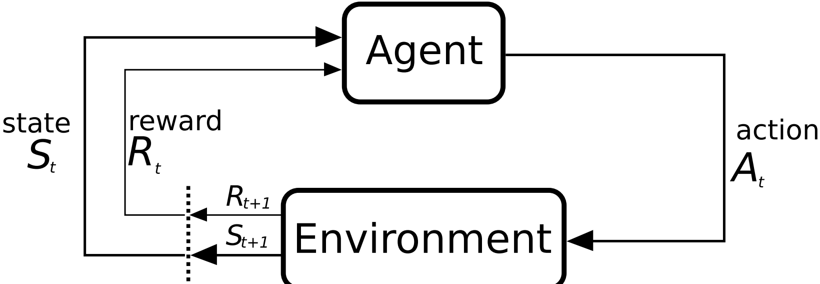
Improving Quality Functions

Reinforcement Learning



Improving Quality Functions

Reinforcement Learning



- S_t = Concept C
- $R_t = \begin{cases} 1 & \text{if } acc(C) = 1 \\ 0 & \text{else} \end{cases}$
- A_t = Transition from concept C to some concept D

Improving Quality Functions

Reinforcement Learning – Q Function

- Maximize

$$G_t = \sum_{i=0}^{n} \gamma^i R_{t+i}$$

Improving Quality Functions

Reinforcement Learning – Q Function

- Maximize

$$G_t = \sum_{i=0}^{n} \gamma^i R_{t+i}$$

- Optimize state-action value function $Q_\pi : S \times A \to \mathbb{R}$ with

$$Q_\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_\pi \left[G_t \mid S_t = \mathbf{s}, A_t = \mathbf{a} \right]$$

Improving Quality Functions

Reinforcement Learning – Q Function

- Maximize

$$G_t = \sum_{i=0}^{n} \gamma^i R_{t+i}$$

- Optimize state-action value function $Q_\pi : S \times A \to \mathbb{R}$ with

$$Q_\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_\pi \left[G_t \mid S_t = \mathbf{s}, A_t = \mathbf{a} \right]$$

- Observation: Infinite number of states as search space is infinite

Improving Quality Functions

Reinforcement Learning – Q Function

- Maximize

$$G_t = \sum_{i=0}^{n} \gamma^i R_{t+i}$$

- Optimize state-action value function $Q_\pi : S \times A \to \mathbb{R}$ with

$$Q_\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_\pi \left[G_t \mid S_t = \mathbf{s}, A_t = \mathbf{a} \right]$$

- Observation: Infinite number of states as search space is infinite
- Apply deep Q learning with target network [Mnih et al., 2015]

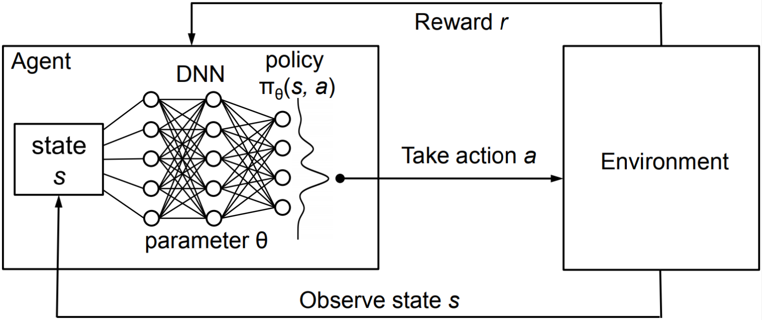
$$\mathcal{L}(\Theta_i) = \mathbb{E}_{(s,a,R,s') \sim U(\mathcal{D})} \left[\left(R + \gamma \max_{\mathbf{a}' \in A(\mathbf{s}')} Q(s', a'; \Theta_i^-) - Q(s, a; \Theta_i) \right)^2 \right]$$

Improving Quality Functions

Reinforcement Learning – DRILL

- Convolutional deep Q-Network with $\Theta = [\omega, \mathbf{W}, \mathbf{H}]$

$$\varphi([s, s', \mathbf{e}_+, \mathbf{e}_-]; \Theta) = ReLU\Big(vec(ReLU\big[\Psi([s, s', \mathbf{e}_+, \mathbf{e}_-]) * \omega\big]) \cdot \mathbf{W}\Big) \cdot \mathbf{H}$$



Source: [Mao et al., 2016]

Improving Quality Functions

TransE

- Assumptions
  - Resources and properties are vectors
  - If $(s, p, o) \in E$, then $\vec{s} + \vec{p} = \vec{o}$

Improving Quality Functions

TransE

- Assumptions
  - Resources and properties are vectors
  - If $(s, p, o) \in E$, then $\vec{s} + \vec{p} = \vec{o}$
- Translates to loss

$$L_{pos} = \sum_{(s,p,o) \in E} d(\vec{s} + \vec{p}, \vec{o})$$

Improving Quality Functions

TransE

- Assumptions
  - Resources and properties are vectors
  - If $(s, p, o) \in E$, then $\vec{s} + \vec{p} = \vec{o}$
- Translates to loss

$$L_{pos} = \sum_{(s,p,o) \in E} d(\vec{s} + \vec{p}, \vec{o})$$

- Problem 1: Loss function converges to trivial solution for vectors of arbitrary length

Improving Quality Functions

TransE

- Assumptions
  - Resources and properties are vectors
  - If $(s, p, o) \in E$, then $\vec{s} + \vec{p} = \vec{o}$
- Translates to loss

$$L_{pos} = \sum_{(s,p,o) \in E} d(\vec{s} + \vec{p}, \vec{o})$$

- Problem 1: Loss function converges to trivial solution for vectors of arbitrary length
- Solution: Normalize vectors for s and o
- Loss is now

$$L_{pos} = \sum_{(s,p,o) \in E} d(\vec{s} + \vec{p}, \vec{o}) \text{ with } ||\vec{s}|| = ||\vec{o}|| = 1$$

Improving Quality Functions

TransE

- Problem 1 not solved yet but
- Problem 2: No use of negative information

Improving Quality Functions

TransE

- Problem 1 not solved yet but
- Problem 2: No use of negative information
- Solution: Add negative information and margin $\gamma \in \mathbb{R}^+$
- Loss is now

$$L = \sum_{(s,p,o) \in E} \sum_{(s',p,o') \in S'(s,p,o)} [\gamma + d(\vec{s} + \vec{p}, \vec{o}) - d(\vec{s'} + \vec{p}, \vec{o'})]_+$$

where

- $S'(s,p,o) = sample(\{(s',p,o) | s' \in V\} \cup \{(s,p,o') | o' \in V\}, 1)$
- $S'(s,p,o) \cap E = \emptyset$
- $[x]_+ = \max\{0, x\}$

Improving Quality Functions

TransE

- Input: Training set S, margin γ, embedding dimension k
- Init
  - $\vec{p} = randomUniformSample(-6/\sqrt{k}, 6/\sqrt{k})$ for all p
  - $\vec{p} = \vec{p}/||\vec{p}||$
  - $\vec{x} = randomUniformSample(-6/\sqrt{k}, 6/\sqrt{k})$ for all $x \in V$

Improving Quality Functions

TransE

- Input: Training set S, margin γ, embedding dimension k
- Init
  - $\vec{p} = randomUniformSample(-6/\sqrt{k}, 6/\sqrt{k})$ for all p
  - $\vec{p} = \vec{p}/||\vec{p}||$
  - $\vec{x} = randomUniformSample(-6/\sqrt{k}, 6/\sqrt{k})$ for all $x \in V$
- Loop until convergence
  - $\vec{x} = \vec{x}/||\vec{x}||$ for all $x \in V$
  - $S_{batch} = sample(S, b)$ // get mini-batch of size b from S
  - $T_{batch} = T_{batch} \cup \{((s,p,o), sample(S'(s,p,o),1))\}$ for all $(s,p,o) \in S_{batch}$
  - Update embeddings w.r.t.

$$\sum_{((s,p,o),(s',p,o')) \in T_{batch}} \nabla[\gamma + d(\vec{s}+\vec{p},\vec{o}) - d(\vec{s'}+\vec{p},\vec{o'})]_+$$

Improving Quality Functions

TransE

- Input: Training set S, margin γ, embedding dimension k
- Init
  - $\vec{p} = randomUniformSample(-6/\sqrt{k}, 6/\sqrt{k})$ for all p
  - $\vec{p} = \vec{p}/||\vec{p}||$
  - $\vec{x} = randomUniformSample(-6/\sqrt{k}, 6/\sqrt{k})$ for all $x \in V$
- Loop until convergence
  - $\vec{x} = \vec{x}/||\vec{x}||$ for all $x \in V$
  - $S_{batch} = sample(S, b)$ // get mini-batch of size b from S
  - $T_{batch} = T_{batch} \cup \{((s,p,o), sample(S'(s,p,o), 1))\}$ for all $(s,p,o) \in S_{batch}$
  - Update embeddings w.r.t.
    $$\sum_{((s,p,o),(s',p,o')) \in T_{batch}} \nabla[\gamma + d(\vec{s}+\vec{p},\vec{o}) - d(\vec{s'}+\vec{p},\vec{o'})]_+$$
- Note: Learning via balanced mini-batches with random negative samples
- Note: Derivative only for portions of the loss > 0

Improving Quality Functions

Quaternions: $\mathbb{H}$

- Multiplication rules
  - $x = x_0 + ix_1 + jx_2 + kx_3$ (with $i^2 = j^2 = k^2 = ijk = -1$)
  - $ij = k, jk = i, ki = j, ji = -k, kj = -i, ik = -j$ (loss of commutativity)

Improving Quality Functions

Quaternions: $\mathbb{H}$

- Multiplication rules
  - $x = x_0 + ix_1 + jx_2 + kx_3$ (with $i^2 = j^2 = k^2 = ijk = -1$)
  - $ij = k, jk = i, ki = j, ji = -k, kj = -i, ik = -j$ (loss of commutativity)
- Can define embeddings in this space: QuatE
  - $\vec{s}, \vec{p}, \vec{o} \in \mathbb{H}^k$
  - $\vec{p}^{\triangleleft} = \vec{p}/||\vec{p}||$ (normalized vector $\vec{p}$)
  - Scoring function $\varphi(s, p, o) = (\vec{s} \otimes \vec{p}^{\triangleleft}) \cdot \vec{o}$, where

Improving Quality Functions

Quaternions: $\mathbb{H}$

- Multiplication rules
  - $x = x_0 + ix_1 + jx_2 + kx_3$ (with $i^2 = j^2 = k^2 = ijk = -1$)
  - $ij = k, jk = i, ki = j, ji = -k, kj = -i, ik = -j$ (loss of commutativity)
- Can define embeddings in this space: QuatE
  - $\vec{s}, \vec{p}, \vec{o} \in \mathbb{H}^k$
  - $\vec{p}^{\triangleleft} = \vec{p}/||\vec{p}||$ (normalized vector $\vec{p}$)
  - Scoring function $\varphi(s, p, o) = (\vec{s} \otimes \vec{p}^{\triangleleft}) \cdot \vec{o}$, where
    - $\otimes$ is the Hamiltonian product ($\mathbb{H} \times \mathbb{H} \to \mathbb{H}$)
    - $\cdot$ is the quaternion inner product ($\mathbb{H} \times \mathbb{H} \to \mathbb{R}$)

Improving Quality Functions

Quaternions: $\mathbb{H}$

- Multiplication rules
  - $x = x_0 + ix_1 + jx_2 + kx_3$ (with $i^2 = j^2 = k^2 = ijk = -1$)
  - $ij = k, jk = i, ki = j, ji = -k, kj = -i, ik = -j$ (loss of commutativity)
- Can define embeddings in this space: QuatE
  - $\vec{s}, \vec{p}, \vec{o} \in \mathbb{H}^k$
  - $\vec{p}^{\triangleleft} = \vec{p}/||\vec{p}||$ (normalized vector $\vec{p}$)
  - Scoring function $\varphi(s, p, o) = (\vec{s} \otimes \vec{p}^{\triangleleft}) \cdot \vec{o}$, where
    - $\otimes$ is the Hamiltonian product ($\mathbb{H} \times \mathbb{H} \rightarrow \mathbb{H}$)
    - $\cdot$ is the quaternion inner product ($\mathbb{H} \times \mathbb{H} \rightarrow \mathbb{R}$)
  - Loss function over training data Γ with $Y_{spo} \in \{-1, +1\}$ is given by
    $$\min_{\vec{s},\vec{p},\vec{o}} \sum_{(s,p,o)\in\Gamma} \log(1 + exp(-Y_{spo}\varphi(s, p, o)))$$

Improving Quality Functions

Quaternions: $\mathbb{H}$

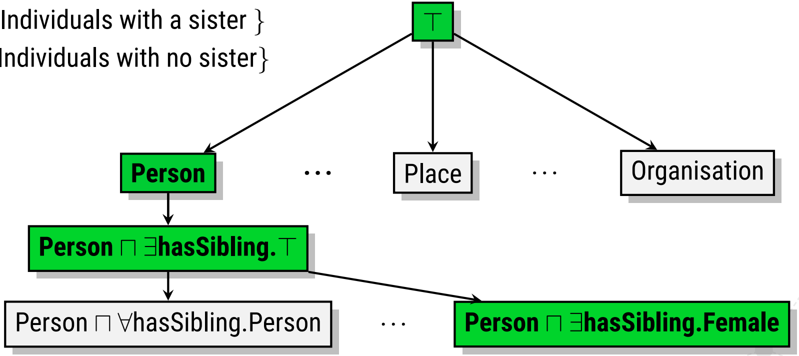
- Multiplication rules
  - $x = x_0 + ix_1 + jx_2 + kx_3$ (with $i^2 = j^2 = k^2 = ijk = -1$)
  - $ij = k, jk = i, ki = j, ji = -k, kj = -i, ik = -j$ (loss of commutativity)
- Can define embeddings in this space: QuatE
  - $\vec{s}, \vec{p}, \vec{o} \in \mathbb{H}^k$
  - $\vec{p}^{\triangleleft} = \vec{p}/||\vec{p}||$ (normalized vector $\vec{p}$)
  - Scoring function $\varphi(s,p,o) = (\vec{s} \otimes \vec{p}^{\triangleleft}) \cdot \vec{o}$, where
    - $\otimes$ is the Hamiltonian product ($\mathbb{H} \times \mathbb{H} \rightarrow \mathbb{H}$)
    - $\cdot$ is the quaternion inner product ($\mathbb{H} \times \mathbb{H} \rightarrow \mathbb{R}$)
  - Loss function over training data Γ with $Y_{spo} \in \{-1, +1\}$ is given by $\min_{\vec{s},\vec{p},\vec{o}} \sum_{(s,p,o)\in\Gamma} \log(1 + exp(-Y_{spo}\varphi(s,p,o)))$
- Similar construction for octonions

Unsupervised Learning – Training Data

- Follow refinement path at random
- Select concept C
- Set $E^+ \subseteq R(C)$ and $E^- \cap R(C) = \emptyset$

$E^+ = \{$Individuals with a sister $\}$

$E^- = \{$Individuals with no sister$\}$



Improving Quality Functions

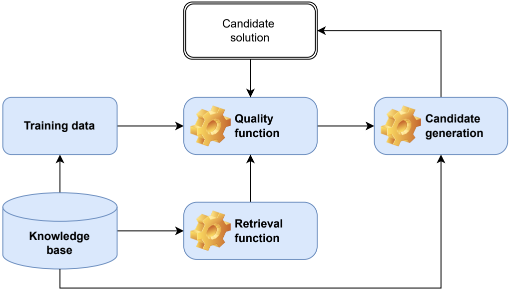
Evaluation

- Used Family und BioPax datasets
- Evaluation on 114 learning problems

Approaches	F1	Acc	Runtime	# Exp.
CELOE	$.995 \pm 0.03$	$.993 \pm 0.04$	7.5 ± 1.1	33.5 ± 129.3
OCEL	*	1.00 ± 0.00	11.0 ± 1.4	2271.6 ± 1269.2
ELTL	$.990 \pm 0.06$	$.984 \pm 0.09$	8.1 ± 1.6	*
DRILL	1.00 ± 0.00	1.00 ± 0.00	1.1 ± 0.5	9.88 ± 38.5

Learning problem

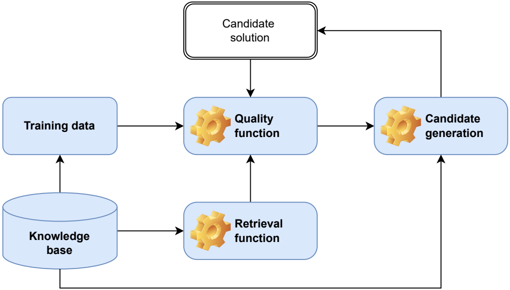
Challenges



- ✓ Retrieval is expensive ⇒ Represent concepts in SPARQL
- ✓ Quality functions are often myopic ⇒ Exploit representation as embeddings

Learning problem

Challenges



- ✓ Retrieval is expensive ⇒ Represent concepts in SPARQL
- ✓ Quality functions are often myopic ⇒ Exploit representation as embeddings
- ▶ Candidate generation is expensive ⇒ Exploit subgraphs for priming
- ▶ Search space is large ⇒ Embed concept representations

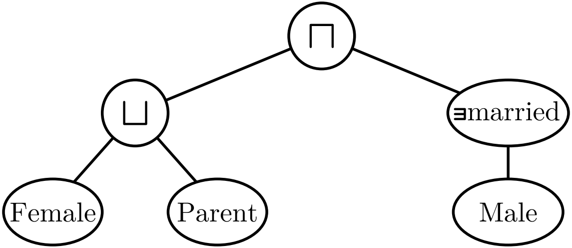
Section 6

Learning with Priming

Learning with Priming

EvoLearner – Idea

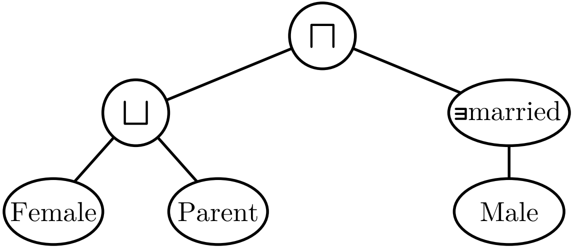
- Represent concepts as trees, e.g.,
  $(\texttt{Female} \sqcup \texttt{Parent}) \sqcap \exists \texttt{married}.\texttt{Male}$



Learning with Priming

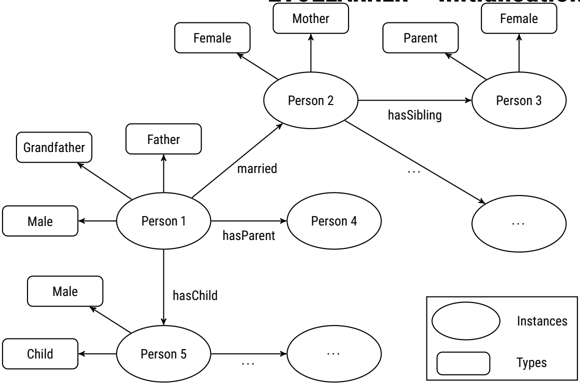
EvoLearner – Idea

- Represent concepts as trees, e.g.,
  $(\texttt{Female} \sqcup \texttt{Parent}) \sqcap \exists\texttt{married.Male}$
- Learn in evolutionary fashion using genetic programming
- Exploit priming effect (remember the green apple)
- Intuition: An individual is an overlap several concepts [Heindorf et al., 2022]



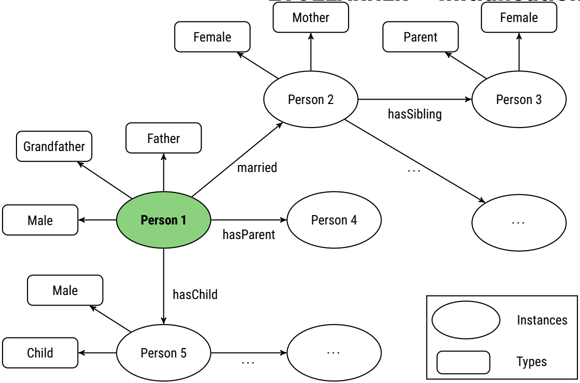
Learning with Priming

EvoLearner – Initialisation



Learning with Priming

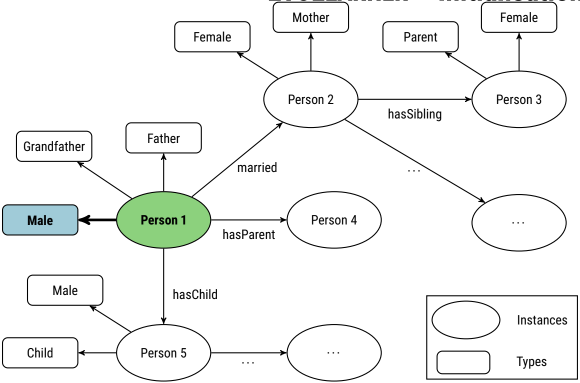
EvoLearner – Initialisation



1. Select a positive example e^+ and one of its types:

Learning with Priming

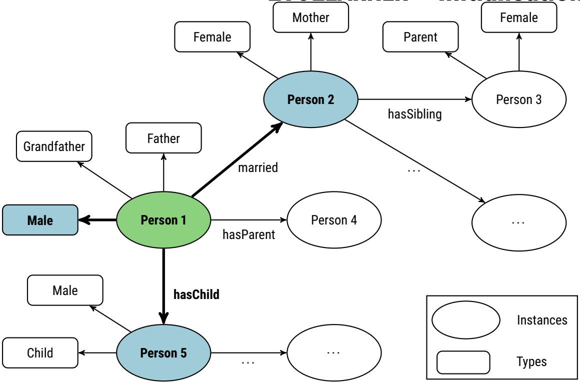
EVOLEARNER – Initialisation



1. Select a positive example e^+ and one of its types: `Male`

Learning with Priming

EvoLearner – Initialisation

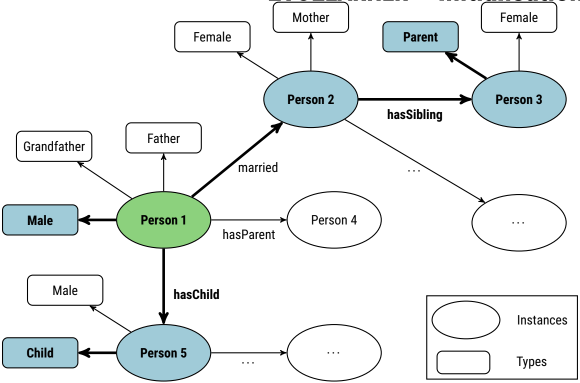


1. Select a positive example e^+ and one of its types: `Male`
2. Randomly select up to *maxT* outgoing triples of e^+:

   $\texttt{Male} \sqcap (\exists \texttt{married} \ldots \sqcap \exists \texttt{hasChild} \ldots)$

Learning with Priming

EvoLearner – Initialisation



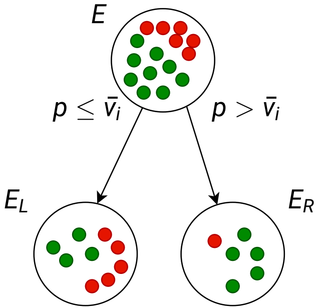
1. Select a positive example e^+ and one of its types: `Male`
2. Randomly select up to *maxT* outgoing triples of e^+:
   $\texttt{Male} \sqcap (\exists\texttt{married} \ldots \sqcap \exists\texttt{hasChild} \ldots)$
3. Complete incomplete subconcepts:
   $\texttt{Male} \sqcap ((\exists\texttt{married}.\,\exists\texttt{hasSibling.Parent}) \sqcap (\exists\texttt{hasChild.Child}))$

Learning with Priming

EvoLearner – Data Properties

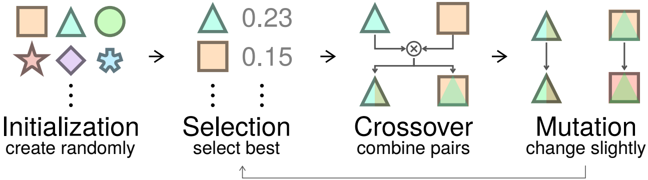
- Given a data property d from the knowledge base $\mathcal{K}$ and a set E of positive and negative examples
- We precompute up to k splits of the form $d \leq \bar{v}_i$ per data property
- Splits are computed to maximize information gain:

$$IG(E, \bar{v}_i) = H(E) - H(E|\bar{v}_i) = H(E) - \left(\frac{|E_L|}{|E|} H(E_L) + \frac{|E_R|}{|E|} H(E_R) \right)$$



Learning with Priming

EvoLearner – Training



Learning with Priming

EvoLearner – Evaluation

Learn. Problem	EvoLearner (ours)	DL-Learner (CELOE)	DL-Learner (OCEL)	Aleph	SPaCEL
Carcinogenesis	0.70 ± 0.12	$\mathbf{0.71} \pm \mathbf{0.01}$	*no results*	0.46 ± 0.12	0.60 ± 0.08
Family	1.00 ± 0.01	0.98 ± 0.05	$\mathbf{1.00} \pm \mathbf{0.00}$	–	0.97 ± 0.11
Hepatitis	$\mathbf{0.79} \pm \mathbf{0.08}$	0.61 ± 0.03	*no results*	0.38 ± 0.12	*no results*
Lymphography	0.84 ± 0.09	0.78 ± 0.10	$\mathbf{0.85} \pm \mathbf{0.10}$	0.84 ± 0.09	0.75 ± 0.13
Mammographic	$\mathbf{0.81} \pm \mathbf{0.06}$	0.64 ± 0.01	0.78 ± 0.08	0.48 ± 0.08	0.64 ± 0.06
Mutagenesis	$\mathbf{1.00} \pm \mathbf{0.00}$	0.93 ± 0.14	*timeout*	0.43 ± 0.47	$\mathbf{1.00} \pm \mathbf{0.00}$
NCTRER	$\mathbf{1.00} \pm \mathbf{0.00}$	0.74 ± 0.01	0.94 ± 0.06	0.71 ± 0.18	$\mathbf{1.00} \pm \mathbf{0.00}$
Premier League	$\mathbf{1.00} \pm \mathbf{0.00}$	0.99 ± 0.04	0.81 ± 0.13	0.94 ± 0.11	0.98 ± 0.04
Pyrimidine	$\mathbf{0.91} \pm \mathbf{0.14}$	0.84 ± 0.15	0.84 ± 0.22	0.90 ± 0.32	0.86 ± 0.29

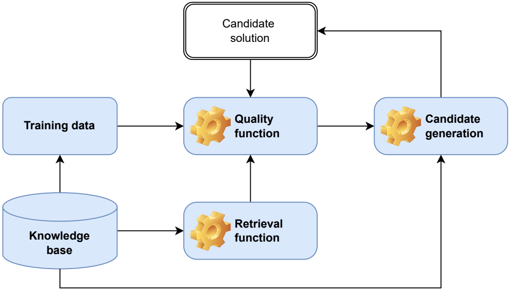
Learning with Priming

EvoLearner – Ablation Study

Learning Problem	EvoLearner (ours)	Without Rand. Walk Init.	Without Data Properties	Without Both
Carcinogenesis	0.70 ± 0.12	0.60 ± 0.21	0.63 ± 0.13	0.62 ± 0.13
Family	1.00 ± 0.01	0.87 ± 0.13	–	0.86 ± 0.14
Hepatitis	0.79 ± 0.08	0.67 ± 0.15	0.46 ± 0.14	0.47 ± 0.13
Lymphography	0.84 ± 0.09	0.83 ± 0.11	–	0.83 ± 0.09
Mammographic	0.81 ± 0.06	0.78 ± 0.08	0.77 ± 0.07	0.75 ± 0.06
Mutagenesis	1.00 ± 0.00	1.00 ± 0.00	0.44 ± 0.48	0.50 ± 0.51
NCTRER	1.00 ± 0.00	1.00 ± 0.00	0.74 ± 0.05	0.75 ± 0.05
Premier League	1.00 ± 0.00	0.98 ± 0.04	0.50 ± 0.23	0.50 ± 0.22
Pyrimidine	0.91 ± 0.14	0.83 ± 0.22	0.67 ± 0.00	0.67 ± 0.00

Learning problem

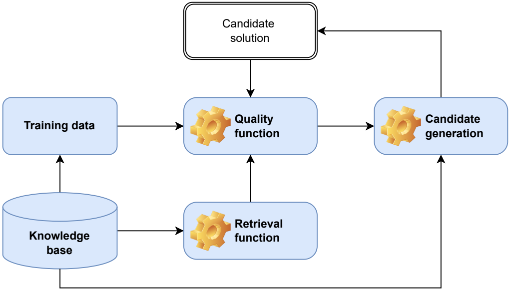
Challenges



- ✓ Retrieval is expensive $\Rightarrow$ Represent concepts in SPARQL
- ✓ Quality functions are often myopic $\Rightarrow$ Exploit representation as embeddings

Learning problem

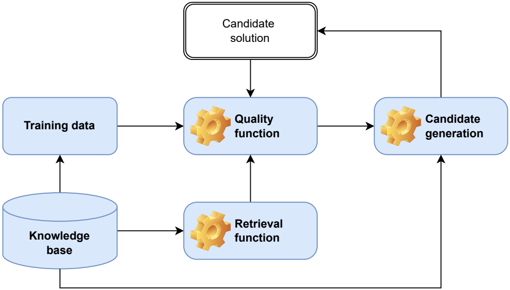
Challenges



- ✓ Retrieval is expensive ⇒ Represent concepts in SPARQL
- ✓ Quality functions are often myopic ⇒ Exploit representation as embeddings
- ✓ Candidate generation is expensive ⇒ Exploit subgraphs for priming
- ▶ Search space is large ⇒ Represent concepts as embeddings

Learning problem

Challenges



- ✓ Retrieval is expensive ⇒ Represent concepts in SPARQL
- ✓ Quality functions are often myopic ⇒ Exploit representation as embeddings
- ✓ Candidate generation is expensive ⇒ Exploit subgraphs for priming
- ► Search space is large ⇒ Represent concepts as embeddings [Kouagou et al., 2022]

Section 7

CLIP

CLIP

Concept Lengths

- $length(A) = length(\top) = length(\bot) = 1$ (A atomic concept)

CLIP

Concept Lengths

- $length(A) = length(\top) = length(\bot) = 1$ (A atomic concept)
- $length(\neg C) = 1 + length(C)$, for all concepts C

CLIP

Concept Lengths

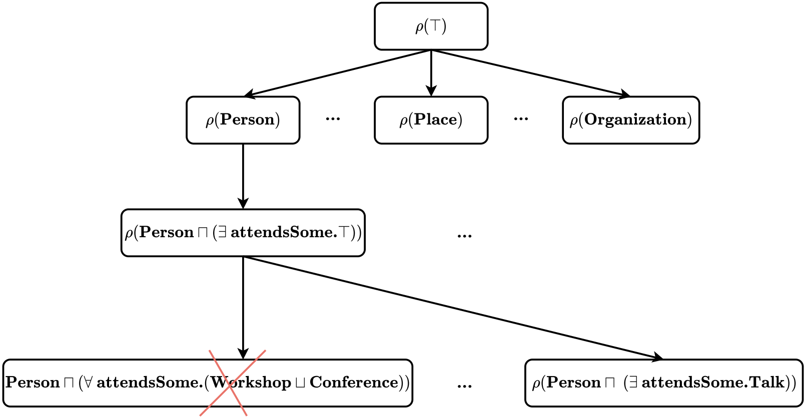
- $length(A) = length(\top) = length(\bot) = 1$ (A atomic concept)
- $length(\neg C) = 1 + length(C)$, for all concepts C
- $length(\exists\, r.C) = length(\forall\, r.C) = 2 + length(C)$, for all concepts C

Concept Lengths

- $length(A) = length(\top) = length(\bot) = 1$ (A atomic concept)
- $length(\neg C) = 1 + length(C)$, for all concepts C
- $length(\exists\, r.C) = length(\forall\, r.C) = 2 + length(C)$, for all concepts C
- $length(C \sqcup D) = length(C \sqcap D) = 1 + length(C) + length(D)$, for all concepts C and D.

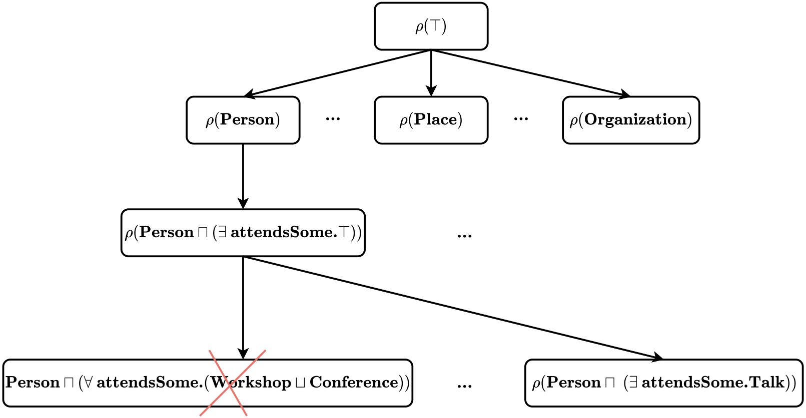
CLIP

Approach



CLIP

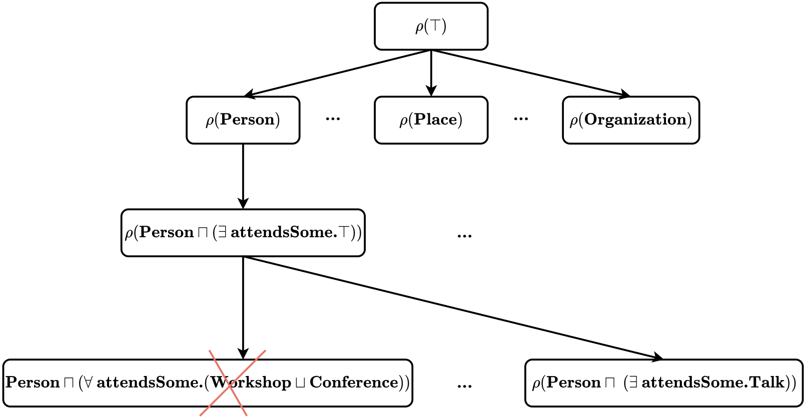
Approach



- Learn concept lengths

CLIP

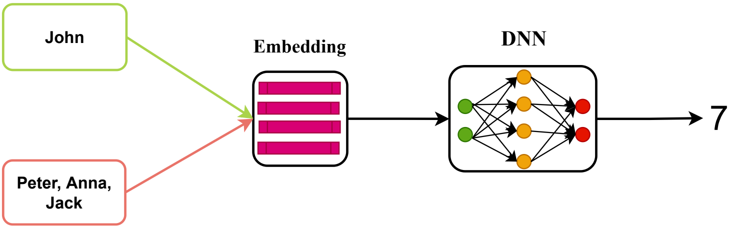
Approach



- Learn concept lengths
- Predict target concept length and discard longer refinements

CLIP

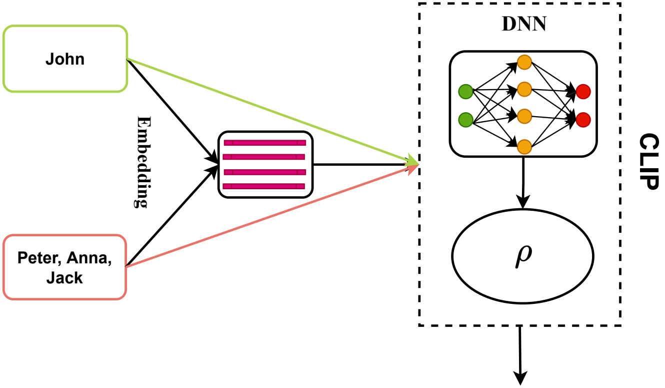
Concept Length Prediction



- Input: positive and negative examples
- Output: length of the target concept

CLIP

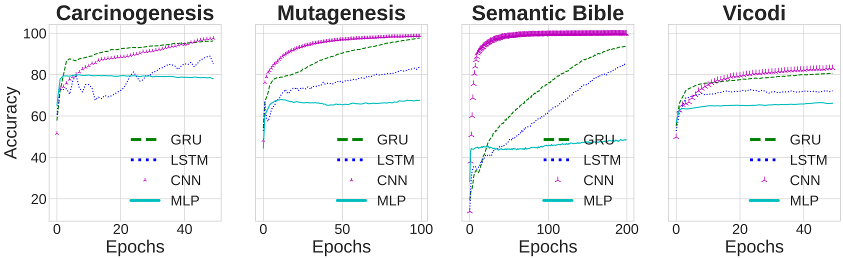
Concept Learning

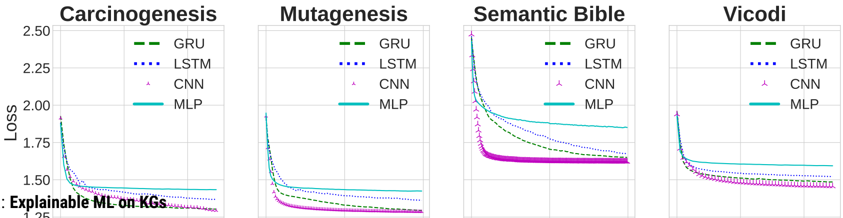


Male $\sqcap \exists$ hasParent.($\exists$ hasChild.Female)

CLIP

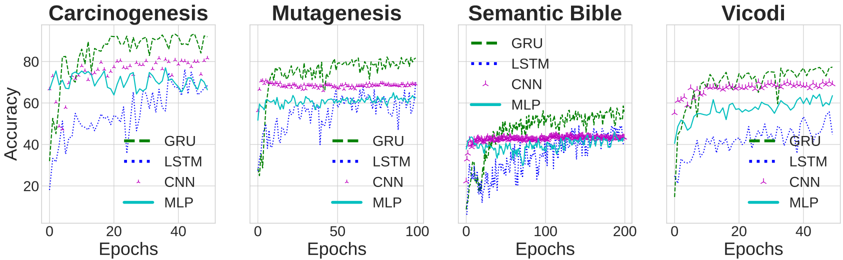
Training

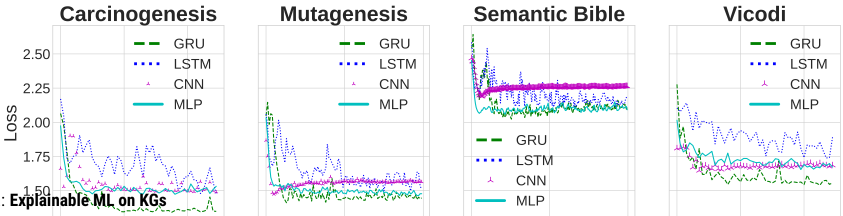




CLIP

Validation





CLIP

Network Architecture

	Carcinogenesis					Mutagenesis				
Metric	**LSTM**	**GRU**	**CNN**	**MLP**	**RM**	**LSTM**	**GRU**	**CNN**	**MLP**	**RM**
Train. Acc.	0.89	0.96	0.97	0.80	0.48	0.83	0.97	0.98	0.68	0.33
Val. Acc.	0.76	0.93	0.82	0.77	0.48	0.70	0.82	0.71	0.65	0.35
Test Acc.	0.92	**0.95**	0.84	0.80	0.49	0.78	**0.85**	0.70	0.68	0.33
Test F1	0.88	**0.92**	0.71	0.59	0.33	0.76	**0.85**	0.70	0.67	0.32

	Semantic Bible					Vicodi				
Metric	**LSTM**	**GRU**	**CNN**	**MLP**	**RM**	**LSTM**	**GRU**	**CNN**	**MLP**	**RM**
Train. Acc.	0.85	0.93	0.99	0.68	0.33	0.73	0.81	0.83	0.66	0.28
Val. Acc.	0.49	0.58	0.44	0.46	0.26	0.55	0.77	0.70	0.64	0.30
Test Acc.	0.52	**0.53**	0.37	0.40	0.25	0.66	**0.80**	0.69	0.66	0.29
Test F1	0.27	**0.38**	0.20	0.22	0.16	0.45	**0.50**	0.45	0.38	0.20

CLIP

Comparison with SOTA

Carcinogenesis				
Metric	**CELOE**	**OCEL**	**ELTL**	**CLIP**
Acc. ↑	0.78 ± 0.27	0.89 ± 0.31	0.58 ± 0.46	**0.99** ± 0.00
F1 ↑	0.62 ± 0.46	–	0.51 ± 0.47	**0.96**∗ ± 0.10
Runtime (min) ↓	0.93 ± 0.94	3.01 ± 0.72	0.75 ± 0.07	**0.10**∗ ± 0.09
Length ↓	**1.69** ± 0.89	7.81 ± 6.88	1.04 ± 0.39	2.00 ± 1.28

Mutagenesis				
Metric	**CELOE**	**OCEL**	**ELTL**	**CLIP**
Acc. ↑	0.99 ± 0.00	0.71 ± 0.45	0.37 ± 0.43	**0.99** ± 0.00
F1 ↑	0.81 ± 0.35	–	0.29 ± 0.40	**0.93**∗ ± 0.18
Runtime (min) ↓	0.70 ± 0.77	2.39 ± 0.18	0.29 ± 0.16	**0.07**∗ ± 0.05
Length ↓	2.79 ± 1.17	12.63 ± 7.03	1.10 ± 0.81	**2.20** ± 1.16

Semantic Bible				
Metric	**CELOE**	**OCEL**	**ELTL**	**CLIP**
Acc. ↑	0.99 ± 0.02	0.66 ± 0.47	0.59 ± 0.37	**0.99** ± 0.00
F1 ↑	0.97 ± 0.10	–	0.57 ± 0.38	**0.98** ± 0.05
Runtime (min) ↓	0.47 ± 0.80	22.15 ± 96.55	0.09 ± 0.07	**0.06**∗ ± 0.05
Length ↓	3.85 ± 2.44	9.54 ± 5.73	1.38 ± 1.76	**2.52**∗ ± 1.26

Vicodi				
Metric	**CELOE**	**OCEL**	**ELTL**	**CLIP**
Acc. ↑	0.29 ± 0.44	0.25 ± 0.43	0.28 ± 0.44	**0.99**∗ ± 0.00
F1 ↑	0.25 ± 0.44	–	0.25 ± 0.44	**0.97**∗ ± 0.09
Runtime (min) ↓	1.30 ± 0.71	4.78 ± 1.12	1.81 ± 0.46	**0.16**∗ ± 0.12
Length ↓	10.79 ± 6.30	11.54 ± 6.00	11.14 ± 6.11	**1.68**∗ ± 0.98

Section 8

Summary

Summary

Open Questions

- Tensors: Variable ordering? Compressed data structure?
- RL: Reduce training costs? Hyperparameters? Embeddings?
- Evolutionary learning: Myopia? Runtime? Continuous data?

Summary

Open Questions

Holy Grail

- Can the selection of representations be automated?
- LEMUR and ENEXA

- Tensors: Variable ordering? Compressed data structure?
- RL: Reduce training costs? Hyperparameters? Embeddings?
- Evolutionary learning: Myopia? Runtime? Continuous data?

Thank You!

Joint works with Alexander Bigerl, Caglar Demir, Hamada Zahera, N'Dah Jean Kouagou, Nikoloas Karalis, Stefan Heindorf, Mohamed Sherif, Muhammed Saleem, and many more

Thank You!
Questions?

- https://dice-research.org
- https://twitter.com/DiceResearch
- https://twitter.com/NgongaAxel

References I

[Barr, 1989] Barr, A. H. (1989).
The einstein summation notation: Introduction and extensions.
SIGGRAPH 89 Course notes# 30 on Topics in Physically-Based Modeling, pages J1–J12.

[Bigerl et al., 2020] Bigerl, A., Conrads, F., Behning, C., Sherif, M. A., Saleem, M., and Ngonga Ngomo, A.-C. (2020).
Tentris–a tensor-based triple store.
In *International Semantic Web Conference*, pages 56–73. Springer.

[Bin et al., 2016] Bin, S., Bühmann, L., Lehmann, J., and Ngonga Ngomo, A.-C. (2016).
Towards sparql-based induction for large-scale rdf data sets.
In *ECAI 2016*, pages 1551–1552. IOS Press.

References II

[Demir and Ngonga Ngomo, 2021] Demir, C. and Ngonga Ngomo, A.-C. (2021).
Drill–deep reinforcement learning for refinement operators in *alc*.
arXiv preprint arXiv:2106.15373.

[Heindorf et al., 2022] Heindorf, S., Blübaum, L., Düsterhus, N., Werner, T., Golani, V. N., Demir, C., and Ngonga Ngomo, A.-C. (2022).
Evolearner: Learning description logics with evolutionary algorithms.
In *Proceedings of the ACM Web Conference 2022*, pages 818–828.

[Kahneman, 2011] Kahneman, D. (2011).
Thinking, fast and slow.
Macmillan.

References III

[Kouagou et al., 2022] Kouagou, N., Heindorf, S., Demir, C., and Ngonga Ngomo, A.-C. (2022).
Learning concept lengths accelerates concept learning in alc.
Proceedings of ESWC.

[Lehmann and Hitzler, 2010] Lehmann, J. and Hitzler, P. (2010).
Concept learning in description logics using refinement operators.
Machine Learning, 78(1):203–250.

[Mao et al., 2016] Mao, H., Alizadeh, M., Menache, I., and Kandula, S. (2016).
Resource management with deep reinforcement learning.
In *Proceedings of the 15th ACM workshop on hot topics in networks*, pages 50–56.

References IV

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015).
Human-level control through deep reinforcement learning.
nature, 518(7540):529–533.