PADERBORN UNIVERSITY

Concept Learning in Description Logics

Neurosymbolic Concept Learning

Axel Ngonga

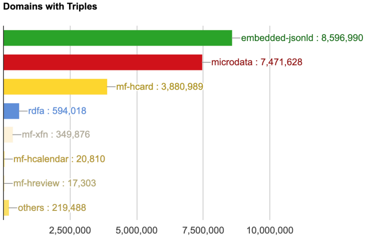
August 14, 2023

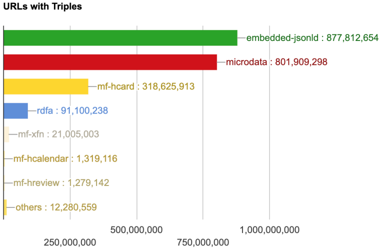
Section 1

Motivation

Introduction

Data Web



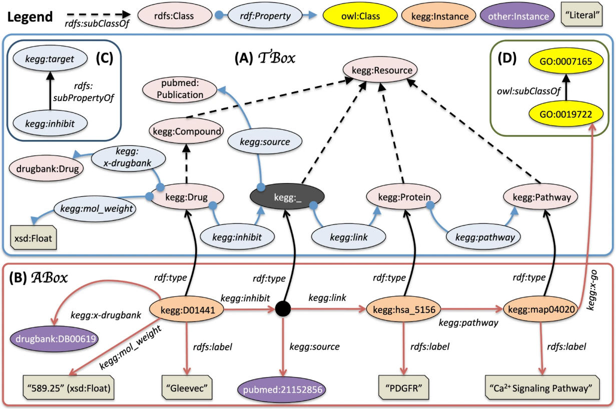


- RDF knowledge bases are now first-class citizens of the Web
- Approx. 50% of websites contain RDF[1]
- 2+ billion URLs contain RDF statements
- Ca. 100 billion statements in Linked Open Data

[1] See http://webdatacommons.org/structureddata/#results-2022-1

Introduction

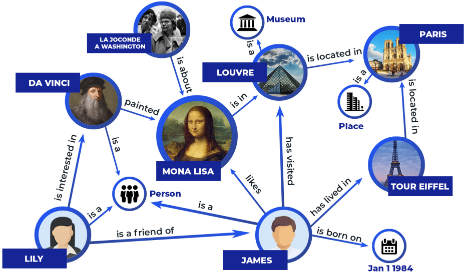
Description Logics



- Terminology of RDF datasets in description logics
- Popular DLs include $\mathcal{ELH}$ (e.g., for biomedical domain), $\mathcal{ALC}$ (e.g., for ML-driven applications), and $\mathcal{SROIQ}$ (e.g., on the Web)

Motivation

Example



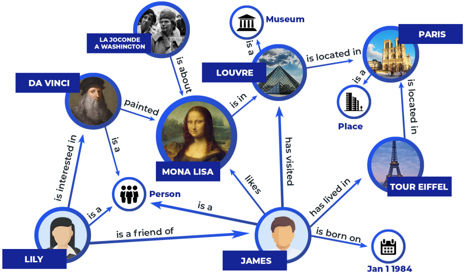
2

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

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

Motivation

Example



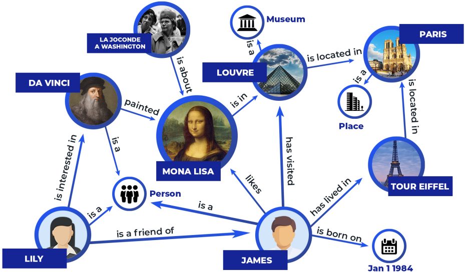
2

- $E^+ = \{Louvre, TourEiffel\}, E^- = \{Lily, James\}$
- Neural solution: $e(v_i) = \varphi\left(\bigoplus_{v_j \in \mathcal{N}_i} e(v_j), e(v_i)\right)$
- Pro: Time-efficient
- Contra: Unintelligible, does not exploits background knowledge

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

Motivation

Example



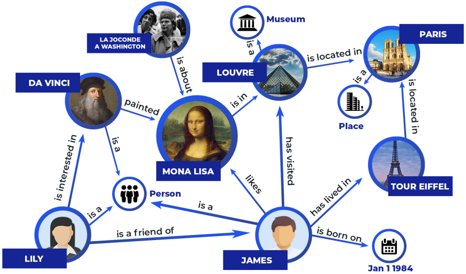
3

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

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

Motivation

Example



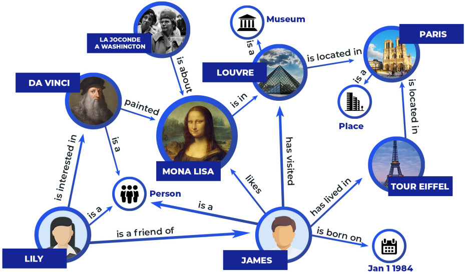
3

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

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

Motivation

Example



[3]

- $E^+ = \{Louvre, TourEiffel\}, E^- = \{Lily, James\}$
- Solution in $\mathcal{ALCO}$: $\mathcal{H} = \{\exists\, isLocatedIn.\{Paris\}\}$
- Pro: explainable, exploits background knowledge
- Contra: slow :-(

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

Goal

- Attempt neuro-symbolic learning on knowledge graphs
- Exploit time efficiency of neural approaches
- Keep explainability of symbolic approaches

Section 2

Class Expression Learning

Class Expression Learning

Formal definition

- Supervised learning with background knowledge (adapted from [?])
- 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 [?])
- 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. $\forall e^- \in E^- : \mathcal{K} \not\models H(e^-)$

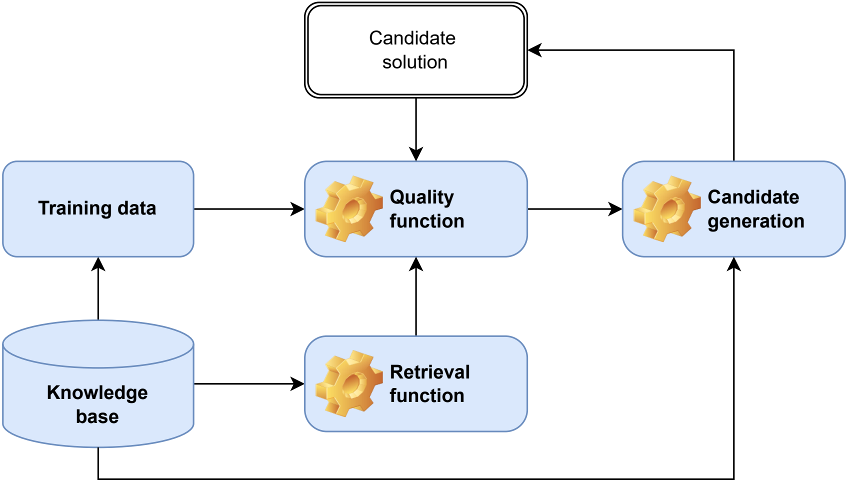
Class Expression Learning

Formal definition

- Supervised learning with background knowledge (adapted from [?])
- 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. $\forall e^- \in E^- : \mathcal{K} \not\models H(e^-)$
- Practically, aim to find $H \in \underset{C \in \mathcal{L}}{argmax}\ Q(C)$ [?]

Class Expression Learning

Common Approach



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 $\rho(x) \sqsubseteq x$ [?]

Example: Refinement Operator

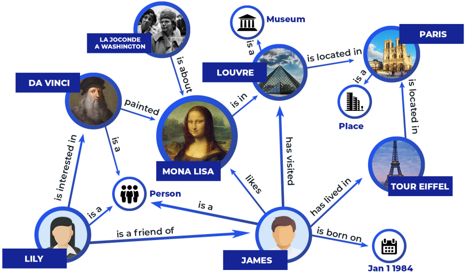
- Let $(S, \sqsubseteq)$ be a space with a quasi-ordering
- A top-down refinement operator $\rho : S \rightarrow 2^S$ is a mapping with $\rho(x) \sqsubseteq x$ [?]

Example

- Let S be the set of all concepts in our language $\mathcal{L} = \mathcal{ALC}$
- The following operator ρ is a top-down refinement operator
- $$\rho(C) = \begin{cases} C & \\ N_C \cup \neg N_C \cup \{\exists r_j.\rho(C_i)\} & \text{if } C = \top \\ \rho(D) & \text{if } D \sqsubseteq C \\ C \sqcap D & \text{with } D \in N_C \\ C \sqcap \exists r.\rho(D) & \text{with } D \in N_C \end{cases}$$

Class Expression Learning

Example



4

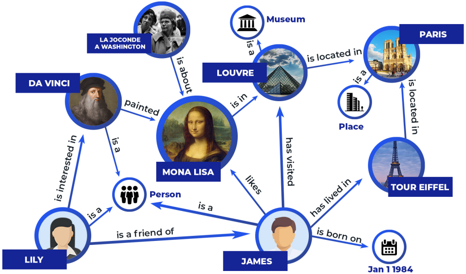
- $E^{+} = \{Louvre, TourEiffel\}, E^{-} = \{Lily, James\}$

---

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

Class Expression Learning

Example



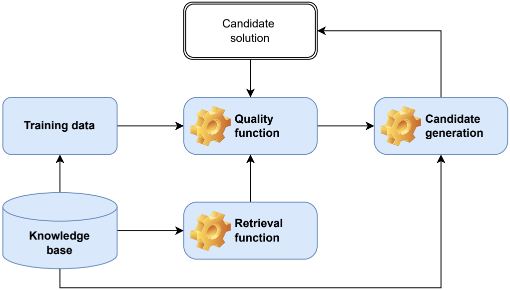
4

- $E^+ = \{Louvre, TourEiffel\}, E^- = \{Lily, James\}$
- $\rho(\top) = \{Person, Museum, Place, \exists is\_located\_in.\top, \ldots\}$

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

Learning problem

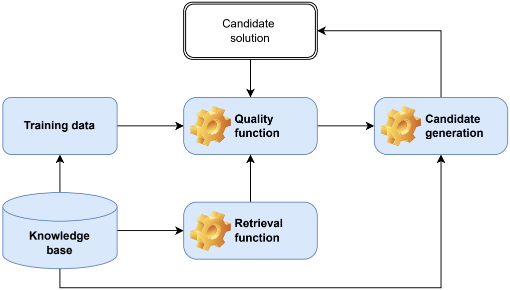
Challenges



- Retrieval is expensive

Learning problem

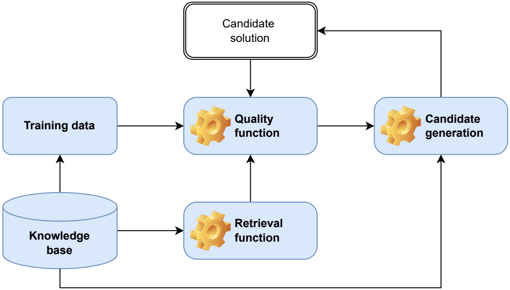
Challenges



- Retrieval is expensive $\Rightarrow$ Exploit SPARQL
- Quality functions are often myopic

Learning problem

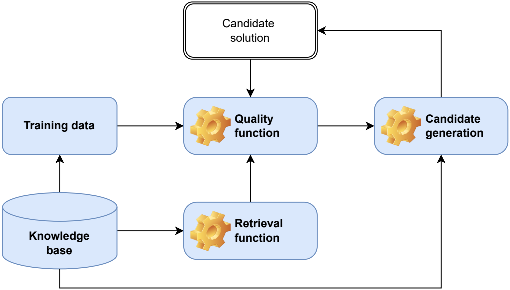
Challenges



- Retrieval is expensive $\Rightarrow$ Exploit SPARQL
- Quality functions are often myopic $\Rightarrow$ Exploit embeddings
- Candidate generation is expensive

Learning problem

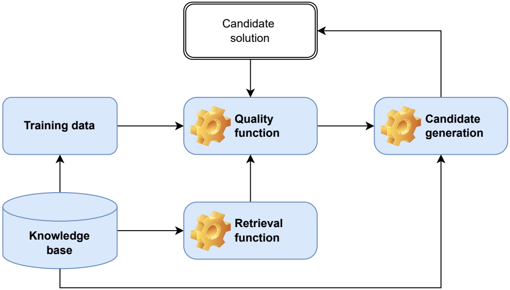
Challenges



- Retrieval is expensive $\Rightarrow$ Exploit SPARQL
- Quality functions are often myopic $\Rightarrow$ Exploit embeddings
- Candidate generation is expensive $\Rightarrow$ Exploit priming

Learning problem

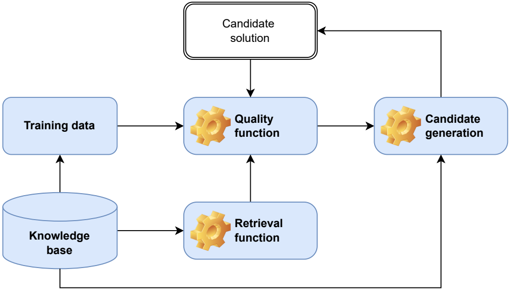
Challenges



- Retrieval is expensive $\Rightarrow$ Exploit SPARQL
- Quality functions are often myopic $\Rightarrow$ Exploit embeddings
- Candidate generation is expensive $\Rightarrow$ Exploit priming
- Search space is large

Learning problem

Challenges



- Retrieval is expensive $\Rightarrow$ Exploit SPARQL
- Quality functions are often myopic $\Rightarrow$ Exploit embeddings
- Candidate generation is expensive $\Rightarrow$ Exploit priming
- Search space is large $\Rightarrow$ Prune by length

Section 3

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 [?]

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 [?]

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 [?]

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 [?]

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` $\{\tau(C, \texttt{?var})\}$
$C_1 \sqcap \ldots \sqcap C_n$	$\{\tau(C_1, ?var) \ldots \tau(C_n, ?var)\}$

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 [?]

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` $\{\tau(C_n, ?var)\}$

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 [?]

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

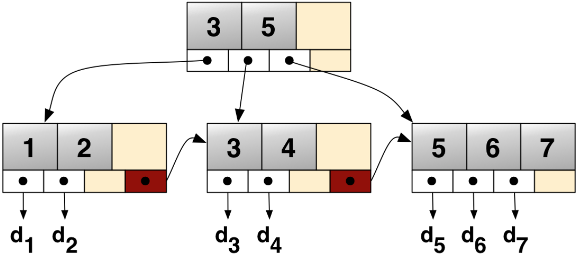
- Assume closed world and fully materialized knowledge graph
- Retrieval in $\mathcal{ALC}$ can be realized by representing concepts as SPARQL queries [?]

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` $\{\tau(C_n, ?\text{var})\}$
$\exists\, r.C$	$\{$`?var r ?s.` $\tau(C, ?\text{s})\}$
$\forall\, r.C$	`{ ?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 ) }`

Representing Concepts as SPARQL

Storage Solutions

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

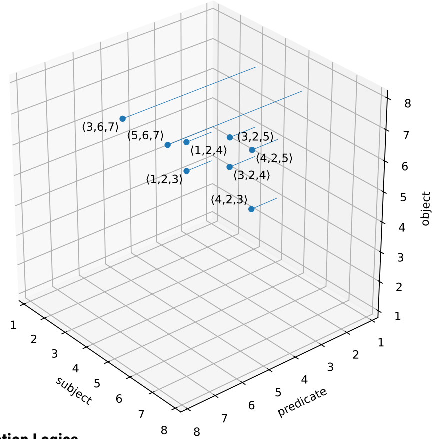


Representing Concepts as SPARQL

Tentris: Idea

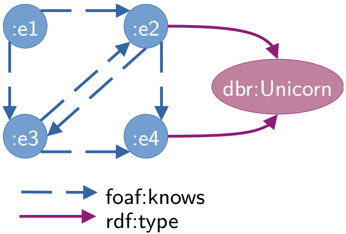
Idea [?]

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



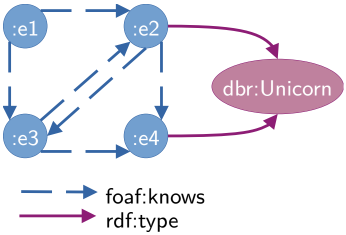
Representing Concepts as SPARQL

From RDF to Tensors



Representing Concepts as SPARQL

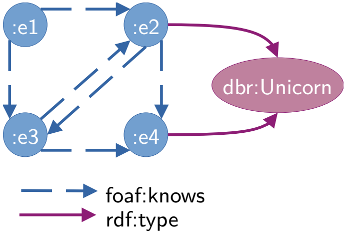
From RDF to Tensors



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

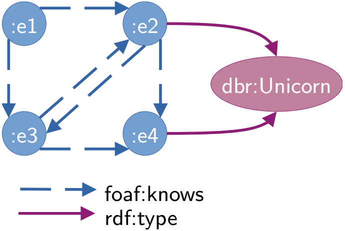


term	*id*(term)
:e1	1
foaf:knows	2
:e2	3
:e3	4
:e4	5
rdf:type	6
dbr:Unicorn	7
unbound	8

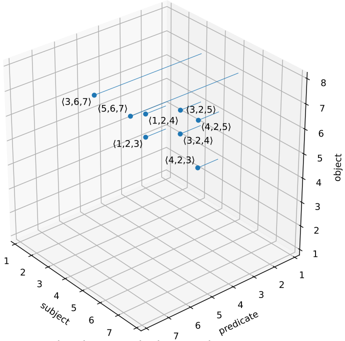
id(s)	*id*(p)	*id*(o)
1	2	3
1	2	4
3	2	4
3	2	5
4	2	3
4	2	5
3	6	7
5	6	7

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 $T : \mathbf{K} = \mathbf{K}_1 \times \cdots \times \mathbf{K}_n \rightarrow V$
  - $\mathbf{K}_1 = \cdots = \mathbf{K}_n \subset \mathbb{N}$
  - $\mathbb{B}$ or $\mathbb{N}$ as co-domain

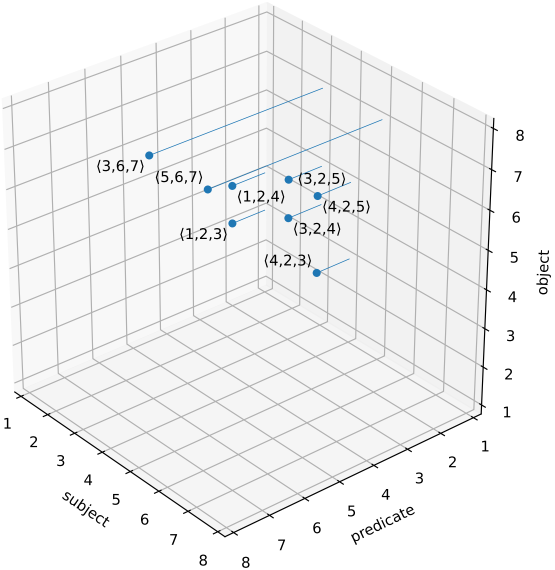
Representing Concepts as SPARQL

TENTRIS: Data Model

- Consider order-n tensors $T : \mathbf{K} = \mathbf{K}_1 \times \cdots \times \mathbf{K}_n \rightarrow V$
  - $\mathbf{K}_1 = \cdots = \mathbf{K}_n \subset \mathbb{N}$
  - $\mathbb{B}$ 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., $T[\mathbf{k}] = v$

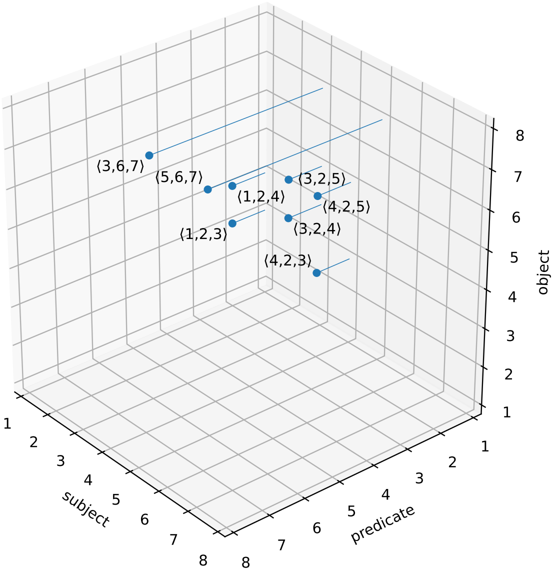
Representing Concepts as SPARQL

TENTRIS: Data Model



Representing Concepts as SPARQL

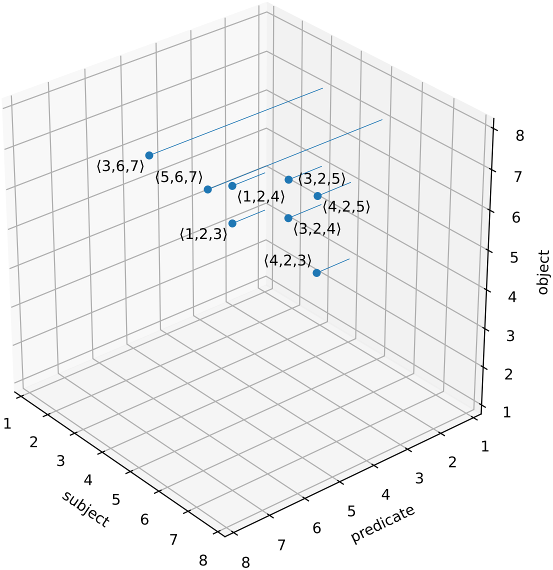
Tentris: Data Model



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

Representing Concepts as SPARQL

Tentris: Data Model

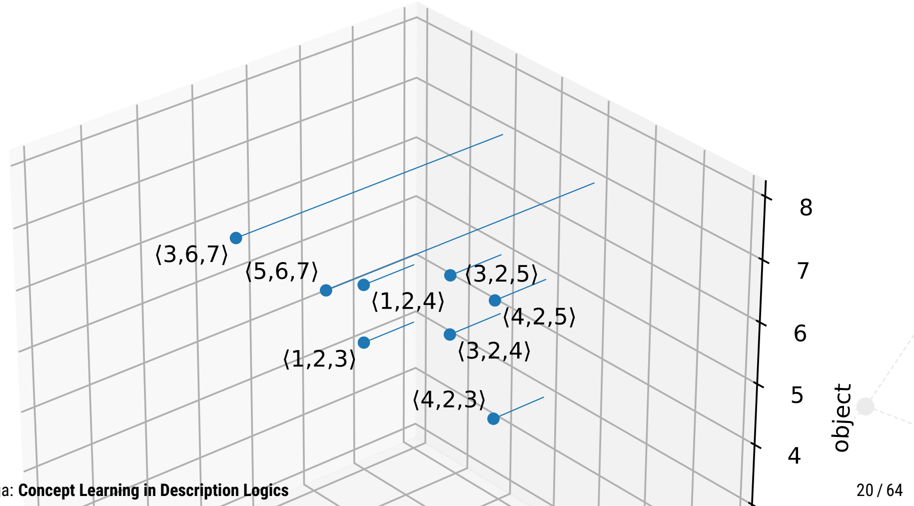


- $\mathbf{K} = \mathbb{N}^3$
- $V = \mathbb{B}$
- $T[\langle 3, 6, 7\rangle] = 1$
- $T[\langle 3, 6, 3\rangle] = 0$

Representing Concepts as SPARQL

TENTRIS: Data Model

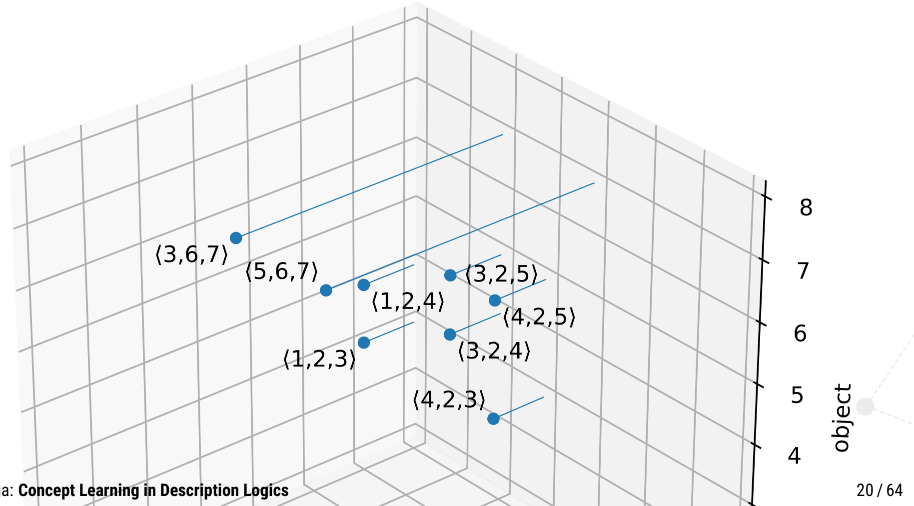
- Slicing selects portion of T, e.g., $T^{(1)} := T[1, 2, :]$ is order-1 tensor



Representing Concepts as SPARQL

Tentris: Data Model

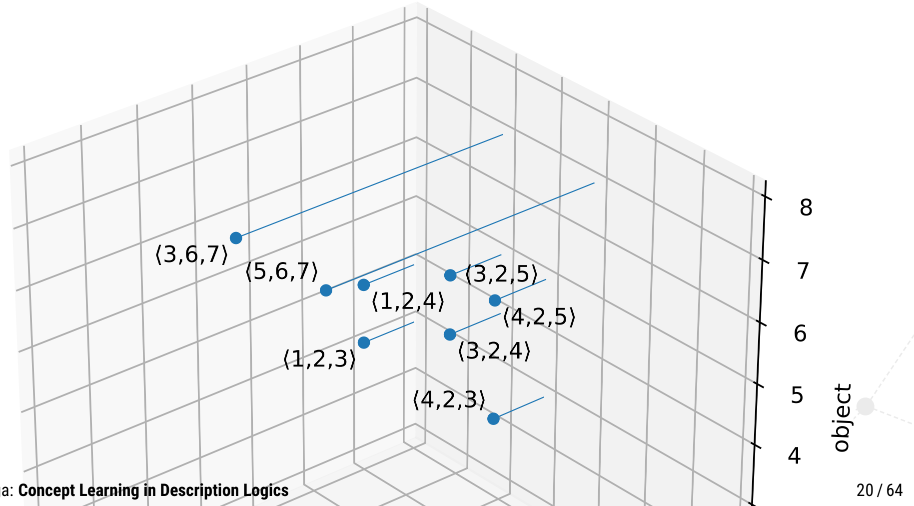
- Slicing selects portion of T, e.g., $T^{(1)} := T[1,2,:]$ is order-1 tensor
- For our example, $T[1,2,:] = [0,0,1,1,0,0,0,0]$



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 [?]



Representing Concepts as SPARQL

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
}
```

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

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

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

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

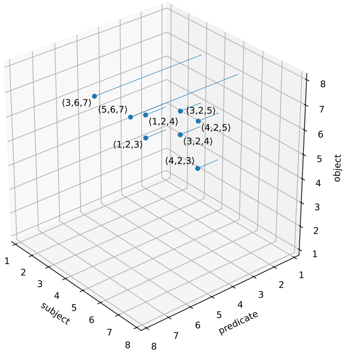
- The projection $\Pi_{U'}(B(g))$ with $U' \subseteq U$ is given by

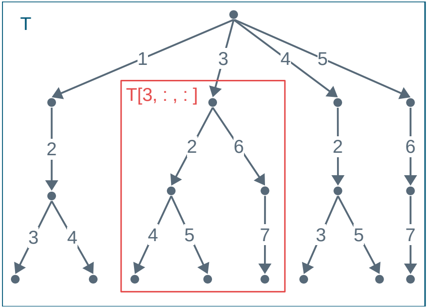
$$T''_{\langle l \in U' \rangle} \leftarrow \bigtimes_i T[\mathbf{k}^{B^{(i)}}]_{\langle l \in B^{(i)} | l \in U \rangle}$$

Representing Concepts as SPARQL

TENTRIS: Hypertrie

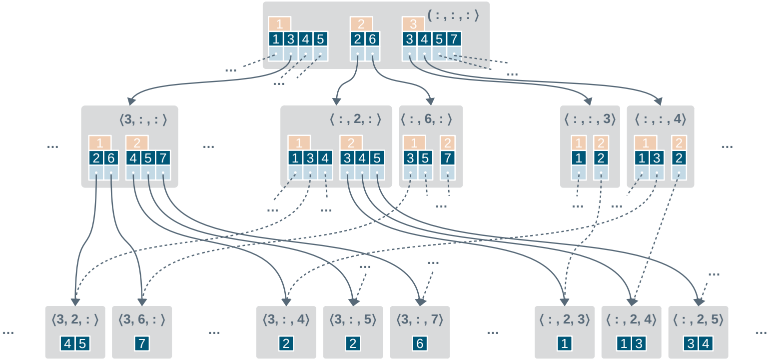
- Query for any tensor slice efficiently
- Allow for efficient querying





Representing Concepts as SPARQL

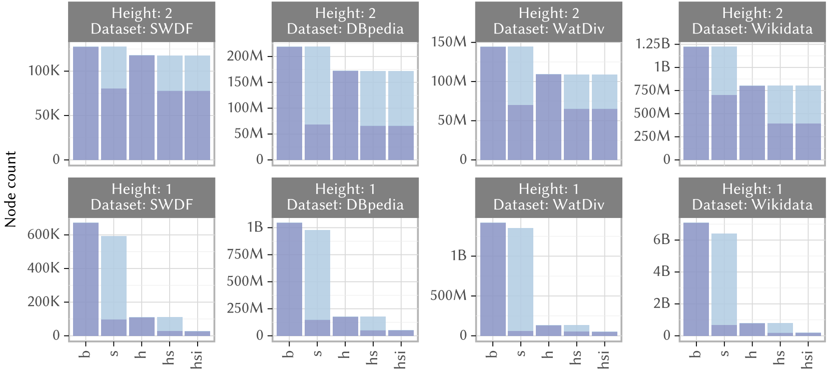
Tentris: Hypertrie



- 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

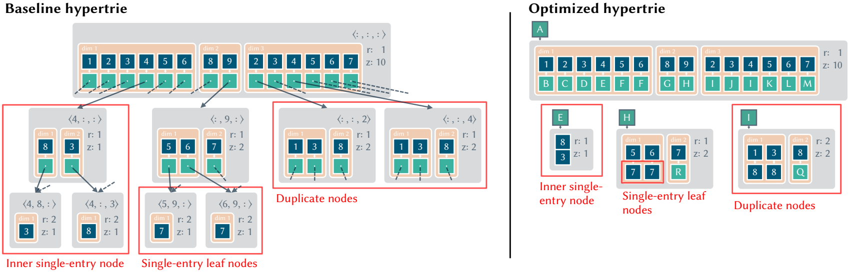
Tentris: Hypertrie



- Hypertrie topology seems sparse
- Compression to improve space, loading and query times [?]

Representing Concepts as SPARQL

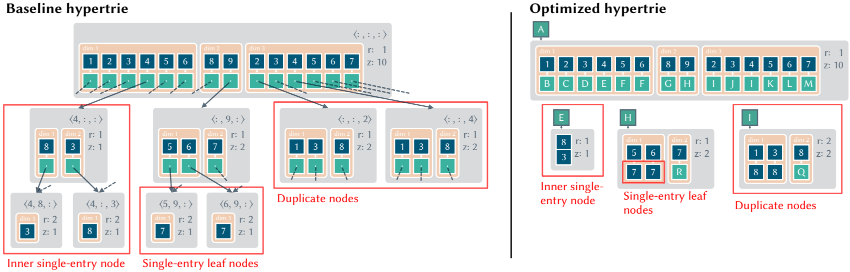
TENTRIS: Compressed Hypertrie



- Compress data based on local and global node topology

Representing Concepts as SPARQL

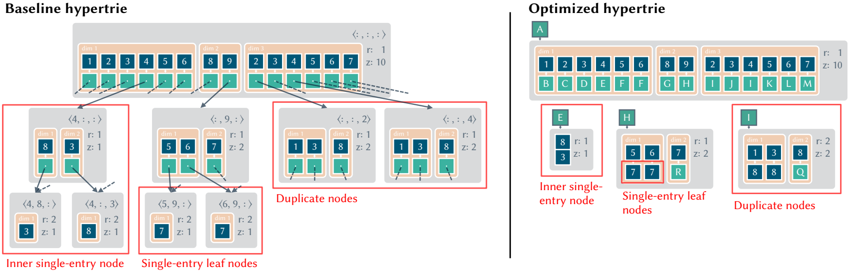
TENTRIS: Compressed Hypertrie



- Compress data based on local and global node topology
- 3 compression approaches
  1. Remove duplicates via hashing (global)

Representing Concepts as SPARQL

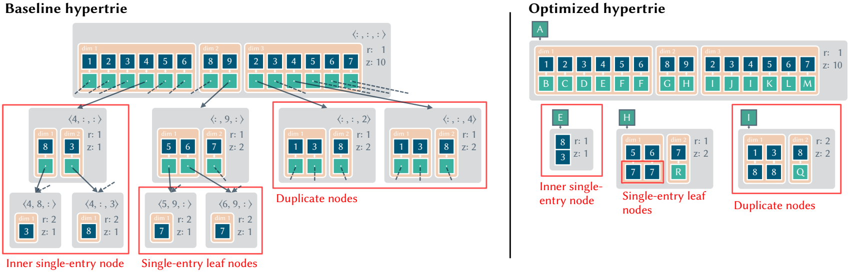
TENTRIS: Compressed Hypertrie



- Compress data based on local and global node topology
- 3 compression approaches
  1. Remove duplicates via hashing (global)
  2. Single-entry inner nodes (local) store sub-hypertries directly

Representing Concepts as SPARQL

TENTRIS: Compressed Hypertrie



- Compress data based on local and global node topology
- 3 compression approaches
  1. Remove duplicates via hashing (global)
  2. Single-entry inner nodes (local) store sub-hypertries directly
  3. Single-entry leaf nodes are eliminated via in-place storage (local)

Representing Concepts as SPARQL

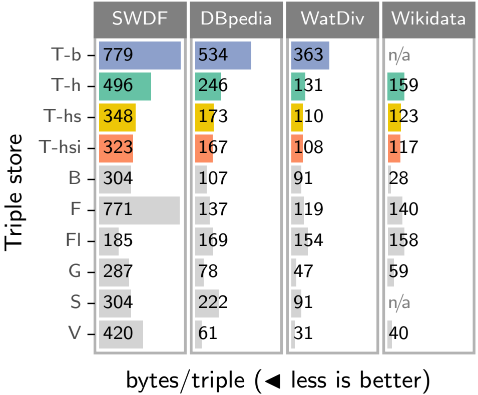
TENTRIS: Compressed Hypertrie

- Comparison with state-of-the-art approaches
- Hardware: AMD EPYC 7742, 1 TB RAM and 2×3 TB NVMe SSDs
- Datasets: Between 372K (SWDF) and 5.5B triples (WikiData)

Representing Concepts as SPARQL

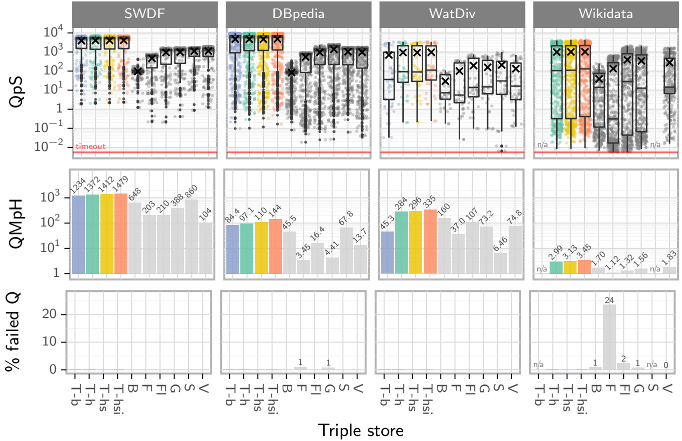
TENTRIS: Compressed Hypertrie

- Comparison with state-of-the-art approaches
- Hardware: AMD EPYC 7742, 1 TB RAM and 2×3 TB NVMe SSDs
- Datasets: Between 372K (SWDF) and 5.5B triples (WikiData)



Representing Concepts as SPARQL

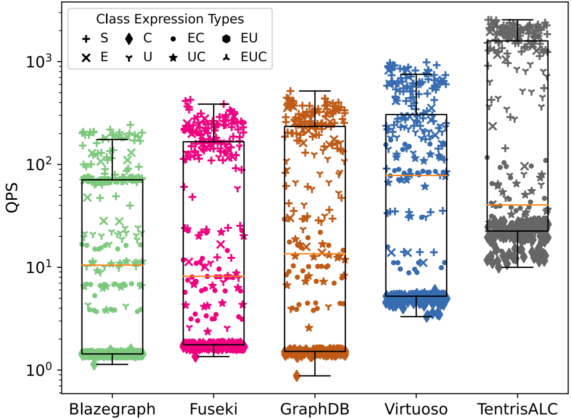
TENTRIS: Compressed Hypertrie



- Better runtimes on all datasets
- Can operate on very large datasets (no time-outs)

Representing Concepts as SPARQL

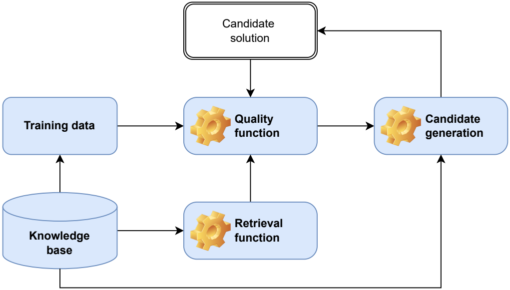
TENTRIS: Carcinogenesis



- Comparison on supervised machine learning tasks in $\mathcal{ALC}$
- Better runtimes on all datasets considered

Learning problem

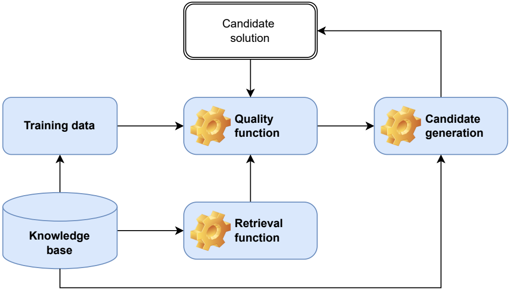
Challenges



- ✓ Retrieval is expensive $\Rightarrow$ Exploit SPARQL
- ▶ Quality functions are often myopic

Learning problem

Challenges



- ✓ Retrieval is expensive $\Rightarrow$ Exploit SPARQL
- ▶ Quality functions are often myopic $\Rightarrow$ Exploit embeddings
- ▶ Candidate generation is expensive $\Rightarrow$ Exploit priming
- ▶ Search space is large $\Rightarrow$ Prune by length

Section 4

Improving Quality Functions

Improving Quality Functions

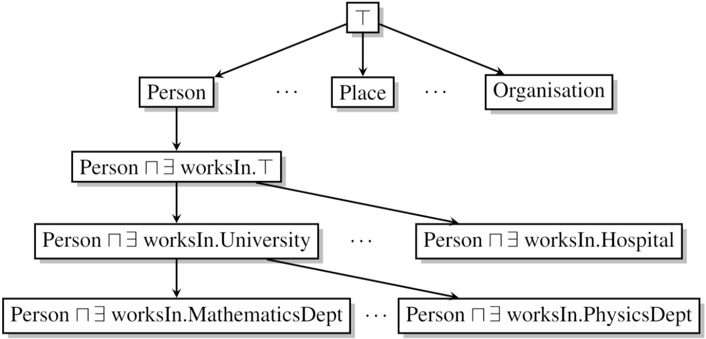
Refinement Operators

- 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:

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

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

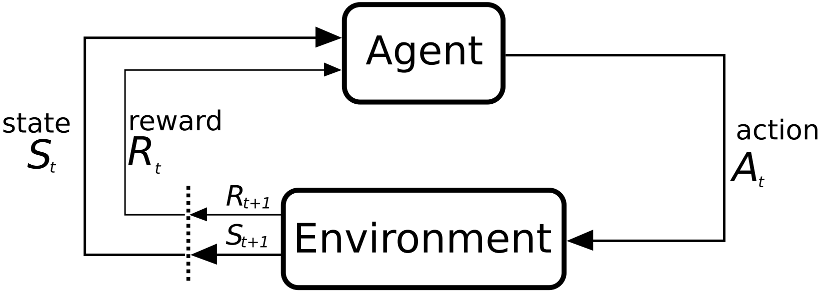
- $\mathrm{score}(C) = \mathrm{acc}_c(C,t) + \alpha \cdot \mathrm{acc\_gain}_c(C) - \beta \cdot |C| \quad (\alpha, \beta \geq 0)$ where typical values are $\alpha = 0.3$ and $\beta = 0.05$.

Problem: Myopia

- Current metrics do not consider future accuracy of concepts
- Optimize for cumulative discounted future rewards **[?]**

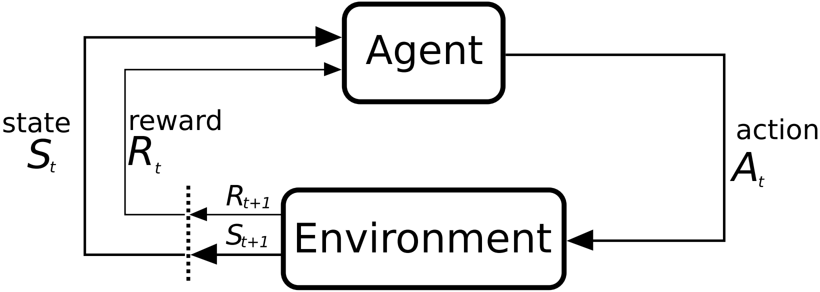
Improving Quality Functions

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 $Q_\pi : S \times A \rightarrow \mathbb{R}$ 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 $Q_\pi : S \times A \rightarrow \mathbb{R}$ 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 [?]

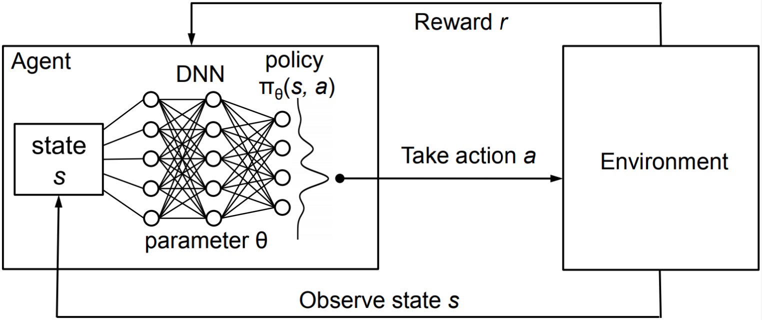
$$\mathcal{L}(\Theta_i) = \mathbb{E}_{(s,a,R,s') \sim U(\mathcal{D})} \left[\left(R + \gamma \max_{\mathbf{a}' \in A(\mathbf{s}')} Q(s', a'; \Theta_i^-) - Q(s, a; \Theta_i) \right)^2 \right]$$

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: [?]

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: Loss function converges to trivial solution

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: Loss function converges to trivial solution
- Solution: Add negative information and margin $\gamma \in \mathbb{R}^+$
- Loss is now

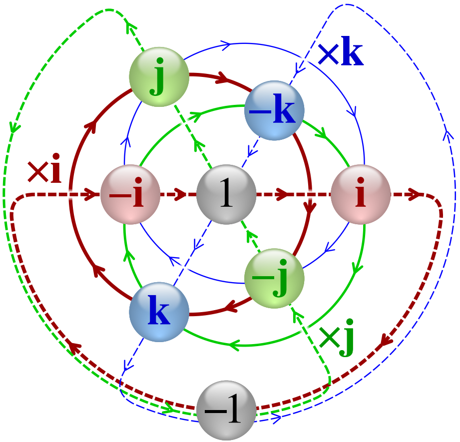
$$L = \sum_{(s,p,o) \in E} \sum_{(s',p,o') \in S'(s,p,o)} [\gamma + d(\vec{s} + \vec{p}, \vec{o}) - d(\vec{s'} + \vec{p}, \vec{o'})]_+$$

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

Quaternions: $\mathbb{H}$



[5]

[5] https://en.wikipedia.org/wiki/Quaternion#/media/File:Cayley_Q8_quaternion_multiplication_graph.svg

Improving Quality Functions

Quaternions: $\mathbb{H}$

- Can define embeddings in this space: QMult [**?**]
  - $\vec{s}, \vec{p}, \vec{o} \in \mathbb{H}^k$
  - Scoring function $\varphi(s, p, o) = (\vec{s} \otimes \vec{p}) \cdot \vec{o}$, where

Improving Quality Functions

Quaternions: $\mathbb{H}$

- Can define embeddings in this space: QMult [**?**]
  - $\vec{s}, \vec{p}, \vec{o} \in \mathbb{H}^k$
  - Scoring function $\varphi(s,p,o) = (\vec{s} \otimes \vec{p}) \cdot \vec{o}$, where
    - $\otimes$ is the Hamiltonian product ($\mathbb{H} \times \mathbb{H} \rightarrow \mathbb{H}$)
    - $\cdot$ is the quaternion inner product ($\mathbb{H} \times \mathbb{H} \rightarrow \mathbb{R}$)

Improving Quality Functions

Quaternions: $\mathbb{H}$

- Can define embeddings in this space: QMult [?]
  - $\vec{s}, \vec{p}, \vec{o} \in \mathbb{H}^k$
  - Scoring function $\varphi(s, p, o) = (\vec{s} \otimes \vec{p}) \cdot \vec{o}$, 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 $$\sum_{(s,p,o)\in\Gamma} \log(1 + exp(-Y_{spo}\varphi(s, p, o)))$$

PADERBORN UNIVERSITY

Improving Quality Functions

DICE

Quaternions: $\mathbb{H}$

- Can define embeddings in this space: QMult [?]
  - $\vec{s}, \vec{p}, \vec{o} \in \mathbb{H}^k$
  - Scoring function $\varphi(s, p, o) = (\vec{s} \otimes \vec{p}) \cdot \vec{o}$, 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
    $$\sum_{(s,p,o)\in\Gamma} \log(1 + exp(-Y_{spo}\varphi(s, p, o)))$$
- Similar construction for octonions

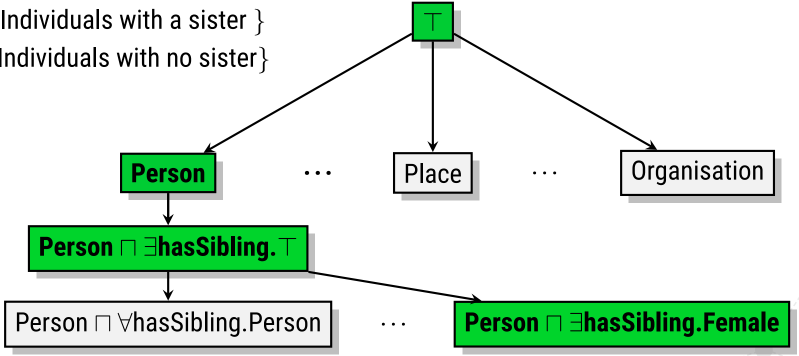
Improving Quality Functions

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

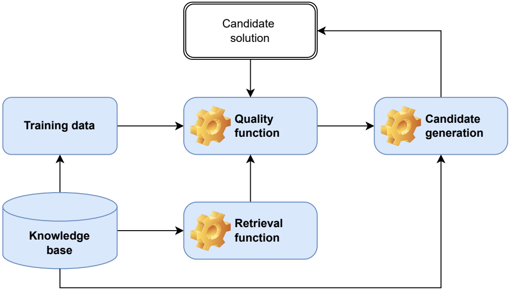
Evaluation

- 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

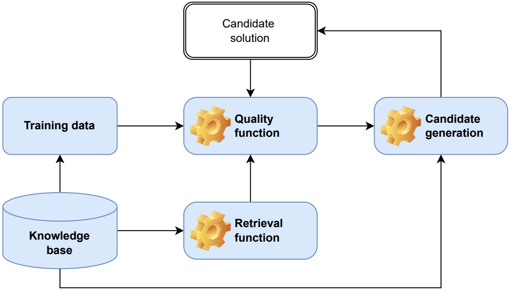
Challenges



✓ Retrieval is expensive

Learning problem

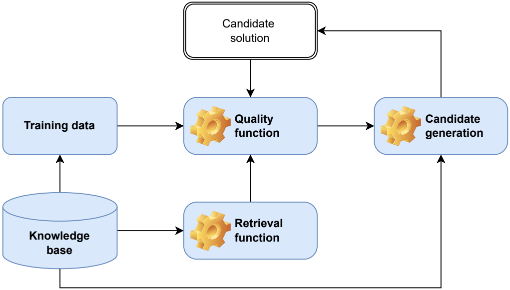
Challenges



- ✓ Retrieval is expensive $\Rightarrow$ Exploit SPARQL
- ✓ Quality functions are often myopic

Learning problem

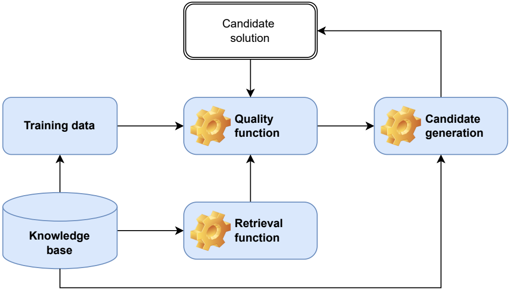
Challenges



- ✓ Retrieval is expensive $\Rightarrow$ Exploit SPARQL
- ✓ Quality functions are often myopic $\Rightarrow$ Exploit embeddings
- ▶ Candidate generation is expensive

Learning problem

Challenges



- ✓ Retrieval is expensive ⇒ Exploit SPARQL
- ✓ Quality functions are often myopic ⇒ Exploit embeddings
- ► Candidate generation is expensive ⇒ Exploit priming
- ► Search space is large ⇒ Prune by length

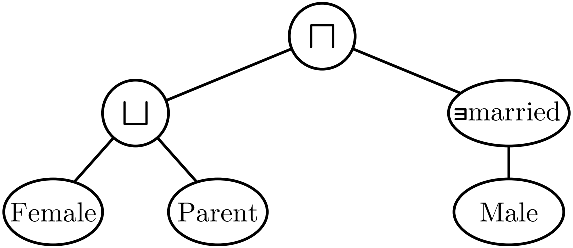
Section 5

Learning with Priming

Learning with Priming

EvoLearner – Idea

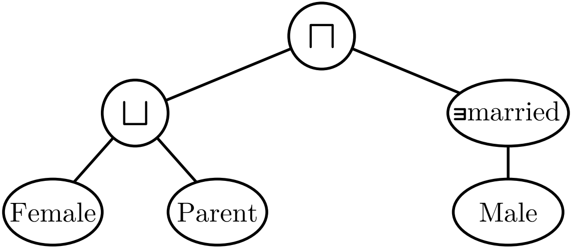
- Represent concepts as trees, e.g.,
  $(\texttt{Female} \sqcup \texttt{Parent}) \sqcap \exists \texttt{married.Male}$



Learning with Priming

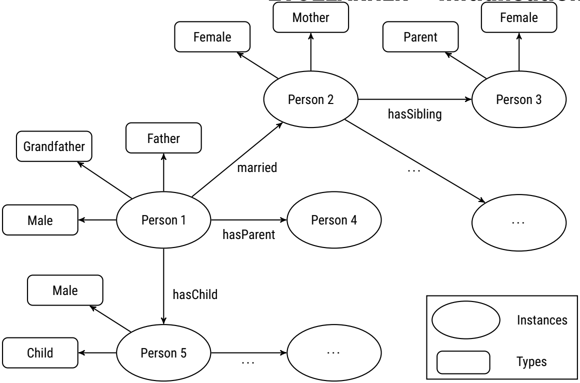
EvoLearner – Idea

- 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 [?]



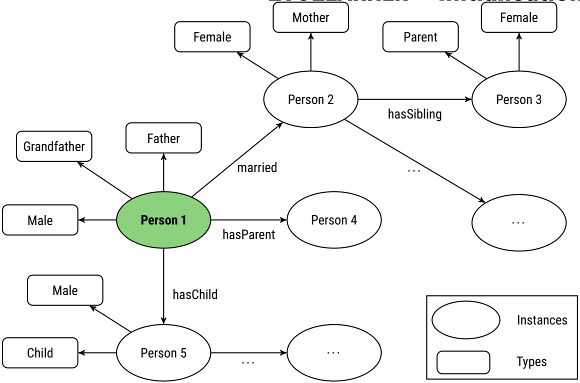
Learning with Priming

EvoLearner – Initialisation



Learning with Priming

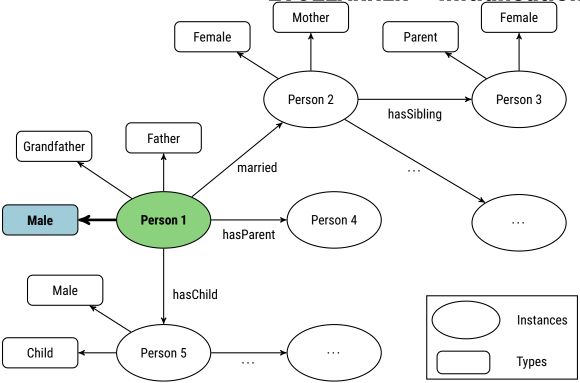
EvoLearner – Initialisation



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

Learning with Priming

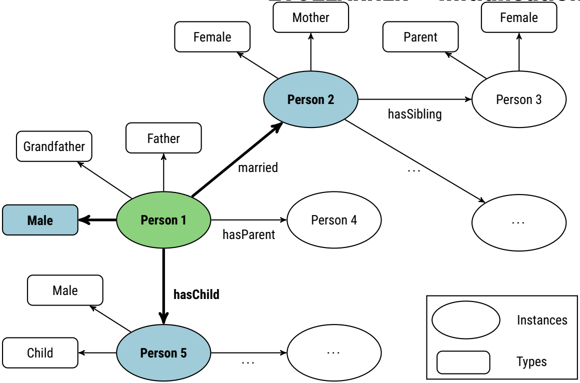
EvoLearner – Initialisation



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

Learning with Priming

EvoLearner – Initialisation

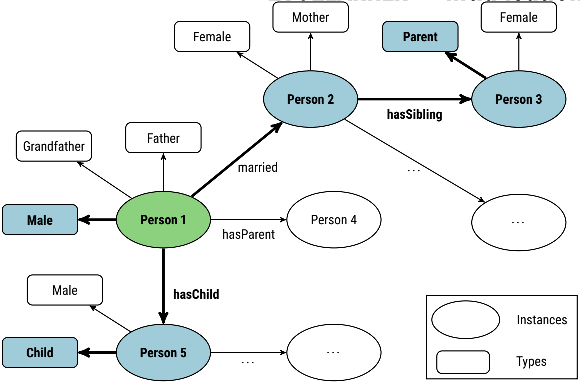


1. Select a positive example e^+ and one of its types: `Male`
2. Randomly select up to *maxT* outgoing triples of e^+:

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

Learning with Priming

EvoLearner – Initialisation



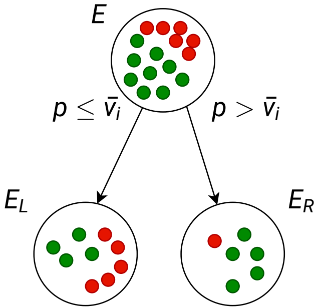
1. Select a positive example e^+ and one of its types: `Male`
2. Randomly select up to *maxT* outgoing triples of e^+:
   $\texttt{Male} \sqcap (\exists\texttt{married} \ldots \sqcap \exists\texttt{hasChild} \ldots)$
3. Complete incomplete subconcepts:
   $\texttt{Male} \sqcap ((\exists\texttt{married}.\exists\texttt{hasSibling}.\texttt{Parent})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

Learning with Priming

EvoLearner – Data Properties

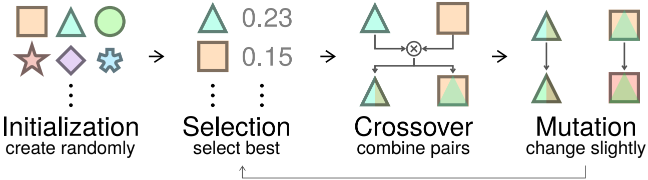
- Given a data property d from the knowledge base $\mathcal{K}$ and a set E of positive and negative examples
- We precompute up to k splits of the form $d \leq \bar{v}_i$ per data property
- Splits are computed to maximize information gain:

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



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

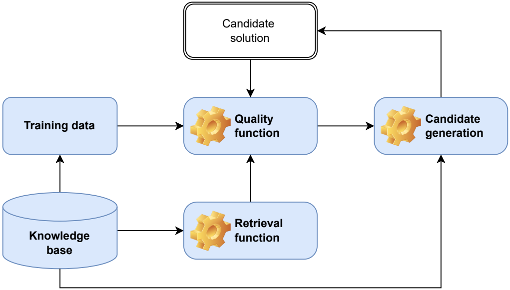
Learning with Priming

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

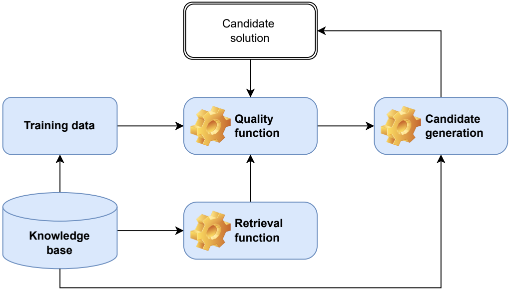
Challenges



- ✓ Retrieval is expensive $\Rightarrow$ Exploit SPARQL
- ✓ Quality functions are often myopic

Learning problem

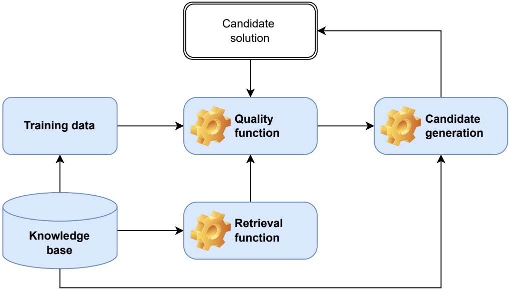
Challenges



- ✓ Retrieval is expensive $\Rightarrow$ Exploit SPARQL
- ✓ Quality functions are often myopic $\Rightarrow$ Exploit embeddings
- ✓ Candidate generation is expensive

Learning problem

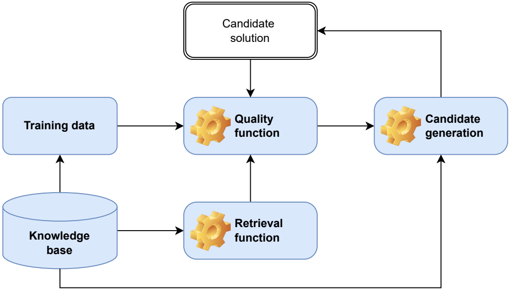
Challenges



- ✓ Retrieval is expensive $\Rightarrow$ Exploit SPARQL
- ✓ Quality functions are often myopic $\Rightarrow$ Exploit embeddings
- ✓ Candidate generation is expensive $\Rightarrow$ Exploit priming

Learning problem

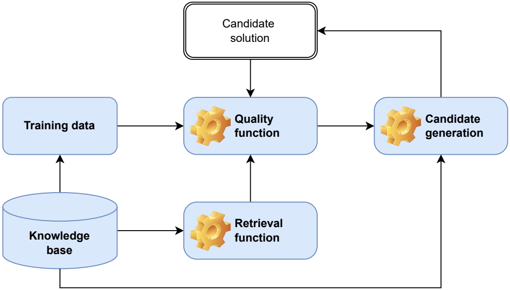
Challenges



- ✓ Retrieval is expensive ⇒ Exploit SPARQL
- ✓ Quality functions are often myopic ⇒ Exploit embeddings
- ✓ Candidate generation is expensive ⇒ Exploit priming
- ▶ Search space is large

Learning problem

Challenges



- ✓ Retrieval is expensive $\Rightarrow$ Exploit SPARQL
- ✓ Quality functions are often myopic $\Rightarrow$ Exploit embeddings
- ✓ Candidate generation is expensive $\Rightarrow$ Exploit priming
- ► Search space is large $\Rightarrow$ Prune by length

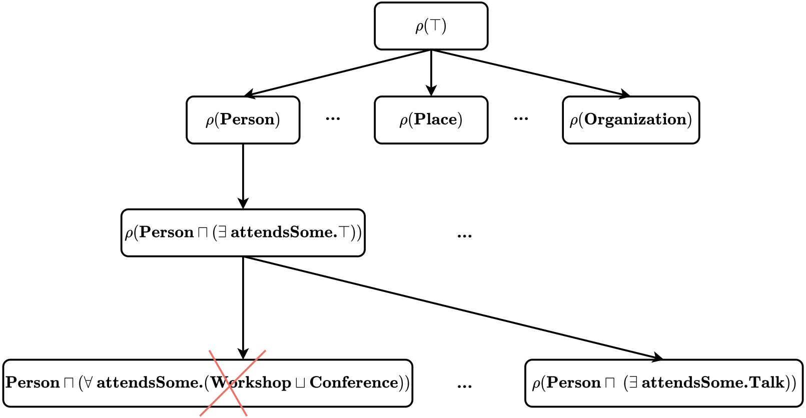
Section 6

CLIP

CLIP

Approach

- Idea: Prune horizontally by
- predicting target concept length and
- discarding longer refinements



CLIP

Concept Lengths

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

CLIP

Concept Lengths

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

Concept Lengths

- $length(A) = length(\top) = length(\bot) = 1$ (if A is an 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

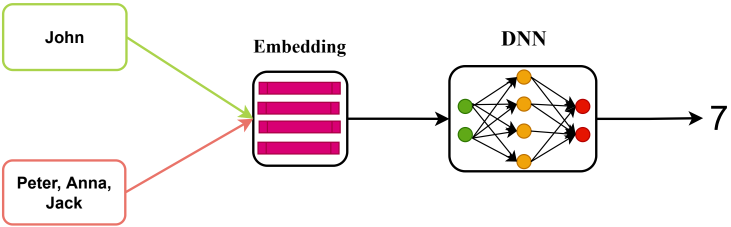
CLIP

Concept Lengths

- $length(A) = length(\top) = length(\bot) = 1$ (if A is an 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

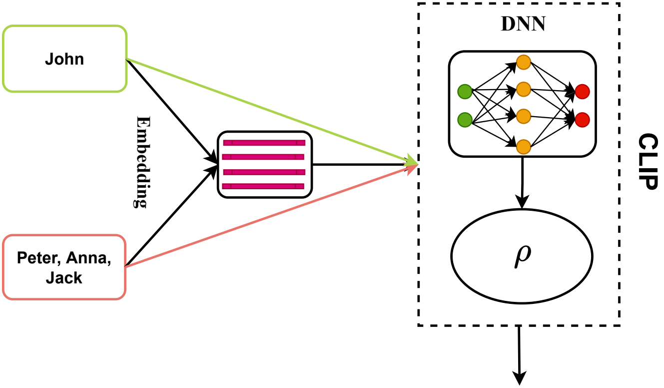
Concept Length Prediction



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

CLIP

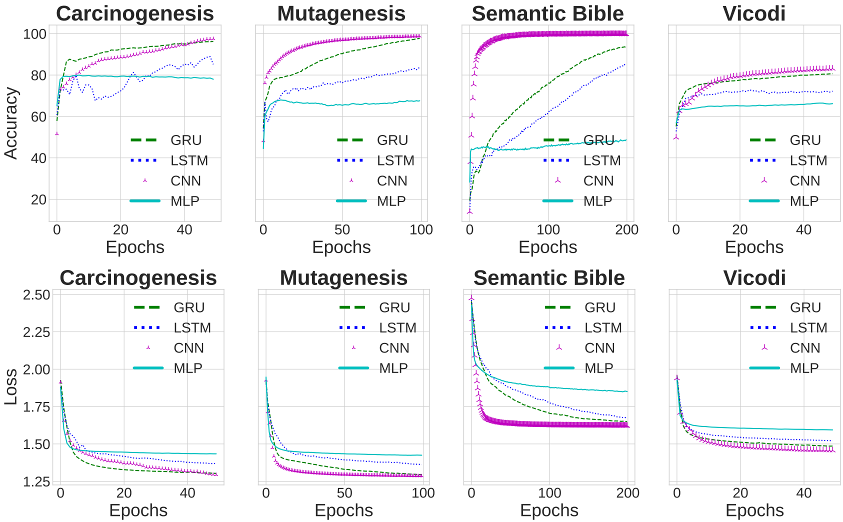
Concept Learning



Male ⊓ ∃ hasParent.(∃ hasChild.Female)

CLIP

Training

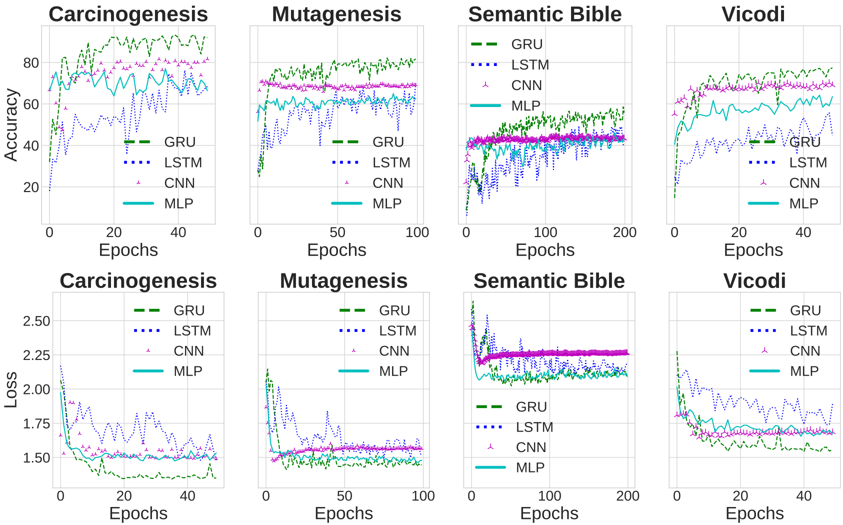


PADERBORN UNIVERSITY

CLIP

DICE

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

Metric	CELOE	OCEL	ELTL	CLIP
Carcinogenesis				
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				
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				
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				
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 7

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?

Summary

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