Finding Semantic-preserved Representation of Knowledge in Description Logic Ontologies: Preliminary Results in RiceDO and TreatO

Phuriwat Angkoondittaphong
Under supervision of ReaLearn Lab

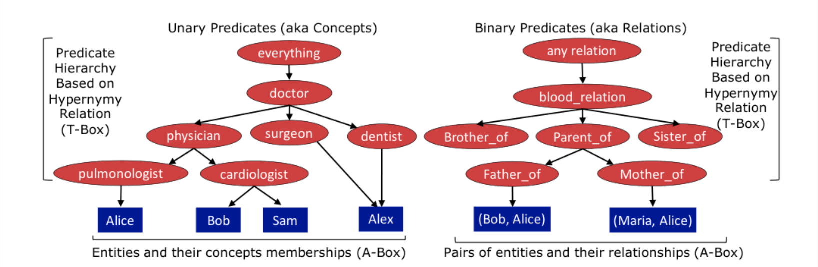
13 Sep 2024

Quantum Embedding of Knowledge for Reasoning

Quantum Embedding of Knowledge for Reasoning

- The core idea of this paper is to represent description logic ($\mathcal{ALC}$) with complex vector space.
- Namely $\Sigma = \mathbb{C}^d$ where $d \in \mathbb{N}$ is embedding size.
- TBox is subspace of $\mathbb{C}^d$ and ABox is vector in the $\mathbb{C}^d$.

Example of description logic



Example of representing Unary ABox in $\mathbb{C}^d$

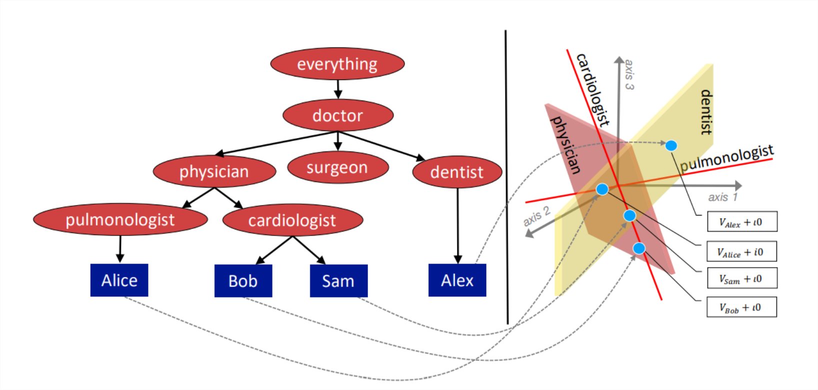


Figure 1: How would unary A-Box represent in $\mathbb{C}^3$

Simply just give each entity one vector with only a real part.

Example of Representing Binary ABox in $\mathbb{C}^d$

- Given Alice ($V_{Alice} + i0$) and Bob ($V_{Bob} + i0$) form a binary relation (Alice, Bob)
- Then the vector represents the relation between them is $V_{Alice} + iV_{Bob}$

Problem formulation

- For implementation, the authors map from $\mathbb{C}^d$ to $\mathbb{R}^{2d}$ for make thing easy to calculate.
- They make the real part mapped on indicies 1 to d and the imaginary part of the vector becomes indices d+1 to 2d of the vector.

$$V_{Bob} + iV_{Alice} \rightarrow [\overbrace{V_{Bob}^{(1)}, V_{Bob}^{(2)}, \ldots, V_{Bob}^{(d)}}^{\text{indices } 1 \text{ to } d}, \underbrace{V_{Alice}^{(1)}, V_{Alice}^{(2)}, \ldots, V_{Alice}^{(d)}}_{\text{indices } d+1 \text{ to } 2d}]^T$$

Here's an example of how to map a pair (Bob, Alice) from $\mathbb{C}^d \rightarrow \mathbb{R}^{2d}$

Problem formulation (cont.)

- There are 3 things that we need to make them learn
  - embedding of the entities $x_i \in \mathbb{R}^d \rightarrow O_i \in \mathcal{N}_O$
  - embedding of the concepts $y_i \in \{0, 1\}^d \rightarrow C_i \in \mathcal{N}_C$
  - embedding of the relations $z_i \in \{0, 1\}^{2d} \rightarrow R_i \in \mathcal{N}_R$

Loss terms proposed in QE paper

- Loss: Assertion, Assert predefined rules. eg. unit length

$$L_{O_i} = (1 - x_i^\top x_i)^2, L_{C_i} = ||y_i \odot \bar{y}_i||^2, L_{R_i} = ||z_i \odot \bar{z}_i||^2$$

- Loss: Membership, Assert membership of entities

$$L_{O_i \in C_j} = ||\bar{y}_j \odot x_i||^2, L_{(O_p, O_q) \in R_k} = ||\bar{z}_k \odot x_{pq}||^2$$

- Loss: Logical Inclusion, $\sqsubseteq$ operator

$$L_{C_i \sqsubseteq C_j} = ||y_i \odot \bar{y}_j||^2, L_{R_i \sqsubseteq R_j} = ||z_i \odot \bar{z}_j||^2$$

Loss terms proposed in QE paper (cont.)

- Loss: Logical Conjunction, $\wedge$ operator

$$L_{C_i=C_j\sqcap C_k} = ||y_i - (y_j \odot y_k)||^2; L_{R_i=R_j\sqcap R_k} = ||z_i - (z_j \odot z_k)||^2$$

- Loss: Logical Disjunction, $\vee$ operator

$$L_{C_i=C_j\sqcup C_k} = ||y_i - \max(y_j, y_k)||^2; L_{R_i=R_j\sqcup R_k} = ||z_i - \max(z_j, z_k)||^2$$

- Loss: Negation, $\neg$ operator

$$L_{C_i=\neg C_j} = (y_i^\top y_j)^2 + (\bar{y}_i^\top \bar{y}_j)^2; L_{R_i=\neg R_j} = (z_i^\top z_j)^2 + (\bar{z}_i^\top \bar{z}_j)^2$$

- Loss: Universal Type Restriction, $\forall$ operator

$$L_{\forall R_i \cdot C_j}(y_k) = (y_k^\top \left(\begin{smallmatrix} 0_d I_d \\ 0_d 0_d \end{smallmatrix} \right) z_i)^2$$

Overall Loss

Overall loss is calculated by adding all loss terms together.

$$\min_{x_i, y_j, z_k} L$$

Minimize loss L with respect to x, y, z

QE Experimental Setup

- Training setup
  - ADAM Optimizer with learning rate 10^{-3}
  - E2R Loss
  - Embedding size $= 100$
- Dataset
  - RiceDO
  - TreatO

Datasets

- RiceDO is Rice disease ontology, containing relationships between disease, symptoms, and causation.
- TreatO is ontology on how to cure rice disease.
- Both ontologies are written using only Existential Language ($\mathcal{EL}$) and only contain TBox.
- QE uses $\mathcal{ALC}$ and necessary to have ABox[1], which makes us need to drop some loss terms.

[1] This is probably my misunderstanding the original QE code

Reduced Loss Terms

Because RiceDO and TreatO are written in $\mathcal{EL}$ and due to how we map the ontology to triples, the loss terms in gray are removed from equation.

- Loss: Assertion, Assert predefined rules. eg. unit length
- Loss: Membership, Assert membership of entities
- Loss: Logical Inclusion, $\sqsubseteq$ operator
- Loss: Logical Conjunction, $\wedge$ operator
- Loss: Logical Disjunction, $\vee$ operator
- Loss: Negation, $\neg$ operator
- Loss: Universal Type Restriction, $\forall$ operator

Example of mapping OWL to Triples

The ontology

```
<SubClassOf>
    <Class IRI="http://purl.obolibrary.org/obo/PDO_0000059"/>
    <ObjectSomeValuesFrom>
        <ObjectProperty IRI="#RiceDO_000147"/>
        <ObjectIntersectionOf>
            <ObjectSomeValuesFrom>
                <ObjectProperty IRI="#RiceDO_000155"/>
                <Class IRI="#RiceDO_000108"/>
            </ObjectSomeValuesFrom>
            <ObjectSomeValuesFrom>
                <ObjectProperty IRI="#RiceDO_000156"/>
                <ObjectIntersectionOf>
                    <Class IRI="#RiceDO_000023"/>
                    <Class IRI="#RiceDO_000135"/>
                </ObjectIntersectionOf>
            </ObjectSomeValuesFrom>
        </ObjectIntersectionOf>
    </ObjectSomeValuesFrom>
</SubClassOf>
```

Figure 2: Example of a restriction that cannot represent with one triple.

$$\text{PDO059} \sqsubseteq (\exists \text{RD147}.(\exists \text{RD155}.\text{RD108} \sqcap \exists \text{RD156}.(\text{RD023} \sqcap \text{RD135})))$$

$\rightarrow$

The generated triples

- (PDO059, subClassOf, N1)
- (N1, onProperty, RD147)
- (N1, someValuesFrom, N2)
- (N2, intersectionOf, N3)
- (N3, rest, N4)
- (N3, first, N5)
- (N5, someValuesFrom, RD108)
- (N5, onProperty, RD155)
- ...

Use alias to represent a group of things.

Existing Techniques

All of the existing techniques mentioned here use pykeen's implementation, and I also leave all hyperparameters to default values in pykeen.

- TransE
- ComplEx
- TransH
- DisMult
- ProjE

Link prediction

- The task that we tackling is link prediction.
- Given a pair of missing head or tail binary A-box, $(H, R, ?)$ or $(?, R, T)$.
- The model is expected to find the missing entity (?).
- QE should perform better than others on this task as it **incorporates logical structure** into the embedding creation process.
- Other techniques only tried to minimize the distance between paired entities without considering any logical structure.

Evaluation Metrics of Link Prediction

- Mean Rank ($\downarrow$)
  - The average rank of the correct entity.
- Hits@1 ($\uparrow$)
  - The percentage of the correct entity got rank 1.
- Hits@10 ($\uparrow$)
  - The percentage of the correct entity got rank 10 or higher.

RiceDO Results Numerically

Techniques	**MR** ($\downarrow$)	**H@1** ($\uparrow$)	**H@10** ($\uparrow$)
QE	425.14	23.30	25.80
TransE	350.62	4.12	22.24
ComplEx	971.12	0.12	0.47
TransH	**50.53**	**32.82**	**49.53**
DistMult	251.64	26.12	43.18
ProjE	512.61	4.82	22.71

Table 1: RiceDO Results metrics. The best number for each metrics is written in **bold** font.

TreatO Results Numerically

Techniques	**MR** (↓)	**H@1** (↑)	**H@10** (↑)
QE	50.78	15.85	16.71
TransE	55.61	9.73	41.15
ComplEx	281.62	0.00	1.33
TransH	**10.86**	**48.23**	**76.11**
DistMult	20.74	42.92	69.91
ProjE	113.27	15.93	35.40

Table 2: TreatO Results metrics. The best number for each metric is written in **bold** font.

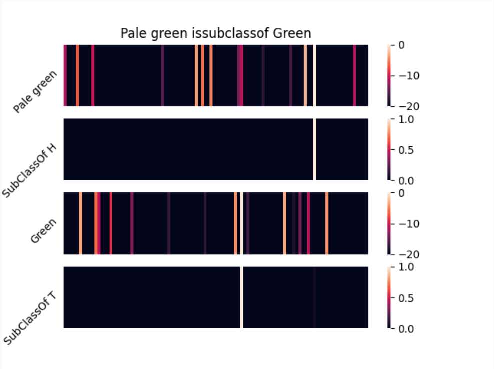
Looking Back

Why are the results of QE worse than those of other techniques?

- The way I convert from ontology to triples convert all of TBox to ABox.
- Which eliminates the logical structure in the ontology.

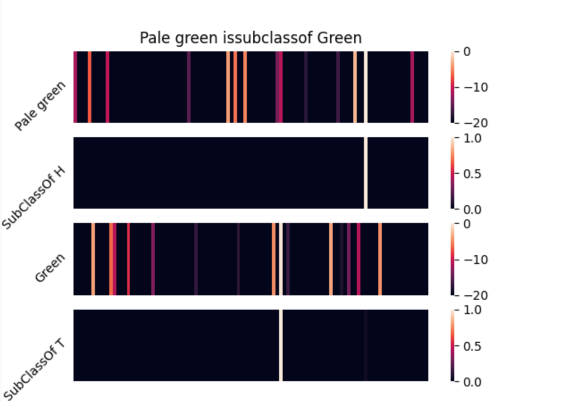
What to expect from QE embeddings

- The head entity embedding (top row) should have a non-zero value on the same indices as relation embedding for the head (upper middle row).
- The same applies to the tail entity embedding (lower middle row) and relation embedding for the tail (bottom row).



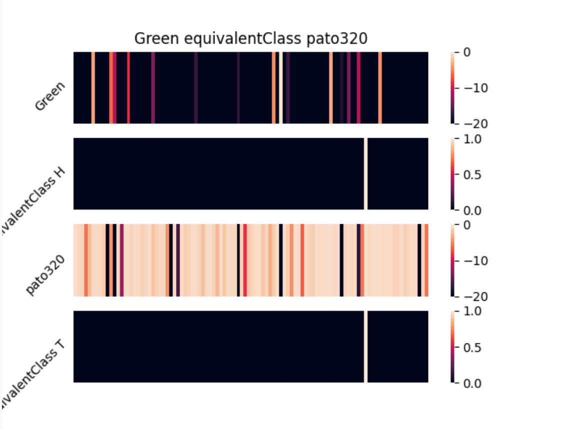
Observations of result QE embeddings

- Green and Pale green are entities, and isSubClassOf is binary relation in the ontology.
- Pale green isSubClassOf Green is a fact stated in RiceDO
- The entity embeddings should have non-zero value at the index where their relation is non-zeros.

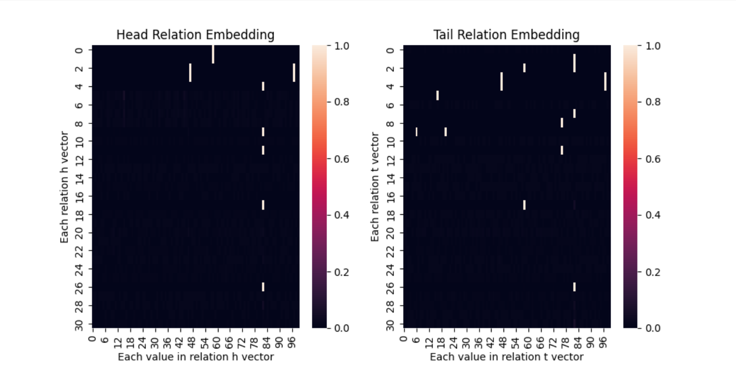


Observations of result QE embeddings

- However, this is not always the case.



Subspace collapses



- There are some of relation embedding, that use the same subspace, which is example of subspace collapse.
- The original paper solved this problem by adding regularization terms.

Potential Future works

- Solve Subspace Collapses
- Employ Abduction
  - Generate minimal sets of ABox axioms as a training data.

Towards Practical ABox Abduction in Large OWL DL Ontologies

Jianfeng Du
Guangdong University of Foreign Studies, Guangzhou 510006, China
State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences

Guilin Qi
School of Computer Science and Engineering, Southeast University, NanJing 211189, China
State Key Laboratory for Novel Software Technology, Nanjing University, China

Yi-Dong Shen
State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China

Jeff Z. Pan
Department of Computing Science, The University of Aberdeen, Aberdeen AB243UE, UK

Abstract

ABox abduction is an important aspect for abductive reasoning in Description Logics (DLs). It finds all minimal sets of ABox axioms that should be added to a background ontology to enforce entailment of a specified set of ABox axioms. As far as we know, by now there is only one ABox abduction method in expressive DLs computing abductive solutions with certain minimality. However, the method targets an ABox abduction problem that may have infinitely many abductive solutions and may not output an abductive solution in finite time. Hence, in this paper we propose a new ABox abduction problem which has only finitely many abductive solutions and also propose a novel method to solve it. The method reduces the original problem to an abduction problem in logic programming and solves it with Prolog engines. Experimental results show that the method is able to compute abductive solutions in benchmark OWL DL ontologies with large ABoxes.

As far as we know, by now there is only one ABox abduction method (Klarman, Endriss, and Schlobach 2011) in expressive DLs computing abductive solutions with certain minimality. The method works on the DL $\mathcal{ALC}$, which is a fragment of OWL DL — a species of the standard OWL corresponding to the DL $\mathcal{SHOIN}$(D) (Horrocks, Patel-Schneider, and van Harmelen 2003). The problem addressed in the method allows abductive solutions to be expressed in the $\mathcal{ALE}$ fragment of $\mathcal{ALC}$, e.g., allows existential restrictions in solutions; thus the problem may have infinitely many solutions. Consider an ontology containing only the following axiom, which says that something has a person as its parent is a person.

$$\exists \mathsf{hasParent}.\mathsf{Person} \sqsubseteq \mathsf{Person}$$

There are infinitely many abductive solutions for the observation $\{\mathsf{Person}(\mathsf{Amy})\}$ (i.e. Amy is a person). Each abductive solution consists of a concept assertion of the form $\exists \mathsf{hasParent}.\exists \mathsf{hasParent}.\ldots.\mathsf{Person}(\mathsf{Amy})$, in which the in-