Encore Abstract: Query Embedding on Hyper-relational Knowledge Graphs*

Dimitrios Alivanistos[1,4], Max Berrendorf[2], Michael Cochez[1,4], and and Mikhail Galkin[3]

[1] Vrije Universiteit Amsterdam {d.alivanistos,m.cochez}@vu.nl
[2] LMU Munich berrendorf@dbs.ifi.lmu.de
[3] Mila, McGill University mikhail.galkin@mila.quebec
[4] Discovery Lab, Elsevier

Graph query answering presents itself as a prevailing field of research following the recent popularity of graph databases and specifically of knowledge graphs (KGs). A major task in graph query answering is having the ability to answer queries with incomplete knowledge, i.e. missing links in the graph. This is called *approximate query answering* or in some settings **multi-hop logical reasoning**.

Multi-hop logical reasoning subsumes both one-hop link prediction as well as other more complex types of logical queries. However, existing algorithms operate only on classical, triple-based graphs, whereas modern KGs often employ a **hyper-relational** modeling paradigm. In this paradigm, typed edges may have several key-value pairs known as *qualifiers* that provide fine-grained context for facts. Hyper-relational queries are often used in real-world KG applications. Also queries can contain qualifiers. This context modifies the meaning of relations, and usually reduces the answer set. However, existing approaches for approximate query answering (QA) cannot make use of qualifier pairs. In this work, we bridge this gap and extend the multi-hop reasoning problem to hyper-relational KGs allowing to tackle this new type of complex queries.

Query embedding(QE) on knowledge graphs (KGs) aims to answer logical queries using neural reasoners instead of traditional databases and query languages. QE bypasses the need for a database or query engine and performs reasoning directly in a latent space by computing a similarity score between the *query representation* and *entity representations*.

StarQE, our model, is not trained directly on the graph, but using a dataset of pre-generated hyper-relational queries and their answers [5]. It combines the message-passing paradigm for query modelling, alongside several StarE layers that enable the inclusion of qualifier information in the learning process. To model queries, we introduce embeddings for query variables and targets and learn them alongside entity and relation representations.

In our **experiments** we use a rank-based evaluation setting and compare performance against well-designed baselines such as *triple-only* where we omit

* This is an abstract of the paper presented earlier at ICLR 2022 https://openreview.net/forum?id=4rLw09TgRw9

[5] For more information on the WD50K-QE dataset, we urge the readers to look into the original paper.

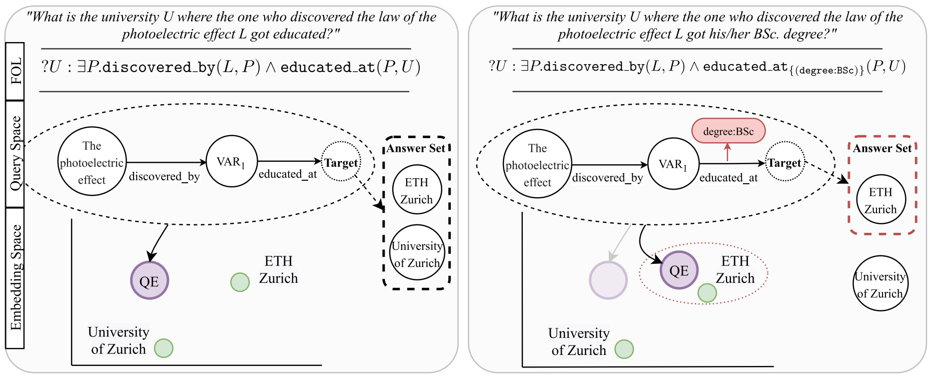
"What is the university U where the one who discovered the law of the photoelectric effect L got educated?"
$?U : \exists P.\texttt{discovered\_by}(L, P) \land \texttt{educated\_at}(P, U)$
FOL
Query Space
Embedding Space
The photoelectric effect
discovered_by
VAR1
educated_at
Target
Answer Set
ETH Zurich
University of Zurich
QE
"What is the university U where the one who discovered the law of the photoelectric effect L got his/her BSc. degree?"
$?U : \exists P.\texttt{discovered\_by}(L, P) \land \texttt{educated\_at}_{\{(\texttt{degree:BSc})\}}(P, U)$
degree:BSc

Fig. 1. Triple-based (left) and hyper-relational (right) queries. The answer set of the hyper-relational query is reduced with the addition of a qualifier pair, and the final query representation moves closer to the narrowed down answer.

the qualifier information, the *reification* where we compare performance over different representations of the hyper-relational query, *zero-layers* where we leave out the message-passing component and the *oracle* which compares to a perfect *triple-only* baseline. We provide with results in Table 1. Very extensive experimentation and results can be found in the original paper.

In **conclusion**, we find that StarQE is well able to produce the correct results for hyper-relational query answering and also perform well over reified queries, showing robustness. For future work, we plan to extend our approach to handle more logical operators. Last but not least, more research is needed to understand in which use cases each query representation works best.

Table 1. QA performance of StarQE and the baselines when training on all hyper-relational query patterns. We omit the `hr-` prefix for brevity. Best results (excluding the Oracle) are marked in **bold**.

Pattern	`1p`	`2p`	`3p`	`2i`	`3i`	`2i-1p`	`1p-2i`
				Hits@10 (%)			
StarQE	51.72	**51.20**	**65.50**	77.78	92.64	**61.81**	**81.60**
Triple-Only	45.04	12.76	24.66	69.74	91.74	16.77	40.67
Reification	**55.17**	50.86	63.65	**81.25**	**95.31**	61.05	80.49
Zero Layers	44.93	29.94	38.45	67.79	90.66	35.48	47.85
Oracle	81.03	24.11	38.47	95.54	99.67	32.74	76.96