Adapting Neural Link Predictors for Data-Efficient Complex Query Answering

Arakelyan, Erik; Minervini, Pasquale; Daza, Daniel; Cochez, Michael; Augenstein, Isabelle

Published in:

Publication date:
2024

Document version
Publisher’s PDF, also known as Version of record

Document license:
CC BY

Citation for published version (APA):
Adapting Neural Link Predictors for Data-Efficient Complex Query Answering

Erik Arakelyan∗1 Pasquale Minervini†2 Daniel Daza3,4,5
Michael Cochez3,5 Isabelle Augenstein1
1University of Copenhagen 2University of Edinburgh 3Vrije Universiteit Amsterdam
4University of Amsterdam 5Discovery Lab, Elsevier, The Netherlands
{erik.a,augenstein}@di.ku.dk p.minervini@ed.ac.uk {d.dazacruz,m.cochez}@vu.nl

Abstract

Answering complex queries on incomplete knowledge graphs is a challenging task where a model needs to answer complex logical queries in the presence of missing knowledge. Prior work in the literature has proposed to address this problem by designing architectures trained end-to-end for the complex query answering task with a reasoning process that is hard to interpret while requiring data and resource-intensive training. Other lines of research have proposed re-using simple neural link predictors to answer complex queries, reducing the amount of training data by orders of magnitude while providing interpretable answers. The neural link predictor used in such approaches is not explicitly optimised for the complex query answering task, implying that its scores are not calibrated to interact together. We propose to address these problems via CQD A, a parameter-efficient score adaptation model optimised to re-calibrate neural link prediction scores for the complex query answering task. While the neural link predictor is frozen, the adaptation component – which only increases the number of model parameters by 0.03% – is trained on the downstream complex query answering task. Furthermore, the calibration component enables us to support reasoning over queries that include atomic negations, which was previously impossible with link predictors. In our experiments, CQD A produces significantly more accurate results than current state-of-the-art methods, improving from 34.4 to 35.1 Mean Reciprocal Rank values averaged across all datasets and query types while using ≤ 30% of the available training query types. We further show that CQD A is data-efficient, achieving competitive results with only 1% of the complex training queries, and robust in out-of-domain evaluations. Source code and datasets are available at https://github.com/EdinburghNLP/adaptive-cqd.

1 Introduction

A Knowledge Graph (KG) is a knowledge base representing the relationships between entities in a relational graph structure. The flexibility of this knowledge representation formalism allows KGs to be widely used in various domains. Examples of KGs include general-purpose knowledge bases such as Wikidata [Vrandečić and Krötzsch 2014], DBpedia [Auer et al. 2007], Freebase [Bollacker et al. 2008], and YAGO [Suchanek et al. 2007]; application-driven graphs such as the Google Knowledge Graph, Microsoft’s Bing Knowledge Graph, and Facebook’s Social Graph [Noy et al. 2019]; and domain-specific ones such as SNOMED CT [Bodenreider et al. 2018], MeSH [Lipscomb 2000], and Hetionet [Himmelstein et al. 2017] for life sciences; and WordNet [Miller 1992] for linguistics.

*Equal contribution, alphabetical order. †Senior author.

Answering complex queries over Knowledge Graphs involves a logical reasoning process where a conclusion should be inferred from the available knowledge.

Neural link predictors [Nickel et al., 2016] tackle the problem of identifying missing edges in large KGs. However, in many domains, it is a challenge to develop techniques for answering complex queries involving multiple and potentially unobserved edges, entities, and variables rather than just single edges.

Prior work proposed to address this problem using specialised neural networks trained end-to-end for the query answering task [Hamilton et al., 2018; Daza and Cochez, 2020; Ren et al., 2020; Ren and Leskovec, 2020; Zhu et al., 2022], which offer little interpretability and require training with large and diverse datasets of query-answer pairs. These methods stand in contrast with Complex Query Decomposition [CQD, Arakelyan et al., 2021; Minervini et al., 2022], which showed that it is sufficient to re-use a simple link prediction model to answer complex queries, thus reducing the amount of training data required by orders of magnitude while allowing the possibility to explain intermediate answers. While effective, CQD does not support negations, and fundamentally, it relies on a link predictor whose scores are not necessarily calibrated for the complex query answering task. Adapting a neural link predictor for the query answering task while maintaining the data and parameter efficiency of CQD, as well as its interpretable nature, is the open challenge we take on in this paper.

We propose CQD\(^A\), a lightweight adaptation model trained to calibrate link prediction scores, using complex query answering as the optimisation objective. We define the adaptation function as an affine transformation of the original score with a few learnable parameters. The low parameter count and the fact that the adaptation function is independent of the query structure allow us to maintain the efficiency properties of CQD. Besides, the calibration enables a natural extension of CQD to queries with atomic negations.

An evaluation of CQD\(^A\) on three benchmark datasets for complex query answering shows an increase from 34.4 to 35.1 MRR over the current state-of-the-art averaged across all datasets while using $\leq 30\%$ of the available training query types. In ablation experiments, we show that the method is data-efficient; it achieves results comparable to the state-of-the-art while using only 1\% of the complex queries. Our experiments reveal that CQD\(^A\) can generalise across unseen query types while using only 1\% of the instances from a single complex query type during training.

2 Related Work

Link Predictors in Knowledge Graphs Reasoning over KGs with missing nodes has been widely explored throughout the last few years. One can approach the task using latent feature models, such as neural link predictors [Bordes et al., 2013; Trouillon et al., 2016; Yang et al., 2014; Dettmers et al., 2018; Sun et al., 2019; Balažević et al., 2019; Amin et al., 2020] which learn continuous representations for the entities and relation types in the graph and can answer atomic queries over incomplete KGs. Other research lines tackle the link prediction problem through graph feature models [Xiong et al., 2017; Das et al., 2017; Hildebrandt et al., 2020; Yang et al., 2017; Sadeghian et al., 2019], and Graph Neural Networks [GNNs, Schlichtkrull et al., 2018; Vashisht et al., 2019; Teru et al., 2020].

Complex Query Answering Complex queries over knowledge graphs can be formalised by extending one-hop atomic queries with First Order Logic (FOL) operators, such as the existential quantifier ($\exists$), conjunctions ($\land$), disjunctions ($\lor$) and negations ($\neg$). These FOL constructs can be represented as directed acyclic graphs, which are used by embedding-based methods that represent the queries
using geometric objects [Ren et al., 2020, Hamilton et al., 2018] or probabilistic distributions [Ren and Leskovec, 2020, Zhang et al., 2021, Choudhary et al., 2021] and search the embedding space for the answer set. It is also possible to enhance the properties of the embedding space using GNNs and Fuzzy Logic [Zhu et al., 2022, Chen et al., 2022]. A recent survey [Ren et al., 2023] provides a broad overview of different approaches. Recent work [Daza and Cochez, 2020, Hamilton et al., 2018, Ren and Leskovec, 2020] suggests that such methods require a large dataset with millions of diverse queries during the training, and it can be hard to explain their predictions.

Our work is closely related to CQD [Arakelyan et al., 2021, Minervini et al., 2022], which uses a pre-trained neural link predictor along with fuzzy logical t-norms and t-conorms for complex query answering. A core limitation of CQD is that the pre-trained neural link predictor produces scores not calibrated to interact during the complex query-answering process. This implies that the final scores of the model are highly dependent on the choice of the particular t-(co)norm aggregation functions, which, in turn, leads to discrepancies within the intermediate reasoning process and final predictions. As a side effect, the lack of calibration also means that the equivalent of logical negation in fuzzy logic does not work as expected.

With CQD-A, we propose a solution to these limitations by introducing a scalable adaptation function that calibrates link prediction scores for query answering. Furthermore, we extend the formulation of CQD to support a broader class of FOL queries, such as queries with atomic negation.

### 3 Background

A Knowledge Graph $\mathcal{G} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ can be defined as a set of subject-predicate-object $(s, p, o)$ triples, where each triple encodes a relationship of type $p \in \mathcal{R}$ between the subject $s \in \mathcal{E}$ and the object $o \in \mathcal{E}$ of the triple, where $\mathcal{E}$ and $\mathcal{R}$ denote the set of all entities and relation types, respectively. A Knowledge Graph can be represented as a First-Order Logic Knowledge Base, where each triple $(s, p, o)$ denotes an atomic formula $p(s, o)$, with $p \in \mathcal{R}$ a binary predicate and $s, o \in \mathcal{E}$ its arguments.

**First-Order Logical Queries** We are concerned with answering logical queries over incomplete knowledge graphs. We consider queries that use existential quantification ($\exists$) and conjunction ($\land$) operations. Furthermore, we include disjunctions ($\lor$) and atomic negations ($\neg$). We follow [Ren et al., 2020] by transforming a logical query into Disjunctive Normal Form [DNF, Davey and Priestley, 2002], i.e. a disjunction of conjunctive queries, along with the subsequent extension with atomic negations in [Ren and Leskovec, 2020]. We denote such queries as follows:

$$Q[A] \triangleq \exists V_1, \ldots, V_m. (c_1^1 \land \ldots \land c_{e_1}^1) \lor \ldots \lor (c_{e_m}^d \land \cdots \land c_{e_m}^d),$$

where $c_i^j = p(c, V)$, with $V \in \{A, V_1, \ldots, V_m\}, c \in \mathcal{E}, p \in \mathcal{R}$,

$$\text{or } c_i^j = p(V, V'), \text{ with } V, V' \in \{A, V_1, \ldots, V_m\}, V \neq V', p \in \mathcal{R}. \quad (1)$$

In Equation (1), the variable $A$ is the target of the query, $V_1, \ldots, V_m$ denote the bound variable nodes, while $c \in \mathcal{E}$ represent the input anchor nodes, which correspond to known entities in the query. Each $c_i$ denotes a logical atom, with either one $(p(c, V))$ or two variables $(p(V, V'))$.

The goal of answering the logical query $Q$ consists in finding the answer set $[Q] \subseteq \mathcal{E}$ such that $a \in [Q]$ iff $Q[a]$ holds true. As illustrated in Figure 1, the dependency graph of a conjunctive query $Q$ is a graph where nodes correspond to variable or non-variable atom arguments in $Q$ and edges correspond to atom predicates. We follow [Hamilton et al., 2018] and focus on queries whose dependency graph is a directed acyclic graph, where anchor entities correspond to source nodes, and the query target $A$ is the unique sink node.

**Example 3.1 (Complex Query).** Consider the question “Which people are German and produced the music for the film Constantine?” It can be formalised as a complex query $Q \equiv ?T : \text{country}(\text{Germany}, T) \land \text{producerOf}(\text{Constantine}, T)$, where $\text{Germany}$ and $\text{Constantine}$ are anchor nodes, and $T$ is the target of the query, as presented in Figure 1. The answer $[Q]$ corresponds to all the entities in the knowledge graph that are German composers for the film Constantine.

**Continuous Query Decomposition** CQD is a framework for answering EPFO logical queries in the presence of missing edges [Arakelyan et al., 2021, Minervini et al., 2022]. Given a query $Q$, CQD
defines the score of a target node \( a \in \mathcal{E} \) as a candidate answer for a query as a function of the score of all atomic queries in \( Q \), given a variable-to-entity substitution for all variables in \( Q \).

Each variable is mapped to an embedding vector that can either correspond to an entity \( c \in \mathcal{E} \) or to a virtual entity. The score of each of the query atoms is determined individually using a neural link predictor [Nickel et al., 2016]. Then, the score of the query with respect to a given candidate answer \( Q[a] \) is computed by aggregating all of the atom scores using t-norms and t-conorms – continuous relaxations of the logical conjunction and disjunction operators.

### Neural Link Predictors

A neural link predictor is a differentiable model where atom arguments are first mapped into a \( d \)-dimensional embedding space and then used to produce a score for the atom. More formally, given a query atom \( p(s, o) \), where \( p \in \mathcal{R} \) and \( s, o \in \mathcal{E} \), the score for \( p(s, o) \) is computed as \( \phi_p(e_s, e_o) \), where \( e_s, e_o \in \mathbb{R}^d \) are the embedding vectors of \( s \) and \( o \), and \( \phi_p : \mathbb{R}^d \times \mathbb{R}^d \rightarrow [0, 1] \) is a scoring function computing the likelihood that entities \( s \) and \( o \) are related by the relationship \( p \). Following [Arakelyan et al., 2021], Minervini et al., 2022, in our experiments, we use a regularised variant of ComplEx [Trouillon et al., 2016] as the neural link predictor of choice, due to its simplicity, efficiency, and generalisation properties [Ruffinelli et al., 2020]. To ensure that the output of the neural link predictor is always in \([0, 1]\), following [Arakelyan et al., 2021], Minervini et al., 2022, we use either a sigmoid function or min-max re-scaling.

### T-norms and Negations

Fuzzy logic generalises over Boolean logic by relaxing the logical conjunction (\( \land \)), disjunction (\( \lor \)) and negation (\( \neg \)) operators through the use of t-norms, t-conorms, and fuzzy negations. A \( t \)-norm \( \top : [0, 1] \times [0, 1] \rightarrow [0, 1] \) is a generalisation of conjunction in fuzzy logic [Klement et al., 2000, 2004]. Some examples include the Gödel t-norm \( \top_{\min}(x, y) = \min(x, y) \), the product t-norm \( \top_{\prod}(x, y) = x \times y \), and the Łukasiewicz t-norm \( \top_{\Luk}(x, y) = \max(0, x + y - 1) \). Analogously, \( t \)-conorms are dual to \( t \)-norms for disjunctions – given a \( t \)-norm \( \top \), \( \bot = 1 - \top \) is a \( t \)-conorm. In our experiments, we use the Gödel t-norm and product t-norm with their corresponding \( t \)-conorms.

Fuzzy logic also encompasses negations \( n : [0, 1] \rightarrow [0, 1] \). The standard \( n_{\text{stand}}(x) = 1 - x \) and strict cosine \( n_{\text{cos}} = \frac{1}{2}(1 + \cos(\pi x)) \) are common examples of fuzzy negations [Kruse and Moewes, 1993]. To support a broader class of queries, we introduce the standard and strict cosine functions to model negations in CQD, which was not considered in the original formulation of CQD.

### Continuous Query Decomposition

Given a DNF query \( Q \) as defined in Equation (1), CQD aims to find the variable assignments that render \( Q \) true. To achieve this, CQD casts the problem of query answering as an optimisation problem. The aim is to find a mapping from variables to entities \( S = \{ A \leftarrow a, V_1 \leftarrow v_1, \ldots, V_m \leftarrow v_m \} \), where \( a, v_1, \ldots, v_m \in \mathcal{E} \) are entities and \( A, V_1, \ldots, V_m \) are variables, that maximises the score of \( Q \):

\[
\arg \max_S \text{score}(Q, S) = \arg \max_{A, V_1, \ldots, V_m \in \mathcal{E}} \left( \prod c_i \right) \left( \prod \phi_p(e_{c_i}, e_{V_i}) \right)
\]

where \( c_i = \phi_p(e_{c_i}, e_{V_i}) \), with \( V \in \{ A, V_1, \ldots, V_m \}, c \in \mathcal{E}, p \in \mathcal{R} \).

or \( c_i = \phi_p(e_{c_i}, e_{V_i}) \), with \( V, V' \in \{ A, V_1, \ldots, V_m \}, V \neq V', p \in \mathcal{R} \).

where \( \prod \) and \( \lor \) denote a \( t \)-norm and a \( t \)-conorm – a continuous generalisation of the logical conjunction and disjunction, respectively – and \( \phi_p(e_{c_i}, e_{V_i}) \in [0, 1] \) denotes the neural link prediction score for the atom \( p(s, o) \).

### Complex Query Answering via Combinatorial Optimisation

Following [Arakelyan et al., 2021], Minervini et al., 2022, we solve the optimisation problem in Equation (2) by greedily searching for a set of variable substitutions \( S = \{ A \leftarrow a, V_1 \leftarrow v_1, \ldots, V_m \leftarrow v_m \} \), with \( a, v_1, \ldots, v_m \in \mathcal{E} \), that maximises the complex query score, in a procedure akin to beam search. We do so by traversing the dependency graph of a query \( Q \) and, whenever we find an atom in the form \( p(c, V) \), where \( p \in \mathcal{R}, c \) is either an entity or a variable for which we already have a substitution, and \( V \) is a variable for which we do not have a substitution yet, we replace \( V \) with all entities in \( \mathcal{E} \) and retain the top-\( k \) entities \( t \in \mathcal{E} \) that maximise \( \phi_p(e_{c}, e_{V}) \) – i.e. the most likely entities to appear as a substitution of \( V \) according to the neural link predictor. As we traverse the dependency graph of a query, we keep a beam with the most promising variable-to-entity substitutions identified so far.
Example 3.2 (Combinatorial Optimisation). Consider the query “Which musicians M received awards associated with a genre g?, which can be rewritten as \(?M : \exists A. \text{assoc}(g, A) \land \text{received}(A, M)\). To answer this query using combinatorial optimisation, we must find the top-k awards a that are candidates to substitute the variable A in \(\text{assoc}(g, A)\). This will allow us to understand the awards associated with the genre g. Afterwards, for each candidate substitution for A, we search for the top-k musicians m that are most likely to substitute M in \(\text{received}(A, M)\), ending up with \(k^2\) musicians. Finally, we rank the \(k^2\) candidates using the final query score produced by a t-norm. ■

4 Calibrating Link Prediction Scores on Complex Queries

The main limitation in the CQD method outlined in Section 3 is that neural link predictors \(\phi\) are trained to answer simple, atomic queries, and the resulting answer scores are not trained to interact with one another.

Example 4.1. Consider the running example query “Which people are German and produced the music for the film Constantine?” which can be rewritten as a complex query \(Q \equiv ?T : \text{country}(\text{Germany}, T) \land \text{producerOf}(\text{Constantine}, T)\). To answer this complex query, CQD answers the atomic sub-queries \(Q_1 = \text{country}(\text{Germany}, T)\) and \(Q_2 = \text{producerOf}(\text{Constantine}, T)\) using a neural link predictor, and aggregates the resulting scores using a t-norm. However, the neural link predictor was only trained on answering atomic queries, and the resulting scores are not calibrated to interact with each other. For example, the scores for the atomic queries about the relations country and producerOf may be on different scales, which causes problems when aggregating such scores via t-norms. Let us assume the top candidates for the variable T coming from the atomic queries \(Q_1, Q_2\) are \(\mathcal{A}_1 \leftarrow \text{Sam Shepard}\) and \(\mathcal{A}_2 \leftarrow \text{Klaus Badelt}\), with their corresponding neural link prediction scores 1.2 and 8.9, produced using \(\phi_{\text{country}}\) and \(\phi_{\text{producerOf}}\). We must also factor in the neural link prediction score of the candidate \(\mathcal{A}_1\) for query \(Q_2\) at 7.4 and vice versa at 0.5. When using the Gödel t-norm \(\min(x, y) = \min\{x, y\}\), the scores associated with the variable assignments \(\mathcal{A}_1, \mathcal{A}_2\) are computed as, \(\min(8.0, 0.5) = 0.5\ min(7.4, 1.2) = 1.2\). For both answers \(\mathcal{A}_1, \mathcal{A}_2\), the scores produced by \(\phi_{\text{country}}\) for \(Q_1\) are always lower than the scores produced with \(\phi_{\text{producerOf}}\) for \(Q_2\), meaning that the scores of the latter are not considered when producing the final answer. This phenomenon can be broadly observed in CQD, illustrated in Figure 2. ■

To address this problem, we propose a method for adaptively learning to calibrate neural link prediction scores by back-propagating through the complex query-answering process. More formally, let \(\phi_p\) denote a neural link predictor. We learn an additional adaptation function \(\rho_\theta\), parameterised by \(\theta = \{\alpha, \beta\}\), with \(\alpha, \beta \in \mathbb{R}\). Then, we use the composition of \(\rho_\theta\) and \(\phi_p\), \(\rho_\theta \circ \phi_p\), such that:

\[
\rho_\theta((\phi_p(e_v, e_{v'}))) = \phi_p(e_v, e_{v'})(1 + \alpha) + \beta.
\]

Here, the function \(\rho\) defines an affine transformation of the score and when the parameters \(\alpha = \beta = 0\), the transformed score \(\rho_\theta((\phi_p(e_v, e_{v'})))\) recovers the original scoring function. The parameters \(\theta\) can be conditioned on the representation of the predicate \(p\) and the entities \(V\) and \(V'\), i.e. \(\theta = \psi(e_v, e_p, e_{v'});\) here, \(\psi\) is an end-to-end differentiable neural module with parameters \(\mathbf{W}\). \(e_v, e_p, e_{v'}\) respectively denote the representations of the subject, predicate, and object of the atomic query. In our experiments, we consider using one or two linear transformation layers with a ReLU non-linearity as options for \(\psi\).

The motivation for our proposed adaptation function is twofold. Initially, it is monotonic, which is desirable for maintaining the capability to interpret intermediate scores, as in the original formulation of CQD. Moreover, we draw inspiration from the use of affine transformations in methodologies such as Platt scaling [Platt et al., 1999], which also use a linear function for calibrating probabilities and have been applied in the problem of calibration of link prediction models [Tabacof and Costabello, 2020]. Parameter-efficient adaptation functions have also been applied effectively in other domains, such as adapter layers [Houlsby et al., 2019] used for fine-tuning language models in NLP tasks.

Training For training the score calibration component in Equation 3, we first compute how likely each entity \(a' \in \mathcal{E}\) is to be an answer to the query \(Q\). To this end, for each candidate answer \(a' \in \mathcal{E}\), we compute the answer score as the complex query score assuming that \(a' \in \mathcal{E}\) is the final answer as:

\[
\text{score}(Q, A \leftarrow a') = \max_S \text{score}(Q, S), \quad \text{where } A \leftarrow a' \in S.
\]
Figure 2: The distributions of two atomic scores $Q_1$ and $Q_2$, and the aggregated results via $\top_{\text{min}}$ – the scores from $Q_2$ dominate the final scores.

Figure 3: Query structures considered in our experiments, as proposed by Ren and Leskovec [2020] – the naming of each query structure corresponds to projection (p), intersection (i), union (u) and negation (n), reflecting how they were generated in the BetaE paper [Ren and Leskovec, 2020]. An example of a pin query is $?T : \exists V. p(a, V), q(V, T), \neg r(b, T)$, where $a$ and $b$ are anchor nodes, $V$ is a variable node, and $T$ is the query target node.

Equation (4) identifies the variable-to-entity substitution $S$ that 1) maximises the query score $\text{score}(Q, S)$, defined in Equation (3), and 2) associates the answer variable $A$ with $a' \in E$, i.e. $A \leftarrow a' \in S$. For computing $S$ with the additional constraint that $A \leftarrow a' \in S$, we use the complex query answering procedure outlined in Section 4. We optimise the additional parameters $W$ introduced in Section 4 by gradient descent on the likelihood of the true answers on a dataset $D = \{(Q_i, a_i)\}_i$ of query-answer pairs by using a 1-vs-all cross-entropy loss, introduced by Lacroix et al. [2018], which was also used to train the neural link prediction model:

$$L(D) = \sum_{(Q_i, a_i) \in D} \text{score}(Q_i, A \leftarrow a_i) + \log \left( \sum_{a' \in E} \exp \left( \text{score}(Q_i, A \leftarrow a') \right) \right).$$

In addition to the 1-vs-all [Ruffinelli et al., 2020] loss in Equation (5), we also experiment with the binary cross-entropy loss, using the negative sampling procedure from Ren and Leskovec [2020].

5 Experiments

Datasets To evaluate the complex query answering capabilities of our method, we use a benchmark comprising of 3 KGs: FB15K [Bordes et al., 2013], FB15K-237 [Toutanova and Chen, 2015] and NELL995 [Xiong et al., 2017]. For a fair comparison with previous work, we use the datasets of FOL queries proposed by Ren and Leskovec [2020], which includes nine structures of EPFO queries and 5 query types with atomic negations, seen in Figure 3. The datasets provided by Ren and Leskovec [2020] introduce queries with hard answers, which are the answers that cannot be obtained by direct graph traversal; in addition, this dataset does not include queries with more than 100 answers, increasing the difficulty of the complex query answering task. The statistics for each dataset can be seen in Table 1. Note that during training, we only use 2i, 3i, 2in, and 3in queries, corresponding to $\leq 30\%$ of the training dataset, for the adaptation of the neural link predictor. To assess the model’s ability to generalise, we evaluate it on all query types.

Table 1: Statistics on the different types of query structures in FB15K, FB15K-237, and NELL995.

<table>
<thead>
<tr>
<th>Split</th>
<th>Query Types</th>
<th>FB15K</th>
<th>FB15K-237</th>
<th>NELL995</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>1p, 2p, 3p, 2i, 3i</td>
<td>273,710</td>
<td>149,689</td>
<td>107,982</td>
</tr>
<tr>
<td></td>
<td>2m, 3m, inp, pin, pni</td>
<td>27,371</td>
<td>14,968</td>
<td>10,798</td>
</tr>
<tr>
<td>Valid</td>
<td>1p</td>
<td>59,078</td>
<td>20,094</td>
<td>16,910</td>
</tr>
<tr>
<td></td>
<td>Others</td>
<td>8,000</td>
<td>5,000</td>
<td>4,000</td>
</tr>
<tr>
<td>Test</td>
<td>1p</td>
<td>66,990</td>
<td>22,804</td>
<td>17,021</td>
</tr>
<tr>
<td></td>
<td>Others</td>
<td>8,000</td>
<td>5,000</td>
<td>4,000</td>
</tr>
</tbody>
</table>
Compared to previous works, we have \( \approx \) Table 3, while maintaining competitive results.

Table 2: MRR results for FOL queries on the testing sets.

<table>
<thead>
<tr>
<th>Model</th>
<th>( \text{avg}_p )</th>
<th>( \text{avg}_{in} )</th>
<th>1p</th>
<th>2p</th>
<th>3p</th>
<th>2i</th>
<th>3i</th>
<th>pi</th>
<th>ip</th>
<th>2u</th>
<th>up</th>
<th>2in</th>
<th>3in</th>
<th>inp</th>
<th>pin</th>
<th>pni</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB15K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GQE</td>
<td>28.0</td>
<td>-</td>
<td>54.6</td>
<td>15.3</td>
<td>10.8</td>
<td>39.7</td>
<td>51.4</td>
<td>27.6</td>
<td>19.1</td>
<td>22.1</td>
<td>11.6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Q2B</td>
<td>38.0</td>
<td>-</td>
<td>68.0</td>
<td>21.0</td>
<td>14.2</td>
<td>55.1</td>
<td>66.5</td>
<td>39.4</td>
<td>26.1</td>
<td>35.1</td>
<td>16.7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>BetaE</td>
<td>41.6</td>
<td>11.8</td>
<td>65.1</td>
<td>25.7</td>
<td>24.7</td>
<td>55.8</td>
<td>66.5</td>
<td>43.9</td>
<td>28.1</td>
<td>40.1</td>
<td>25.2</td>
<td>15.3</td>
<td>16.7</td>
<td>11.5</td>
<td>6.5</td>
<td>12.4</td>
</tr>
<tr>
<td>CQD-CO</td>
<td>46.9</td>
<td>-</td>
<td>89.2</td>
<td>25.3</td>
<td>13.4</td>
<td>74.4</td>
<td>78.3</td>
<td>44.1</td>
<td>33.2</td>
<td>41.8</td>
<td>21.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>CQD-Beam</td>
<td>68.4</td>
<td>-</td>
<td>89.2</td>
<td>65.3</td>
<td>29.7</td>
<td>76.1</td>
<td>79.3</td>
<td>70.6</td>
<td>70.6</td>
<td>72.3</td>
<td>59.4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>ConE</td>
<td>49.8</td>
<td>14.8</td>
<td>73.3</td>
<td>33.8</td>
<td>29.2</td>
<td>64.4</td>
<td>73.7</td>
<td>50.9</td>
<td>35.7</td>
<td>55.7</td>
<td>31.4</td>
<td>17.9</td>
<td>18.7</td>
<td>12.5</td>
<td>9.8</td>
<td>15.1</td>
</tr>
<tr>
<td>GNN-QE</td>
<td>72.8</td>
<td>38.6</td>
<td>88.5</td>
<td>69.3</td>
<td>58.7</td>
<td>79.7</td>
<td>83.5</td>
<td>69.9</td>
<td>70.4</td>
<td>74.1</td>
<td>61.0</td>
<td>44.7</td>
<td>47.1</td>
<td>42.0</td>
<td>30.1</td>
<td>34.3</td>
</tr>
<tr>
<td>CQD(^4)</td>
<td>70.4</td>
<td>42.8</td>
<td>89.2</td>
<td>64.5</td>
<td>57.9</td>
<td>76.1</td>
<td>79.4</td>
<td>70.0</td>
<td>70.6</td>
<td>68.4</td>
<td>57.9</td>
<td>54.7</td>
<td>47.1</td>
<td>37.6</td>
<td>35.3</td>
<td>24.6</td>
</tr>
<tr>
<td>NELL1.995</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GQE</td>
<td>16.3</td>
<td>-</td>
<td>35.0</td>
<td>7.2</td>
<td>5.3</td>
<td>23.3</td>
<td>34.6</td>
<td>16.5</td>
<td>10.7</td>
<td>8.2</td>
<td>5.7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Q2B</td>
<td>20.1</td>
<td>-</td>
<td>40.6</td>
<td>9.4</td>
<td>6.8</td>
<td>29.5</td>
<td>42.3</td>
<td>21.2</td>
<td>12.6</td>
<td>11.3</td>
<td>7.6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>BetaE</td>
<td>20.9</td>
<td>5.5</td>
<td>39.0</td>
<td>10.9</td>
<td>10.0</td>
<td>28.8</td>
<td>42.5</td>
<td>22.4</td>
<td>12.6</td>
<td>12.4</td>
<td>9.7</td>
<td>5.1</td>
<td>7.9</td>
<td>7.4</td>
<td>3.5</td>
<td>3.4</td>
</tr>
<tr>
<td>CQD-CO</td>
<td>21.8</td>
<td>-</td>
<td>46.7</td>
<td>9.5</td>
<td>6.3</td>
<td>31.2</td>
<td>40.6</td>
<td>23.6</td>
<td>16.0</td>
<td>14.5</td>
<td>8.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>CQD-Beam</td>
<td>25.3</td>
<td>-</td>
<td>46.7</td>
<td>13.3</td>
<td>7.9</td>
<td>34.4</td>
<td>48.3</td>
<td>27.1</td>
<td>20.4</td>
<td>17.6</td>
<td>11.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>ConE</td>
<td>23.4</td>
<td>5.9</td>
<td>41.8</td>
<td>12.8</td>
<td>11.0</td>
<td>32.6</td>
<td>47.3</td>
<td>25.5</td>
<td>14.0</td>
<td>14.5</td>
<td>10.8</td>
<td>5.4</td>
<td>8.6</td>
<td>7.8</td>
<td>4.0</td>
<td>3.6</td>
</tr>
<tr>
<td>GNN-QE</td>
<td>26.8</td>
<td>10.2</td>
<td>42.8</td>
<td>14.7</td>
<td>11.8</td>
<td>38.3</td>
<td>54.1</td>
<td>31.1</td>
<td>18.9</td>
<td>16.2</td>
<td>13.4</td>
<td>10.0</td>
<td>16.8</td>
<td>9.3</td>
<td>7.2</td>
<td>7.8</td>
</tr>
<tr>
<td>CQD(^4)</td>
<td>25.7</td>
<td>10.7</td>
<td>46.7</td>
<td>13.6</td>
<td>11.4</td>
<td>34.5</td>
<td>48.3</td>
<td>27.4</td>
<td>20.9</td>
<td>17.6</td>
<td>11.4</td>
<td>13.6</td>
<td>16.8</td>
<td>7.9</td>
<td>8.9</td>
<td>5.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Details</td>
</tr>
<tr>
<td>Model Details</td>
</tr>
<tr>
<td>Model Details</td>
</tr>
<tr>
<td>Model Details</td>
</tr>
<tr>
<td>Model Details</td>
</tr>
<tr>
<td>Model Details</td>
</tr>
<tr>
<td>Model Details</td>
</tr>
<tr>
<td>Model Details</td>
</tr>
<tr>
<td>Model Details</td>
</tr>
<tr>
<td>Model Details</td>
</tr>
<tr>
<td>Model Details</td>
</tr>
<tr>
<td>Model Details</td>
</tr>
<tr>
<td>Model Details</td>
</tr>
<tr>
<td>Model Details</td>
</tr>
<tr>
<td>Model Details</td>
</tr>
</tbody>
</table>

Table 2: MRR results for FOL queries on the testing sets. \( \text{avg}_p \) designates the averaged results for EPFO queries (\( \wedge, \lor \)), while \( \text{avg}_{in} \) pertains to queries including atomic negations (\( \neg \)). The results for the baselines are from [Zhu et al., 2022].

**Evaluation Protocol** For a fair comparison with prior work, we follow the evaluation scheme in [Ren and Leskovec, 2020] by separating the answer of each query into easy and hard sets. For test and validation splits, we define hard queries as those that cannot be answered via direct traversal along the edges of the KG and can only be answered by predicting at least one missing link, meaning non-trivial reasoning should be completed. We evaluate the method on non-trivial queries by calculating the rank \( r \) for each hard answer against non-answers and computing the Mean Reciprocal Rank (MRR).

**Baselines** We compare CQD\(^4\) with state-of-the-art methods from various solution families in Section 2. In particular, we choose GQE [Hamilton et al., 2018], Query2Box [Ren et al., 2020], BetaE [Ren and Leskovec, 2020] and ConE [Zhang et al., 2021] as strong baselines for query embedding methods. We also compare with methods based on GNNs and fuzzy logic, such as FuzzQE [Chen et al., 2022], GNN-QE [Zhu et al., 2022], and the original CQD [Arakelyan et al., 2021] [Minervini et al., 2022], which uses neural link predictors for answering EPFO queries without any fine-tuning on complex queries.

**Model Details** Our method can be used with any neural link prediction model. Following [Arakelyan et al., 2021], [Minervini et al., 2022], we use ComplEx-N3 [Lacroix et al., 2018]. We identify the optimal hyper-parameters using the validation MRR. We train for 50,000 steps using Adagrad as an optimiser and 0.1 as the learning rate. The beam-size hyper-parameter \( k \) was selected in \( k \in \{512, 1024, \ldots, 8192\} \), and the loss was selected across \( L\)-vs-all [Lacroix et al., 2018] and binary cross-entropy with one negative sample.

**Parameter Efficiency** We use the query types \( 2i, 3i, 2in, 3in \) for training the calibration module proposed in Section 4. We selected these query types as they do not require variable assignments other than for the answer variable \( A \), making the training process efficient. As the neural link prediction model is frozen, we only train the adapter layers that have a maximum of \( W \in \mathbb{R}^{2 \times 2d} \) learnable weights. Compared to previous works, we have \( \approx 10^3 \) times fewer trainable parameters, as shown in Table 3 while maintaining competitive results.
We can also confirm that the adaptation process converges after \( \leq 2 \) with an average MRR difference of \( 2 \) in with query types. This shows that CQD does not require data-intensive training and allows the model to generalise to training on all training query types. CQD of the simplest complex query types. We see in Table 4 that CQD shows the most substantial increase in predictive accuracy on NELL995 by producing more accurate results than all other methods for all query types. CQD achieves a relative improvement of 6.8% to 37.1%, which can be attributed to the fact that the adaptation is completed with query types \( 2m \) and \( 3m \) that include negation, which allows for learning an adaptation layer that is robust for these types of queries. In our experiments, we found that calculating the neural adaptation parameters \( \theta \) of the adaptation function \( \rho_0 \) in Equation (3) as a function of the predicate representation yields the most accurate results followed by computing \( \theta \) as a function of the source entity and predicate representation, which is strictly more expressive. In Appendix A, we show the impact of the adaptation layers on the neural link prediction scores.

The adaptation process does not require data-intensive training and allows the model to generalise to query types not observed during training. This prompts us to investigate the minimal amount of data samples and query types required for adaptation.

### Data Efficiency

To analyse the data efficiency of CQD, we compare the behaviour of the pre-trained link predictors tuned with 1% and 100% of the training complex query examples in FB15K-237, presented in Table 4. For adapting on 1% of the training complex queries, we used the same hyper-parameters we identified when training on the full dataset. Even when using 1% of the complex training queries (3290 samples) for tuning, the model still achieves competitive results, with an average MRR difference of 2.2 compared to the model trained using the entire training set. CQD also produces higher test MRR results than GNN-QE with an average MRR increase of 30% across each dataset and query type. For queries including negations, CQD achieves accurate results than all other methods for all query types. CQD shows higher test MRR results also produces significantly higher test MRR results compared to the model trained using the entire training set.

We can also confirm that the adaptation process converges after \( \leq 10\% \) of the training epochs as seen in Figure 4. The convergence rate is not hindered when using only 1% of the training queries. This shows that CQD is a scalable method with a fast convergence rate that can be trained in a data-efficient manner.

### Out-of-Distribution Generalisation

To study the generalisation properties of CQD, we trained the adaptation layer on all atomic queries and only 1% of samples for one training query type \( 2i \), one of the simplest complex query types. We see in Table 4 that CQD can generalise to other types of complex queries not observed during training with an average MRR difference of 2.9 compared to training on all training query types. CQD also produces significantly higher test MRR results.
Table 4: Comparison of test MRR results for queries on FB15K-237 using the following training sets – FB237, 1% (resp. FB237 2i, 1%) means that, in addition to all 1p (atomic) queries, only 1% of the complex queries (resp. 2i queries) was used during training. As CQD^A uses a pre-trained link predictor, we also include all 1p queries when training GNN-QE for a fair comparison.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>1p</th>
<th>2p</th>
<th>3p</th>
<th>2i</th>
<th>3i</th>
<th>pi</th>
<th>ip</th>
<th>2u</th>
<th>up</th>
<th>2in</th>
<th>3in</th>
<th>inp</th>
<th>pin</th>
<th>pni</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB237, 1%</td>
<td>CQD^A</td>
<td>46.7</td>
<td>11.8</td>
<td>11.4</td>
<td>33.6</td>
<td>41.2</td>
<td>24.82</td>
<td>17.81</td>
<td>16.45</td>
<td>8.74</td>
<td>10.8</td>
<td>13.86</td>
<td>5.93</td>
<td>5.38</td>
<td>14.82</td>
</tr>
<tr>
<td>FB237 2i</td>
<td>GNN-QE</td>
<td>36.82</td>
<td>8.96</td>
<td>8.13</td>
<td>33.02</td>
<td>49.28</td>
<td>24.58</td>
<td>14.18</td>
<td>10.73</td>
<td>8.47</td>
<td>4.89</td>
<td>12.31</td>
<td>3.21</td>
<td>4.17</td>
<td>4.09</td>
</tr>
<tr>
<td>BetaE</td>
<td></td>
<td>36.80</td>
<td>6.89</td>
<td>5.94</td>
<td>22.84</td>
<td>34.34</td>
<td>17.03</td>
<td>13.43</td>
<td>9.56</td>
<td>4.66</td>
<td>1.84</td>
<td>2.54</td>
<td>2.54</td>
<td>2.54</td>
<td>2.54</td>
</tr>
</tbody>
</table>

Table 5: Test MRR results for FOL queries on FB15K-237 using the following CQD extensions: CQD from Arakelyan et al. [2021], Minervini et al. [2022] with the considered normalisation and negations; CQD_F, where we fine-tune all neural link predictor parameters in CQD; CQD^A_F, where we fine-tune all link predictor parameters in CQD^A; CQD_C, where we learn a transformation for the entity and relation embeddings and we use it to replace the initial entity and relation representations; and CQD_{RC}, where we learn a transformation for the entity and relation embeddings, and we concatenate it to the initial entity and relation representations.

<table>
<thead>
<tr>
<th>Model</th>
<th>CQD</th>
<th>CQD_F</th>
<th>CQD^A_F</th>
<th>CQD_C</th>
<th>CQD_{RC}</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB237</td>
<td>13.2</td>
<td>34.5</td>
<td>48.2</td>
<td>26.8</td>
<td>20.3</td>
</tr>
<tr>
<td>FB237 2i</td>
<td>9.3</td>
<td>22.8</td>
<td>34.9</td>
<td>19.8</td>
<td>14.5</td>
</tr>
<tr>
<td>BetaE</td>
<td>9.5</td>
<td>23.9</td>
<td>39.0</td>
<td>19.8</td>
<td>14.5</td>
</tr>
<tr>
<td>CQD</td>
<td>10.9</td>
<td>33.7</td>
<td>47.3</td>
<td>25.6</td>
<td>18.9</td>
</tr>
<tr>
<td>CQD_R</td>
<td>6.4</td>
<td>22.2</td>
<td>31.0</td>
<td>16.6</td>
<td>11.2</td>
</tr>
<tr>
<td>CQD^A</td>
<td>13.2</td>
<td>35.0</td>
<td>48.5</td>
<td>27.3</td>
<td>20.7</td>
</tr>
</tbody>
</table>

than GNN-QE, with an average increase of 5.1 MRR. The greatest degradation in predictive accuracy occurs for the queries containing negations, with an average decrease of 2.7. This prompts us to conjecture that being able to answer general EPFO queries is not enough to generalise to the larger set of queries, which include atomic negation. However, our method can generalise on all query types, using only 1% of the 2i queries, with 1496 overall samples for adaptation.

Fine-Tuning All Model Parameters One of the reasons for the efficiency of CQD^A is that the neural link predictor is not fine-tuned for query answering, and only the parameters in the adaptation function are learned. We study the effect of fine-tuning the link predictor using the full training data for CQD and CQD^A on FB15K-237. We consider several variants: 1) CQD_F, where we fine-tune all neural link predictor parameters in CQD; 2) CQD^A_F, where we fine-tune all link predictor parameters in CQD^A; 3) CQD_C, where we learn a transformation for the entity and relation embeddings and we use it to replace the initial entity and relation representations; and 4) CQD_{RC}, where we learn a transformation for the entity and relation embeddings, and we concatenate it to the initial entity and relation representations.

It can be seen from Table 5 that CQD^A yields the highest test MRR results across all query types while fine-tuning all the model parameters produces significant degradation along all query types, which we believe is due to catastrophic forgetting of Goodfellow et al. [2013] of the pre-trained link predictor.

6 Conclusions

In this work, we propose the novel method CQD^A for answering complex FOL queries over KGs, which increases the averaged MRR over the previous state-of-the-art from 34.4 to 35.1 while using ≤ 30% of query types. Our method uses a single adaptation layer over neural link predictors, which allows for training in a data-efficient manner. We show that the method can maintain competitive predictive accuracy even when using 1% of the training data. Furthermore, our experiments on training on a subset (1%) of the training queries from a single query type (2i) show that it can generalise to
new queries that were not used during training while being data-efficient. Our results provide further evidence for how neural link predictors exhibit a form of compositionality that generalises to the complex structures encountered in the more general problem of query answering. CQD4 is a method for improving this compositionality while preserving computational efficiency. As a consequence, rather than designing specialised models trained end-to-end for the query answering task, we can focus our efforts on improving the representations learned by neural link predictors, which would then transfer to query answering via efficient adaptation, as well as other downstream tasks where they have already proved beneficial, such as clustering, entity classification, and information retrieval.

Acknowledgements

Pasquale was partially funded by the European Union’s Horizon 2020 research and innovation programme under grant agreement no. 875160, ELIAI (The Edinburgh Laboratory for Integrated Artificial Intelligence) EPSRC (grant no. EP/W002876/1), an industry grant from Cisco, and a donation from Accenture LLP, and is grateful to NVIDIA for the GPU donations. Daniel and Michael were partially funded by Elsevier’s Discovery Lab. Michael was partially funded by the Graph-Massivizer project (Horizon Europe research and innovation program of the European Union under grant agreement 101093202). Erik is partially funded by a DFF Sapere Aude research leader grant under grant agreement No 0171-00034B, as well as by a NEC PhD fellowship, and is supported by the Pioneer Centre for AI, DNRF grant number P1. Isabelle is partially funded by a DFF Sapere Aude research leader grant under grant agreement No 0171-00034B, as well as by the Pioneer Centre for AI, DNRF grant number P1. This work was supported by the Edinburgh International Data Facility (EIDF) and the Data-Driven Innovation Programme at the University of Edinburgh.

References


A Impact of adaptation

We investigate the effect of the adaptation process in CQD$^A$ by comparing the score of the neural link predictor before and after applying the adaptation layer. As we see from Figure 5, the scores before adaptation have a variation of 0.04 with the boundaries at $[-8, 12]$. This makes them problematic for complex query answering as discussed in Section 4. The Adapted scores have a smaller variation at 0.03 while the maximum and minimum lie in the $[0, 1]$ range.

B On data efficiency and generalisation of CQD$^A$

We conduct a series of experiments comparing the performance of CQD$^A$ trained while only using 1% of the training data to the complete training set in FB15K-237. We see from Figure 6 (top) that the model maintains a strong performance on the complex reasoning task with a 2.4 averaged MRR degradation compared to using the complete data. This phenomenon is even more pronounced when we use only 1% of the queries while using 2i as our training query types Figure 6 (bottom). In this data and query type constrained mode CQD$^A$ maintains competitive performance with a degradation of 2.7 averaged MRR for complex queries compared to training with complete data. This highlights the data-efficient nature of CQD$^A$ and shows that training a score adaptation layer does not require significant data for training. This is also accompanied by the observation that CQD$^A$ is able to generalise to unseen query types while training only on 2i. This behaviour is not inherent, as seen from Table 4.