Complex Query Answering in the Biomedical Domain

Fredrik Skjelvik[1]

Vrije Universiteit Amssterdam, Netherlands fredrikerichsen16@gmail.com

Abstract. Continuous Query Decomposition (CQD) is a framework for answering complex logical queries on incomplete knowledge graphs. The atomic propositions in these logical queries correspond to the presence or absence of triples in the knowledge graph, and their truth value is calculated as a probability using a pre-trained link predictor. The total query score is calculated with fuzzy logic equivalents of logical conjunction and disjunction. CQD is able to answer multi-hop queries that require projecting the value of one or more variable nodes. Two methods are proposed to optimize the embedding of the variable node(s) such that the query score is maximized. One of the potential applications of Complex Query Answering is drug development. In this paper, CQD is tested on the biomedical knowledge graph BioKG and the results are shown and evaluated. The results are promising, but should be interpreted with caution as the link predictor achieved much higher accuracy than what was reported in a different paper with the same model and dataset.

Keywords: Query answering · Knowledge graphs · Knowledge graph embedding

1 Introduction

A knowledge graph (KGs) is a data model that stores information as a directed, labeled, multi-relational graph consisting of interconnected entities connected by relations. The basic building block of knowledge graphs is the triple, which has the form (head entity, relation, tail entity). Triples indicate that two entities are connected by a relation, thus denoting a fact. For example (Rome, capital_of, Italy). Knowledge Graphs are very flexible and effective at storing structured knowledge about different domains, ranging from general-purpose KGs like FB15k [3] and NELL995 [5] to domain-specific ones such as BioKG [2], which is used in the biomedical domain for drug discovery, drug repurposing, and other tasks.

By means of knowledge graph embedding, it is possible to learn a dense, low-dimensional vector representation of KG entities and relations which retains information about the structure and/or semantics of the components. This vector representation is in a form that can be used in downstream machine learning tasks, such as link prediction (predicting whether two entities are connected by a relation) and query answering (answering complex logical queries on KGs with multiple atoms). [11] [12]

Drug development and research is costly and time-consuming. Getting a drug to market takes upwards of 15 years and costs around 2 billion USD [20]. The majority of drugs never make it through the complex and highly regulated drug development pipeline to market. Various computational approaches are being investigated, including relational machine learning, to improve the success rate of the drug discovery process. In 2011, AstraZeneca revised its RD process to include, among other things, a greater emphasis on computational methods like high throughput screening and virtual screening in the early lead generation stage. From 2005-2010, before the introduction of AstraZeneca's new framework, only 23% of new drug target discovery projects delivered leads of sufficient quality to move on to the next stage of testing, and the overall success rate from drug discovery to phase III completion was 4%. After implementing the new framework, from 2012-2016, those figures increased to 48% and 19%, respectively. [19]

Relational learning on KGs is a promising new research field that seems to be well-suited for biomedical applications. It may be applied in the pharmaceutical industry to a variety of tasks including drug- discovery, repurposing, interaction screening and toxicity screening [2] [21]. There is an ever increasing amount of structured data in this domain so the central question becomes how to derive insights from the data. There are many properties of KGs and relational learning that make them suited for this task. KGs are capable of storing heterogenous data (i.e. multiple entity types and multiple relation types), whereas traditional graphs used before were homogenous [20]. So, while the graphs used before might only contain information about proteins, KGs can store information about proteins, genes, drugs, diseases, etc. in one database, thereby making it possible to model complex biological systems. Given a biomedical KG, many problems can be framed as a link prediction problem. For example, drug interaction screening can be done by predicting links between drug entities (drug-drug interactions) and drug target identification can be done by predicting links between drug entities and protein entities. Complex query answering goes a step further by finding connections between nodes separated by more than one edge, and as a result could unlock exponentially more information from the same model.

Recently a framework named Continuous Query Decomposition (CQD) [1] has been proposed for answering complex queries on incomplete Knowledge graphs. Only training the model on link prediction, it demonstrated the ability to answer logical queries using conjunction ($\wedge$), disjunction ($\vee$), and the existential quantifier ($\exists$) with state-of-the-art accuracy.

The purpose of this paper is to experiment with CQD on the biomedical knowledge graph BioKG and report the results. In the original paper, the performance of CQD was evaluated on three common benchmark KGs with general information, namely FB15k [3], FB15k-237 [4], and NELL995 [5]. It is not yet known how CQD will perform in the biomedical domain, specifically on the BioKG dataset. Typical benchmark KGs may not reflect the domain-specific properties of biomedical KGs, as there are some fundamental differences between them. Biomedical knowledge graphs tend to be larger, display higher average connectivity [9], and contain richly structured ontological hierarchies [7].

2 Background

2.1 Knowledge Graphs

A knowledge graph $\mathcal{G} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is a set of triples $(h, r, t) \in \mathcal{F}$ denoting a relationship between the head and tail entities $h, t \in \mathcal{E}$ of relationship type $r \in \mathcal{R}$, with $\mathcal{E}$ and $\mathcal{R}$ representing the set of all entities and relationship types, respectively.

2.2 Knowledge graph embedding

Knowledge graph embedding (KGE) is the machine learning task of embedding components of the KG (entities and relations) into a low dimensional vector space which encodes information about the components. KG embedding usually consists of three steps: (i) representing entities and relations (usually as a vector), (ii) defining a scoring function to calculate the plausibility of some fact (h, r, t), and (iii) learning entity and relation embeddings by optimizing the embedding such that the plausibility of observed facts is maximized and (optionally) the plausibility of some generated negative samples is minimized. [11]

KGE techniques are roughly categorized into translational distance models, which use distance-based scoring functions, and semantic matching models, which use similarity-based scoring functions. The KGE model used in this paper is called ComplEx and is of the latter type. In this section, ComplEx is explained by first introducing RESCAL and then which modifications ComplEx has compared to RESCAL.

Notation

- $\mathcal{G}$ A knowledge graph
- $\mathcal{E}$ The set of entities
- $\mathcal{R}$ The set of relations
- N_e Number of entities in the knowledge graph
- N_r Number of relations in the knowledge graph
- H_e Dimension size of entity/relation embedding
- h, r, t - Head entity, relation, tail entity
- $\mathbf{e}_x$ - Embedding of entity x
- $\mathbf{r}_x$ - Embedding of relation x
- $\phi(\mathbf{e}_h, \mathbf{r}_r \mathbf{e}_t)$ - Score function of triple $\langle h, r, t \rangle$
- $\mathbf{X} \in \{0, 1\}^{N_e \times N_e \times N_e}$ - Adjacency tensor representing a knowledge graph. The entry $x_{ijk} = 1$ if the triple $\langle e_i, r_k, e_j \rangle$ exists, and $x_{ijk} = 0$ otherwise.

Semantic matching models RESCAL is a semantic matching model and the first modern KGE approach. [13] It it still the foundation of later models in the same category, such as DistMult and ComplEx. The RESCAL score function in compact matrix notation is $\mathbf{F}_k = \mathbf{E}\mathbf{W}_k\mathbf{E}^T$, where $\mathbf{E} \in \mathbb{R}^{N_e \times H_e}$ is a matrix of entity embeddings, $\mathbf{W}_k \in \mathbb{R}^{H_e \times H_e}$ is the k-th slice of the weight matrix $\mathbf{W}$, i.e. the weights associated with the k-th relation, and $\mathbf{F}_k \in \mathbb{R}^{N_e \times N_e}$ is the k-th slice of the matrix holding the scores for all the triples. $\mathbf{F}$ and $\mathbf{X}$, the adjacency tensor representation of the KG, have the same dimension size $N_e \times N_e \times N_r$, and while the entry x_{ijk} holds the truth value of whether the triple $\langle e_i, r_k, e_j \rangle$ exists, the entry f_{ijk} holds the score of the triple x_{ijk}. The goal is to jointly optimize the entity embeddings $\mathbf{E}$ and weight matrix $\mathbf{W_k}$ by minimizing the loss of $\mathbf{F}$ against $\mathbf{X}$. The weight matrix specifies at entry w_{abk} how much the latent features a and b interact in the k-th relation. The scores are calculated as the weighted sum of every pair of entities and every pair of latent features associated with the k-th relation (i.e. every entry in the weight matrix at the k-th slice). [11]

DistMult is a simplification of RESCAL which restricts $\mathbf{W}_k$ to diagonal matrices, but it is only able to model symmetric relationships. [11]

ComplEx extends DistMult by introducing complex valued representations for entities and relationships. The embeddings are represented as vectors $\mathbf{e}, \mathbf{r} \in \mathbb{C}^{H_e}$ and the score of a triple (h, r, t) is defined as

$$\phi(\mathbf{e}_h, \mathbf{r}_r, \mathbf{e}_t) = Re(\mathbf{e}_h^\top diag(\mathbf{r}_r)\overline{\mathbf{e}_t}) = Re(\sum_{i=0}^{d-1} [\mathbf{r}_r]_i \cdot [\mathbf{e}_h]_i \cdot \overline{[\mathbf{e}_t]}_i) \tag{1}$$

where $\overline{\mathbf{e}_t}$ is the complex conjugate of $\mathbf{e}_t$ and $Re(\mathbf{e}_x)$ denotes the real part of the complex valued vector $\mathbf{e}_x$. ComplEx is able to model symmetric and antisymmetric relations. [8]

2.3 Link Prediction

Link prediction is the task of predicting missing relations in a knowledge graph. Given a triple with one entity missing, the task is to predict the missing entity, denoted (?, r, t) for head prediction and (h, r, ?) for tail prediction. Every candidate entity is scored by a differentiable function which takes as its arguments embeddings of a head entity, relation, and tail entity and returns the plausibility that those the entities are connected by that relation. I.e. $\phi(\mathbf{e}_h, \mathbf{r}_r, \mathbf{e}_t) : \mathbb{R}^{H_e} \times \mathbb{R}^{H_e} \times \mathbb{R}^{H_e} \rightarrow [0, 1]$

Link prediction is often necessary because of the incompleteness of virtually all KGs, so subgraph matching with e.g. the SPARQL querying language cannot be relied upon.

Link prediction based on KG embeddings have achieved strong results on benchmarks in recent years [17]. KGs contain patterns of connections which are encoded in the embeddings thereby enabling link prediction. Semantic matching models encode semantic information about entities by the principle that entities tend to be related to entities with similar characteristics (i.e. connected to similar entities via similar relations). [11]

3 Complex Query Decomposition

The CQD framework enables answering queries on KGs, provided that the queries are expressed as an Existential Positive First-Order (EPFO) logical query and the atomic propositions denote the presence (true) or absence (false) of a triple in the KG. The class of EPFO logical formulas includes those that may use the operators conjunction ($\wedge$), disjunction ($\vee$), and the existential quantifier ($\exists$). The atomic propositions are represented with an atomic formula $p(s, o)$, with $p \in \mathcal{R}$ a binary predicate and $s, o \in \mathcal{E}$ its arguments. The formula is true if the triple $\langle s, p, o \rangle$ exists in the KG. (s, p, and o stand for subject, predicate and object).

CQD can in principle generalize to any EPFO query whose dependency graph is directed and acyclic, but a selection of nine structures have been evaluated. In this paper four query structures will be tested, which only include conjunction and the existential quantifier, not disjunction.

So a query like *"Which drug D targets protein P associated with genetic disorder Cystic Fibrosis?"* can be expressed as $?D : \exists P.assoc(\text{Cystic Fibrosis}, P) \wedge target\_of(P, D)$. *"Which protein interacts with insulin and is associated with diabetes?"* can be expressed as $?P : \exists P.ppi(P, \text{insulin}) \wedge assoc(P, \text{diabetes})$ (ppi stands for protein-protein interaction). The former has a query structure denoted 2-hop or 2p because it involves double link prediction and projecting the value of the bound variable node, and the latter is called 2i because it involves an intersection where two atoms have the same object. These two query types, as well as 1p (one-hop link prediction) and ip (intersection followed by projection) will be experimented with in this paper. See figure 1 for an illustration of the different query structures.

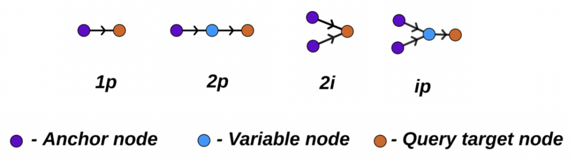


Fig. 1: Illustration of the query types included in experiments

3.1 Conjunctive Queries

The form of conjunctive queries that can be answered is defined as follows

$$\begin{aligned} Q[A] = ?A : \exists V_1, ..., V_m . e_1 \wedge ... \wedge e_n \\ \text{where } e_i = p(c, V), \text{ with } V \in \{A, V_1, ..., V_m\}, c \in \mathcal{E}, p \in \mathcal{R} \\ \text{or } e_i = p(V, V'), \text{ with } V, V' \in \{A, V_1, ..., V_m\}, V \neq V', p \in \mathcal{R} \end{aligned} \tag{2}$$

where A is the *target node* of the query (the answer), $V_1, ..., V_m$ are the *bound variable nodes*, and $c \in \mathcal{E}$ represent the *input anchor nodes*. Each e_i denotes an atomic proposition, with either one $(p(c, V))$ or two $(p(V, V'))$ variable nodes. The goal is to find an answer set of entities $a \in \mathcal{E}$ for which $Q[a]$ holds. The bound variable nodes and target node are hereafter collectively referred to as variable nodes.

A pre-trained link predictor is used to calculate the plausibility of each atomic proposition. Recall that link predictors take embeddings as arguments, so the vector representations of c, p and V_i are used. c and p use the embedding learned during training on link prediction, but V_i is an unknown variable node, so it either initialized with a random embedding and optimized (continuous CQD) or filled with the embedding of different combinations of entities (discrete CQD).

Equation 3 expands upon equation 2, introducing the optimization task, link prediction, and t-norm.

$$\begin{aligned} &\underset{A, V_1, ..., V_m \in \mathcal{E}}{arg\,max} \; e_1 \top ... \top e_n \\ \text{where } e_i &= \phi(\mathbf{e}_c, \mathbf{r}_p, \mathbf{e}_V), \text{ with } V \in \{A, V_1, ..., V_m\}, c \in \mathcal{E}, p \in \mathcal{R} \\ \text{or } e_i &= \phi(\mathbf{e}_V, \mathbf{r}_p, \mathbf{e}'_V), \text{ with } V, V' \in \{A, V_1, ..., V_m\}, p \in \mathcal{R} \end{aligned} \tag{3}$$

$\phi(\mathbf{e}_s, \mathbf{r}_p, \mathbf{e}_o)$ is the link prediction score for atomic proposition $p(s, o)$. $\top$ denotes t-norm - a fuzzy logic generalization of conjunction in propositional logic with various implementations, including Gödel t-norm ($\top_{min}(x, y) = min\{x, y\} \in [0, 1]$) and product t-norm ($\top_{prod}(x, y) = x \cdot y$). The t-norms aggregate the individual link prediction scores into an overall score for a given query and candidate answer. Finally, the embeddings for the variable nodes ($\mathbf{e}_V$) that maximizes the query score are found.

3.2 Continuous Optimization

In continuous optimization (CQD-CO), the variable embedding representations are optimized directly without any correspondence to or mapping from any known entities. That is, the optimization task is equivalent to equation 3 but with the following as the argmax.

$$\underset{\mathbf{e}_A, \mathbf{e}_{V_1}, ..., \mathbf{e}_{V_m} \in \mathcal{E}}{arg\,max} \tag{4}$$

Continuous Optimization can be performed with any gradient-based optimization method such as gradient descent or Adam. When the optimal variable embeddings $\mathbf{e}_A, \mathbf{e}_{V_1}, ..., \mathbf{e}_{V_m}$ have been identified, every entity in the KG is scored as a candidate answer, i.e. $\mathbf{e}_A$ is replaced with $\mathbf{e}_t \in \mathbb{R}^{H_e}$ once for every $t \in \mathcal{E}$. The query score is calculated with this substitution and the t substitution with the highest query score is returned as the answer.

3.3 Combinatorial Optimization

In combinatorial optimization (CQD-Beam) a set of variable substitutions $S = \{A \leftarrow a, V_1 \leftarrow v_1, ..., V_n \leftarrow v_n\}$ with $a, v_1, ..., v_n \in \mathcal{E}$ that maximizes the query score (equation 3) is found by means of a greedy algorithm that finds the top k best variable substitutions for each atomic query. Starting with atoms in the form $p(c, V)$, the link prediction score is calculated for every substitution of V with entity $t \in \mathcal{E}$: $\phi(\mathbf{e}_c, \mathbf{r}_r, \mathbf{e}_t)$. The top-k scoring substitutions t are retained. Next, atoms in the form $p(V', V)$ are handled, where V' is a variable node for which a set of variable-to-entity substitutions has been found. For each substitution of V', $p(t, V)$, we find k substitutions for V by the same process. We end up with k^m combinations of variable-to-entity substitutions, where m is the number of edges in the query graph. The combination of variable substitutions that maximizes the query score is calculated. This contains both the answer to the query and intermediate variable assignments as well as several potential other assignments. This feature makes CQD-Beam explainable, which can be useful in many cases.

4 BioKG

BioKG [2] is a biomedical knowledge base generated from highly reputable, expert-curated open biological databases, including DrugBank [16], UniProt [15] and Gene Ontology [14]. The main biological entities in BioKG are drugs (DR), proteins (PR), pathways (PA), genetic disorders (GD), and diseases (DS) and the relations between these comprise the main relations, e.g. drug-protein interactions (DPIs), protein-protein interactions (PPIs), and protein/drug/disease pathways. There are some additional entities and relations which are subtypes. For example, enzymes are types of proteins, and drug transporters and drug carriers are types of DPIs. Figure 2 shows the schema of BioKG.

BioKG was developed to address the lack of high quality, unified biomedical knowledge graphs. Most open biological datasets cover a specific domain such as gene ontologies or drug development. Researchers often had to construct custom biomedical knowledge graphs from the available open-source databases. However, this process was often carried out inconsistently across efforts and sometimes low quality datasets were used. BioKG aims to make relational learning on biomedical data more standardized and reproducible. [2]

5 Experiments

5.1 Dataset and Query Sampling

BioKG consists of 2,067,997 triples, 105,524 unique entities and 17 relation types. The dataset was split into a training-, validation-, and test set in the ratio 0.70/0.10/0.20.

For the complex query answering task, queries were sampled using an open source Graph Query Sampler[1], which uses the RDF query language SPARQL to

[1] https://github.com/miselico/graph_query_sampler

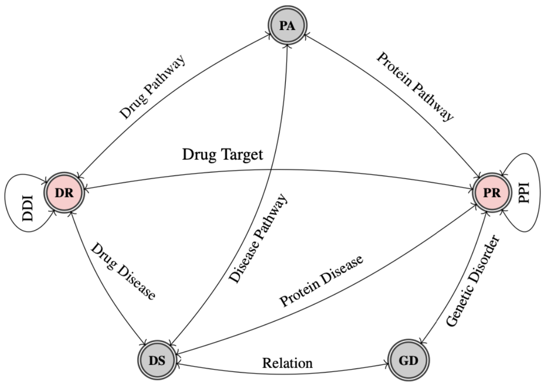


Fig. 2: BioKG Shema

generate queries and answers for different query structures (specifically 1p, 2p, ip, and pi in these experiments). The dataset generated by this process was also split into a training set, validation set, and test set. (Only one-hop queries are included in the training set, while the remaining sets contain queries of all types). One-hop queries are sampled only from triples present in the corresponding initial train/test/validation split, so each split generated by the query sampler is equal to or a subset of the initial corresponding split. The remaining complex query types were sampled from the entire KG, irrespective of splits. As a consequence, some answers are trivial since they can be answered only using triples present in the training set. Only the hard answers, which require the link predictor to correctly predict one or more unseen edges, are counted for evaluating the model.

Entities with high in-degree (i.e. with many incoming connections) can present a problem for representative query sampling as a small number of entities might be vastly overrepresented as the answer to queries, thereby enabling query answering methods to achieve good Hits@K results by simply guessing the top K most frequent answers every time. For example, in BioKG the highest observed in-degree is 2872. For the 2i pattern, $\binom{2872}{2} = 4\,122\,756$ different queries can be generated with that entity as the answer. For query types that do not include intersection the effect is similar, but less extreme. It has been proposed [10] to remove entities with an in-degree above a certain threshold as a solution to this problem. For these experiments, the in-degree threshold was set at 50 for the complex query types (i.e. not including one-hop) which removed 5993 entities, around 5% of the total. A crude model was created that always predicts the top 10 most frequent answers (calculated separately per query type and after entities above the in-degree threshold were removed) in descending order of frequency. Table 1 shows the results of applying this model to BioKG. The poor performance shows that the dataset does not have the mentioned problem after removing high in-degree entities. This table can also be seen as a kind of benchmark because any model can achieve roughly these results by overfitting to the most frequent answers. Performance that exceeds this benchmark reflects genuine model performance.

Query Structure	**Hits@1**	**Hits@3**	**Hits@10**
1p	0.69%	0.95%	4.14%
2p	0.01%	0.12%	0.29%
2i	0.09%	0.26%	0.74%
ip	0.06%	0.14%	0.63%

Table 1: Hits@K for model which always guesses the top ten most frequent answers

5.2 Model details

The link predictor was trained with ComplEx using a variational approximation of the nuclear tensor p-norm for regularization. The learning rate was fixed at 0.1, batch size at 200, and the Adagrad optimizer was used. Grid search was performed for hyperparameter optimization (HPO) with the following hyperparamters and values: embedding dimension in $\{100, 200, 500\}$ and regularization coefficient in $\{0.1, 0.01, 0.001\}$. The best performance was achieved with embedding dimension 500 and regularization coefficient 0.01.

For CQD-CO, variable and target embeddings were optimized with the Adam optimizer with an initial learning rate of 0.1 and a maximum of 1,000 iterations. For CQD-Beam, the optimal beam size was searched in the set $k \in \{16, 32, ..., 128\}$. In addition, for both types of CQD, two variations of t-norm - Gödel and product t-norm - were tested. Table 2 shows the best configuration that was found per query type.

Query Type	**CQD Type**	**T-Norm**	**K**	**Hits@10**
2p	CO	Gödel	N/A	25.08%
2i	Beam	Product	16∼128	66.72%
ip	Beam	Product	64∼128	31.07%

Table 2: Configuration with best query answering performance found in grid search. Different values of K within the range indicated in the table resulted in negligible changes $< 0.05\%$ in all performance metrics. The reported result uses the highest K in the range.

5.3 Evaluation

For each query in the test set, every entity in the KG is scored as a possible answer and this list is sorted decreasingly by score. An entity's rank is its position in this list. Moreover, the filtered setting [3] will be used, whereby other true answers do not affect the ranking of an entity. So, if three correct entities occupy the top three positions in the list, their filtered ranks are all 1. The filtered setting is important for the trivial answers which are not included in the calculation of performance metrics. If it were not for the filtered settings, trivial answers would often outrank hard answers and thereby reduce the measured performance.

Given $\mathcal{Q}$ as the set of filtered ranks for all ground-truth correct answers, the performance metric reported in the results is defined as follows.

Hits@K measures the proportion of model predictions that are among the top K highest ranked. Typically $k = \{1, 3, 10\}$ are used.

$$\text{Hits@k} = \frac{|\{q \in Q : q < k\}|}{|Q|} \tag{5}$$

6 Results

The results of complex query answering are shown in table 3. The performance metrics are shown for the best model achieved per query type and CQD method.

Method	**Avg**	**1p**	**2p**	**2i**	**ip**
Hits@1					
CQD-CO	**23.60%**	**43.34%**	**13.08%**	27.28%	10.72%
CQD-Beam	23.53%	**43.34%**	8.64%	**29.27%**	**12.88%**
Hits@3					
CQD-CO	33.67%	**54.97%**	**18.42%**	44.02%	17.27%
CQD-Beam	**34.50%**	**54.97%**	13.95%	**48.44%**	**20.63%**
Hits@10					
CQD-CO	44.06%	**65.91%**	**25.08%**	59.13%	26.02%
CQD-Beam	**46.41%**	**65.91%**	21.95%	**66.72%**	**31.07%**

Table 3: CQD complex query answering results by query type and CQD variation.

In table 4 the query answering performance is compared between BioKG and the three benchmark datasets evaluated in the paper that introduced the CQD framework [1]. The cells contain the highest Hits@3 achieved per query type and dataset with either CQD-CO or CQD-Beam. CQD on BioKG performed slightly worse on average than on NELL995 and slightly better than on FB15k-237. Thus, despite the fundamental differences between these benchmark datasets

Dataset	Avg	1p	2p	2i	ip
FB15k	71.70%	91.80%	77.90%	79.60%	37.50%
FB15k-237	32.45%	51.20%	28.80%	35.20%	14.60%
NELL995	40.58%	66.70%	35.00%	41.00%	19.60%
BioKG	35.60%	54.90%	18.42%	48.44%	20.63%

Table 4: CQD complex query answering results by query type and CQD variation.

and BioKG mentioned in the introduction, CQD achieved similar performance on BioKG as with two out of the three benchmark datasets.

Query answering may be trained on heterogenous KGs, but in practice someone might be interested in predicting a small number of relationship types for a specific use case. Therefore, looking at the performance broken down by what relation types are involved in the query atoms can be enlightening. Another reason why this breakdown might be interesting is that the test queries were generated automatically, without taking into consideration what types of queries are realistic and interesting to ask.

Table 3 shows the query answering performance broken down by the type of the last atom's predicate, i.e. the triple with the target entity that is ultimately being predicted. 2i queries are not included since they do not have a single last atom, but two atoms with the target entity as the tail. The y-axis shows the Hits@10 performance as a ratio compared to the Hits@10 for that query type in general. The table shows that ip queries are relatively invariant to the last atom's predicate. 2p queries have more variation in this respect, and are not good at predicting drug-protein interactions and drug-drug interactions, which are the relevant relationships for drug target and drug interaction identification.

In figures 4a and 4b the relative performance of queries are broken down by the predicate of the first atom and the predicate of the second atom. Figure 4a shows this data for 2p queries and Figure 4b for 2i queries. The shade indicates the relative Hits@10 performance of queries with the given predicate types as a ratio compared to the Hits@10 for the query type in general.

The main insight from these figures is that the combination of predicate types the query consists of matters to a moderate degree. This can be seen from the degree of variation in the shade of cells in the same row. The most pronounced example is rows 5 and 6 in 4a. The model is good at predicting disease-genetic disorder (DS-GD) and disease-pathway (DS-PA) relationships when the first atom's predicate is a protein-disease (PR-DS) relationship, but not when it is a drug-disease (DR-DS) relationship.

A domain expert could interpret these figures in more detail and perhaps evaluate the query answering performance based on the importance or relevance of different queries.

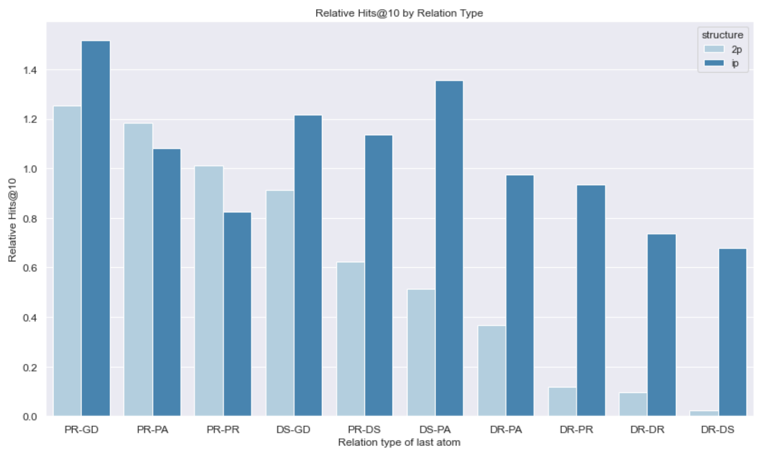
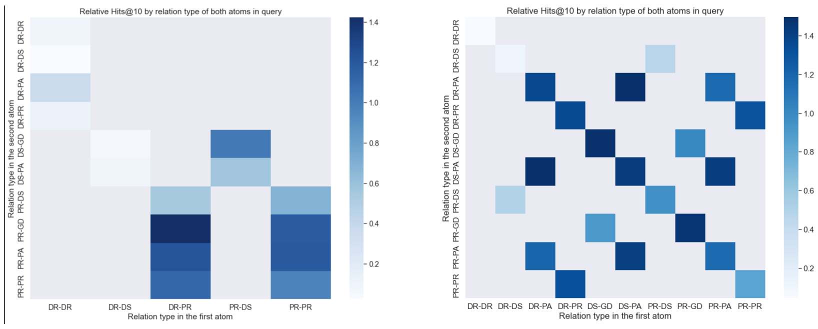


Fig. 3: Relative Hits@10 of complex queries by query type and the predicate of the last atom



(a) Query type = 2p (b) Query type = 2i

Fig. 4: Relative Hits@10 of complex queries by query type and the predicates of the first and second atom

7 Limitations

A recent paper [6] investigates the performace of several link prediction models on two biomedical KGs, including BioKG. Thousands of experiments were performed to investigate the effect different choices regarding training setup, hyperparameters, and other configuration settings have on performance. They found that ComplEx achieved a Hits@10 score of only 1.2% with optimal hyperparameters on BioKG. Their findings call into question the results reported in this paper. The most likely cause of the discrepancy is a flaw in this paper. Many attempts were made to track down the cause of the discrepancy without success, but the query sampler is a more likely source than the query answering codebase, as the former is more of a work in progress and less tested.

Another limitation is that only 3 complex query structures were tested, compared to the 8 tested in the original CQD paper. This is an obvious improvement to include in future work.

Finally, the depth of analysis of the results is limited by a lack of domain expertise. Future work could be multidisciplinary to solve this limitation.

8 Conclusion

Complex query answering was performed with the CQD framework on the BioKG knowledge graph. The accuracy was quite high and comparable to two of the benchmark that CQD has been tested on previously. The BioKG dataset has some fundamental differences than those datasets, so this strong result provides further evidence for the effectiveness of CQD. However, further analysis showed that the performance varied significantly depending on the relationships being predicted in the queries. This suggests that one should not assume the overall performance scores reflect the performance for specific focused use cases. Finally, the results from a different paper using the same model and dataset cast doubts on these results. However, if the results are correct they demonstrate strong potential for CQD to be applied to the biomedical domain, which has positive implications for drug development since complex query answering is so flexible.

References

1. Arakelyan, E., Daza, D., Minervini, P., & Cochez, M. (2020). Complex query answering with neural link predictors. *arXiv preprint arXiv:2011.03459*
2. Walsh, B., Mohamed, S. K., & Nováček, V. (2020, October). BioKG: A knowledge graph for relational learning on biological data. *Proceedings of the 29th ACM International Conference on Information Knowledge Management (pp. 3173-3180)*
3. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., & Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems, 26*
4. Toutanova, K., & Chen, D. (2015, July). Observed versus latent features for knowledge base and text inference. *In Proceedings of the 3rd workshop on continuous vector space models and their compositionality (pp. 57-66).*
5. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E. R., & Mitchell, T. M. (2010, July). Toward an architecture for never-ending language learning. In Twenty-Fourth AAAI conference on artificial intelligence.
6. Bonner, S., Barrett, I. P., Ye, C., Swiers, R., Engkvist, O., Hoyt, C. T., & Hamilton, W. L. (2022). Understanding the performance of knowledge graph embeddings in drug discovery. Artificial Intelligence in the Life Sciences, 100036.
7. Breit, A., Ott, S., Agibetov, A., & Samwald, M. (2020). OpenBioLink: a benchmarking framework for large-scale biomedical link prediction. Bioinformatics, 36(13), 4097-4098.
8. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., & Bouchard, G. (2016, June). Complex embeddings for simple link prediction. In International conference on machine learning (pp. 2071-2080). PMLR.

9. Liu, Y., Hildebrandt, M., Joblin, M., Ringsquandl, M., Raissouni, R., & Tresp, V. (2021, June). Neural multi-hop reasoning with logical rules on biomedical knowledge graphs. In European Semantic Web Conference (pp. 375-391). Springer, Cham.
10. Alivanistos, D., Berrendorf, M., Cochez, M., & Galkin, M. (2021). Query Embedding on Hyper-relational Knowledge Graphs. arXiv preprint arXiv:2106.08166.
11. Nickel, M., Murphy, K., Tresp, V., & Gabrilovich, E. (2015). A review of relational machine learning for knowledge graphs. Proceedings of the IEEE, 104(1), 11-33.
12. Wang, Q., Mao, Z., Wang, B., & Guo, L. (2017). Knowledge graph embedding: A survey of approaches and applications. IEEE Transactions on Knowledge and Data Engineering, 29(12), 2724-2743.
13. Nickel, M., Tresp, V., Kriegel, H. P. (2011, January). A three-way model for collective learning on multi-relational data. In Icml.
14. Gene Ontology Consortium. (2006). The gene ontology (GO) project in 2006. Nucleic acids research, 34(suppl_1), D322-D326.
15. UniProt Consortium. (2019). UniProt: a worldwide hub of protein knowledge. Nucleic acids research, 47(D1), D506-D515.
16. Knox, C., Law, V., Jewison, T., Liu, P., Ly, S., Frolkis, A., ... & Wishart, D. S. (2010). DrugBank 3.0: a comprehensive resource for ‘omics’ research on drugs. Nucleic acids research, 39(suppl_1), D1035-D1041.
17. Ali, M., Berrendorf, M., Hoyt, C. T., Vermue, L., Galkin, M., Sharifzadeh, S., ... & Lehmann, J. (2021). Bringing light into the dark: A large-scale evaluation of knowledge graph embedding models under a unified framework. IEEE Transactions on Pattern Analysis and Machine Intelligence.
18. Ye, Q., Hsieh, C. Y., Yang, Z., Kang, Y., Chen, J., Cao, D., ... & Hou, T. (2021). A unified drug–target interaction prediction framework based on knowledge graph and recommendation system. Nature communications, 12(1), 1-12.
19. Morgan, P., Brown, D. G., Lennard, S., Anderton, M. J., Barrett, J. C., Eriksson, U., ... & Pangalos, M. N. (2018). Impact of a five-dimensional framework on RD productivity at AstraZeneca. Nature reviews Drug discovery, 17(3), 167-181.
20. Zeng, X., Tu, X., Liu, Y., Fu, X., & Su, Y. (2022). Toward better drug discovery with knowledge graph. Current opinion in structural biology, 72, 114-126.
21. Ye, Q., Hsieh, C. Y., Yang, Z., Kang, Y., Chen, J., Cao, D., ... & Hou, T. (2021). Toward better drug discovery with knowledge graph. Current opinion in structural biology, 72, 114-126.