Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22)
Sister Conferences Best Papers Track

Complex Query Answering with Neural Link Predictors (Extended Abstract)*

Pasquale Minervini[1†], **Erik Arakelyan**[1†], **Daniel Daza**[2,3,4†], **Michael Cochez**[2,4]
[1]UCL Centre for Artificial Intelligence, University College London, United Kingdom
[2]Vrije Universiteit Amsterdam, The Netherlands
[3]University of Amsterdam, The Netherlands
[4]Discovery Lab, Elsevier, The Netherlands
{p.minervini,erik.arakelyan.18}@ucl.ac.uk
{d.dazacruz,m.cochez}@vu.nl

Abstract

Neural link predictors are useful for identifying missing edges in large scale Knowledge Graphs. However, it is still not clear how to use these models for answering more complex queries containing logical conjunctions ($\wedge$), disjunctions ($\vee$), and existential quantifiers ($\exists$). We propose a framework for efficiently answering complex queries on incomplete Knowledge Graphs. We translate each query into an end-to-end differentiable objective, where the truth value of each atom is computed by a pre-trained neural link predictor. We then analyse two solutions to the optimisation problem, including gradient-based and combinatorial search. In our experiments, the proposed approach produces more accurate results than state-of-the-art methods — black-box models trained on millions of generated queries — without the need for training on a large and diverse set of complex queries. Using orders of magnitude less training data, we obtain relative improvements ranging from 8% up to 40% in Hits@3 across multiple knowledge graphs. We find that it is possible to *explain* the outcome of our model in terms of the intermediate solutions identified for each of the complex query atoms. All our source code and datasets are available online [1].

1 Introduction

Knowledge Graphs (KGs) are graph-structured knowledge bases, where knowledge about the world is stored in the form of relationship between entities. KGs are an extremely flexible and versatile knowledge representation formalism – examples include general purpose knowledge bases such as DBpedia [Auer *et al.*, 2007] and YAGO [Suchanek *et al.*, 2007], domain-specific ones such as Bio2RDF [Dumontier *et al.*, 2014] and Hetionet [Himmelstein *et al.*, 2017] for life sciences and WordNet [Miller, 1992] for linguistics, and application-driven graphs such as the Google Knowledge Graph, Microsoft's Bing Knowledge Graph, and Facebook's Social Graph [Noy *et al.*, 2019].

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

In this work, we propose Complex Query Decomposition (CQD) – a framework for answering First-Order Logic Queries, where the query is compiled in an end-to-end differentiable function, modelling the interactions between its atoms. The truth value of each atom is computed by a neural link predictor [Nickel *et al.*, 2016] – a differentiable model that, given an atomic query, returns the likelihood that the fact it represents holds true. We then propose two approaches for identifying the most likely values for the variable nodes in a query – either by continuous or by combinatorial optimisation.

Recent work on embedding logical queries on KGs [Hamilton *et al.*, 2018; Daza and Cochez, 2020; Ren *et al.*, 2020] has suggested that in order to go beyond link prediction, more elaborate architectures, and a large and diverse dataset with millions of queries is required. In this work, we show that this is not the case, and demonstrate that it is possible to use an efficient neural link predictor trained for 1-hop query answering, to generalise to up to 8 complex query structures. By doing so, we produce more accurate results than state-of-the-art models, while using orders of magnitude less training data.

Summarising, in comparison with other approaches in the literature such as Query2Box [Ren *et al.*, 2020] and BetaE [Ren and Leskovec, 2020], we find that the proposed framework i) achieves significantly better or equivalent predictive accuracy on a wide range of complex queries, ii) is capable of out-of-distribution generalisation, since it is trained on simple queries only and evaluated on complex queries, and iii) is more explainable, since the intermediate results for its

*This is an extended abstract of Arakelyan *et al.* [2021], published in the proceedings of the 9th International Conference on Learning Representations (ICLR 2021), where it was awarded an *Outstanding Paper Award*.

[1]At https://github.com/uclnlp/cqd

[†]Equal contribution, alphabetical order.

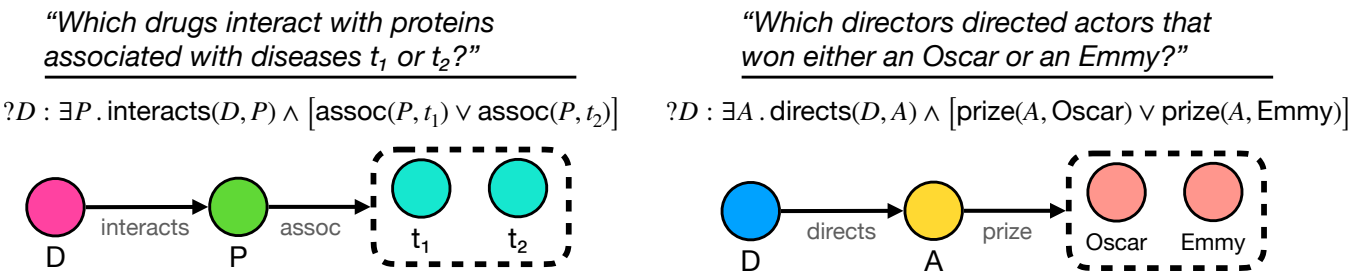

Figure 1: Examples of First-Order Logical Queries using existential quantification ($\exists$), conjunction ($\wedge$), and disjunction ($\vee$) operators — their dependency graphs are $D \leftarrow P \leftarrow \{t_1, t_2\}$, and $D \leftarrow A \leftarrow \{\text{Oscar}, \text{Emmty}\}$, respectively.

sub-queries and variable assignments can be used to explain any given answer.

2 Existential Positive First-Order Logical Queries

A Knowledge Graph $\mathcal{G} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ can be defined as a set of subject-predicate-object $\langle s, p, o \rangle$ 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. One can think of a Knowledge Graph as a labelled multi-graph, where entities $\mathcal{E}$ represent nodes, and edges are labelled with relation types $\mathcal{R}$. Without loss of generality, a Knowledge Graph can be represented as a First-Order Logic Knowledge Base, where each triple $\langle s, p, o \rangle$ denotes an atomic formula $p(s, o)$, with $p \in \mathcal{R}$ a binary predicate and $s, o \in \mathcal{E}$ its arguments.

Conjunctive queries are a sub-class of First-Order Logical queries that use existential quantification ($\exists$) and conjunction ($\wedge$) operations. We consider conjunctive queries $\mathcal{Q}$ in the following form:

$$\begin{aligned} \mathcal{Q}[A] \triangleq ?A : \quad & \exists V_1, \ldots, V_m . e_1 \wedge \ldots \wedge e_n \\ & \text{where } e_i = p(c, V), \\ & \quad \text{with } V \in \{A, V_1, \ldots, V_m\}, c \in \mathcal{E}, p \in \mathcal{R}, \\ & \text{or } e_i = p(V, V'), \\ & \quad \text{with } V, V' \in \{A, V_1, \ldots, V_m\}, V \neq V', p \in \mathcal{R}. \end{aligned} \tag{1}$$

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*. Each e_i denotes a logical atom, with either one ($p(c, V)$) or two variables ($p(V, V')$), and $e_1 \wedge \ldots \wedge e_n$ denotes a conjunction between n atoms.

The goal of answering the logical query $\mathcal{Q}$ consists in finding a set of entities $[\![\mathcal{Q}]\!] \subseteq \mathcal{E}$ such that $a \in [\![\mathcal{Q}]\!]$ iff $\mathcal{Q}[a]$ holds true, where $[\![\mathcal{Q}]\!]$ is the *answer set* of the query $\mathcal{Q}$.

As illustrated in Figure 1, the *dependency graph* of a conjunctive query $\mathcal{Q}$ is a graph representation of $\mathcal{Q}$ where nodes correspond to variable or non-variable atom arguments in $\mathcal{Q}$ and edges correspond to atom predicates. We follow Hamilton *et al.* [2018] and focus on *valid* conjunctive queries – i.e. the dependency graph needs to be a directed acyclic graph, where anchor entities correspond to source nodes, and the query target A is the unique sink node.

Handling Disjunctions Our aim is answering a wider class of logical queries, namely Existential Positive First-Order (EPFO) queries [Dalvi and Suciu, 2004] that in addition to existential quantification and conjunction, also involve disjunction ($\vee$). We follow Ren *et al.* [2020] by transforming a given EPFO query into Disjunctive Normal Form [DNF, Davey and Priestley, 2002], i.e. a disjunction of conjunctive queries. See Figure 1 for an example.

3 Complex Query Answering via Optimisation

We propose Complex Query Decomposition (CQD) as a framework for answering EPFO logical queries in the presence of missing edges. Given a query $\mathcal{Q}$, we define 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 $\mathcal{Q}$, given a variable-to-entity substitution for all variables in $\mathcal{Q}$.

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 $\mathcal{Q}[a]$ is computed by aggregating all atom scores using t-norms and t-conorms – continuous relaxations of the logical conjunction and disjunction operators.

Neural Link Prediction A neural link predictor is a differentiable model where atom arguments are first mapped into a k-dimensional embedding space, and then used for producing 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 $\phi_p(\mathbf{e}_s, \mathbf{e}_o)$, where $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^k$ are the embedding vectors of s and o, and $\phi_p : \mathbb{R}^k \times \mathbb{R}^k \mapsto [0, 1]$ is a *scoring function* computing the likelihood that entities s and o are related by the relationship p.

In our experiments, as neural link predictor, we use a regularised variant of ComplEx [Trouillon *et al.*, 2016; Lacroix *et al.*, 2018].

T-Norms A *t-norm* $\top : [0, 1] \times [0, 1] \mapsto [0, 1]$ is a generalisation of conjunction in logic [Klement *et al.*, 2000, 2004]. Some examples include the *Gödel t-norm* $\top_{\min}(x, y) = \min\{x, y\}$, the *product t-norm* $\top_{\text{prod}}(x, y) = x \cdot y$, and the *Łukasiewicz t-norm* $\top_{\text{Luk}}(x, y) = \max\{0, x + y - 1\}$. Analogously, *t-conorms* are dual to t-norms for disjunctions –

given a t-norm $\top$, the complementary t-conorm is defined by $\bot(x, y) = 1 - \top(1 - x, 1 - y)$.

Continuous Reformulation of Complex Queries Let $\mathcal{Q}$ denote the following DNF query:

$$\begin{aligned}\mathcal{Q}[A] \triangleq ?A : \exists V_1, \ldots, V_m. \left(e_1^1 \wedge \ldots \wedge e_{n_1}^1\right) \vee \\ \ldots \vee \left(e_1^d \wedge \ldots \wedge e_{n_d}^d\right),\end{aligned}$$

where $e_i^j = p(c, V)$,
with $V \in \{A, V_1, \ldots, V_m\}, c \in \mathcal{E}, p \in \mathcal{R}$,
or $e_i^j = p(V, V')$,
with $V, V' \in \{A, V_1, \ldots, V_m\}, V \neq V', p \in \mathcal{R}$.

We want to know the variable assignments that render $\mathcal{Q}$ true. To achieve this, we can cast this as an optimisation problem, where the aim is finding a mapping from variables to entities that *maximises* the score of $\mathcal{Q}$:

$$\arg\max_{A, V_1, \ldots, V_m \in \mathcal{E}} \left(e_1^1 \top \ldots \top e_{n_1}^1\right) \bot \ldots \bot \left(e_1^d \top \ldots \top e_{n_d}^d\right) \quad (2)$$

where $e_i^j = \phi_p(\mathbf{e}_c, \mathbf{e}_V)$,
with $V \in \{A, V_1, \ldots, V_m\}, c \in \mathcal{E}, p \in \mathcal{R}$
or $e_i^j = \phi_p(\mathbf{e}_V, \mathbf{e}_{V'})$,
with $V, V' \in \{A, V_1, \ldots, V_m\}, V \neq V', p \in \mathcal{R}$,

where $\top$ and $\bot$ denote a t-norm and a t-conorm – a continuous generalisation of the logical conjunction and disjunction, respectively – and $\phi_p(\mathbf{e}_s, \mathbf{e}_o) \in [0, 1]$ denotes the neural link prediction score for the atom $p(s, o)$. We write t-norms and t-conorms as infix operators since they are both associative.

Note that, in Equation (2), the bound variable nodes $V_1, \ldots, V_m$ are only used through their embedding vector: to compute $\phi_p(\mathbf{e}_c, \mathbf{e}_V)$ we only use the embedding representation $\mathbf{e}_V \in \mathbb{R}^k$ of V, and do not need to know which entity the variable V corresponds to. This means that we have two possible strategies for finding the optimal variable embeddings $\mathbf{e}_V \in \mathbb{R}^k$ with $V \in \{A, V_1, \ldots, V_m\}$ for maximising the objective in Equation (2), namely *continuous optimisation*, where we optimise $\mathbf{e}_V$ using gradient-based optimisation, and *combinatorial optimisation*, where we search for the optimal variable-to-entity assignment.

3.1 Complex Query Answering via Continuous Optimisation

One way we can solve the optimisation problem in Equation (2) is by finding the variable embeddings that maximise the score of a complex query. Equation (2) is modified into the following continuous optimisation problem:

$$\arg\max_{\mathbf{e}_A, \mathbf{e}_{V_1}, \ldots, \mathbf{e}_{V_m} \in \mathbb{R}^k} \left(e_1^1 \top \ldots \top e_{n_1}^1\right) \bot .. \bot \left(e_1^d \top \ldots \top e_{n_d}^d\right) \quad (3)$$

In Equation (3) we directly optimise the embedding representations $\mathbf{e}_A, \mathbf{e}_{V_1}, \ldots, \mathbf{e}_{V_m} \in \mathbb{R}^k$ of variables $A, V_1, \ldots, V_m$, rather than exploring the combinatorial space of variable-to-entity mappings. In this way, we can tackle the maximisation problem in Equation (3) using gradient-based optimisation methods, such as Adam [Kingma and Ba, 2015]. Then, after we identified the optimal representation for variables $A, V_1, \ldots, V_m$, we replace the query target embedding $\mathbf{e}_A$ with the embedding representations $\mathbf{e}_c \in \mathbb{R}^k$ of all entities $c \in \mathcal{E}$, and use the resulting complex query score to compute the likelihood that such entities answer the query. We denote this variant of CQD as CQD-CO.

3.2 Complex Query Answering via Combinatorial Optimisation

Another way we tackle the optimisation problem in Equation (2) is 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 $\mathcal{Q}$ 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 $\phi_p(\mathbf{e}_c, \mathbf{e}_t)$ – i.e. the most likely entities to appear as a substitution of V according to the neural link predictor.

Our procedure is akin to beam search: as we traverse the dependency graph of a query, we keep a beam with the most promising variable-to-entity substitutions identified so far. We thus denote this variant as CDQ-Beam. Note that scoring all possible entities can be done efficiently and in a single step on a GPU by replacing V with the entity embedding matrix.

4 Experiments

Datasets Following Ren *et al.* [2020], we evaluate our approach on FB15k [Bordes *et al.*, 2013] and FB15k-237 [Toutanova and Chen, 2015] – two subset of the Freebase knowledge graph – and NELL995 [Xiong *et al.*, 2017], a KG generated by the NELL system [Mitchell *et al.*, 2015]. We use the datasets generated by Ren and Leskovec [2020], who notice that in previous benchmarks, test queries may have more than 5,000 answers. To make the task more challenging, they regenerate the same number of validation and test queries for each of the 9 query structures, keeping only those with a number of answer smaller than a threshold. We consider a total of 9 query types, including atomic queries, and 2 query types that contain disjunctions – the different query types are shown in Figure 2. In our framework, the neural link predictor is only trained on atomic queries, while the evaluation is carried out on the complete set of query types in Figure 2.

4.1 Results

We detail the results of H@3 for all different query types in Table 1. We observe that, on average, both CQD-CO and CQD-Beam produce more accurate results than GQE and Q2B, while using orders of magnitude less training data. In particular, combinatorial optimisation in CQD-Beam outperforms the baselines across all datasets.

The results for chained queries (*2p* and *3p*) show that CQD-Beam is effective, even when increasing the length of the chain. The most difficult case corresponds to 3p queries,

Method	Avg	1p	2p	3p	2i	3i	pi	ip	2u	up
FB15k										
BetaE	0.416	0.651	0.257	0.247	0.558	0.665	0.439	0.281	0.401	0.252
Query2Box	0.380	0.680	0.210	0.142	0.551	0.665	0.394	0.261	0.351	0.167
GQE	0.280	0.546	0.153	0.108	0.397	0.514	0.276	0.191	0.221	0.116
CQD-CO	0.469	**0.892**	0.256	0.136	**0.774**	0.783	0.442	0.332	0.417	0.221
CQD-Beam	**0.684**	**0.892**	**0.653**	**0.297**	**0.771**	**0.806**	**0.706**	**0.716**	**0.723**	**0.594**
FB15k-237										
BetaE	0.209	0.390	0.109	**0.100**	0.288	0.425	0.224	0.126	0.124	0.097
Query2Box	0.201	0.406	0.094	0.068	0.295	0.423	0.212	0.126	0.113	0.076
GQE	0.163	0.350	0.072	0.053	0.233	0.346	0.165	0.107	0.082	0.057
CQD-CO	0.219	**0.467**	0.096	0.062	0.312	0.406	0.236	0.160	0.145	0.082
CQD-Beam	**0.253**	**0.467**	**0.133**	0.079	**0.349**	**0.486**	**0.271**	**0.204**	**0.176**	**0.115**
NELL995										
BetaE	0.246	0.530	0.130	0.114	0.376	0.475	0.241	0.143	0.122	0.086
Query2Box	0.229	0.422	0.140	0.112	0.333	0.445	0.224	0.168	0.113	0.103
GQE	0.186	0.328	0.119	0.096	0.275	0.352	0.184	0.144	0.085	0.088
CQD-CO	0.288	**0.604**	0.178	0.128	0.393	0.466	0.301	0.221	0.173	0.132
CQD-Beam	**0.318**	**0.604**	**0.226**	**0.136**	**0.436**	**0.530**	**0.312**	**0.256**	**0.199**	**0.167**

Table 1: Complex query answering results (MRR) across all query types in the datasets from Ren and Leskovec [2020]; results for GQE [Hamilton *et al.*, 2018], Query2Box [Ren *et al.*, 2020], and BetaE [Ren and Leskovec, 2020] are from Ren and Leskovec [2020].

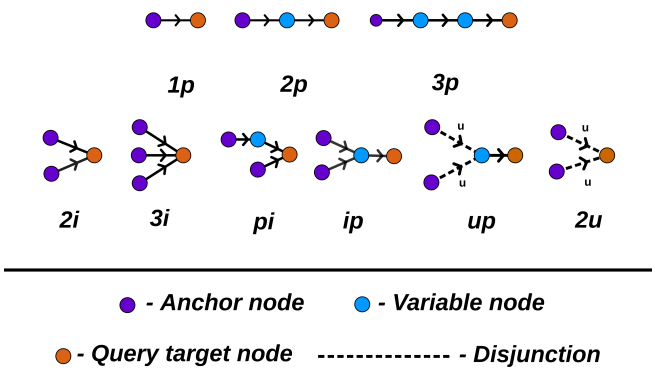

Figure 2: Query structures considered in our experiments, as proposed by Ren *et al.* [2020] – the naming of each query structure corresponds to *projection* (**p**), *intersection* (**i**), and *union* (**u**), and reflects how they were implemented in the Query2Box model [Ren *et al.*, 2020]. An example of a **pi** query is $?T : \exists V.p(a, V), q(V, T), r(b, T)$, where a and b are anchor nodes, V is a variable node, and T is the query target node.

where the number of candidate variable substitutions increases due to the branching factor of the search procedure.

Explaining Answers to Complex Queries A useful property of our framework is its transparency when computing scores for distinct atoms in a query. Unlike GQE and Q2B – two neural models that encode a query into a vector via a set of non-linear transformations – CQD-Beam is able to produce an explanation for a given answer in terms of intermediate variable assignments as it selects a top-k list of candidates for each atom. By inspecting these decisions we can thus identify failure modes of our framework, even when it produces seemingly correct answers. This is in contrast with previously proposed black-box models for query answering, where such an analysis is not possible.

5 Conclusions

We proposed a framework — Complex Query Decomposition (CQD) — for answering Existential Positive First-Order logical queries by reasoning over sets of entities in embedding space. In our framework, answering a complex query is reduced to answering each of its sub-queries, and aggregating the resulting scores via t-norms. The benefit of the method is that we only need to train a neural link prediction model on atomic queries to use our framework for answering a given complex query, without the need of training on millions of generated complex queries. This comes with the added value that we are able to explain each step of the query answering process regardless of query complexity, instead of using a black-box neural query embedding model.

The proposed method is agnostic to the type of query, and is able to generalise without explicitly training on a specific variety of queries. Experimental results show that CQD produces significantly more accurate results than current state-of-the-art complex query answering methods on incomplete Knowledge Graphs.

References

Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. Complex query answering with neural link predictors. In *9th International Conference on Learn-*

ing Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net, 2021.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. DBpedia: A nucleus for a web of open data. In *ISWC/ASWC*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer, 2007.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795, 2013.

Nilesh N. Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, pages 864–875. Morgan Kaufmann, 2004.

Brian A. Davey and Hilary A. Priestley. *Introduction to Lattices and Order, Second Edition*. Cambridge University Press, 2002.

Daniel Daza and Michael Cochez. Message passing query embedding. In *ICML Workshop - Graph Representation Learning and Beyond*, 2020.

Michel Dumontier, Alison Callahan, Jose Cruz-Toledo, Peter Ansell, Vincent Emonet, François Belleau, and Arnaud Droit. Bio2RDF release 3: A larger, more connected network of linked data for the life sciences. In *International Semantic Web Conference (Posters & Demos)*, volume 1272 of *CEUR Workshop Proceedings*, pages 401–404. CEUR-WS.org, 2014.

William L. Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. Embedding logical queries on knowledge graphs. In *NeurIPS*, pages 2030–2041, 2018.

Daniel S. Himmelstein, Antoine Lizee, Christine Hessler, Leo Brueggeman, Sabrina L. Chen, Dexter Hadley, Ari Green, Pouya Khankhanian, and Sergio E. Baranzini. Systematic integration of biomedical knowledge prioritizes drugs for repurposing. *bioRxiv*, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

Erich-Peter Klement, Radko Mesiar, and Endre Pap. *Triangular Norms*, volume 8 of *Trends in Logic*. Springer, 2000.

Erich-Peter Klement, Radko Mesiar, and Endre Pap. Triangular norms. position paper I: basic analytical and algebraic properties. *Fuzzy Sets Syst.*, 143(1):5–26, 2004.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. Canonical tensor decomposition for knowledge base completion. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 2869–2878. PMLR, 2018.

George A. Miller. WORDNET: a lexical database for english. In *HLT*. Morgan Kaufmann, 1992.

Tom M. Mitchell, William W. Cohen, Estevam R. Hruschka Jr., Partha Pratim Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapandula Nakashole, Emmanouil A. Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard C. Wang, Derry Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. Never-ending learning. In *AAAI*, pages 2302–2310. AAAI Press, 2015.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016.

Natalya Fridman Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. Industry-scale knowledge graphs: lessons and challenges. *Commun. ACM*, 62(8):36–43, 2019.

Hongyu Ren and Jure Leskovec. Beta embeddings for multi-hop logical reasoning in knowledge graphs. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Hongyu Ren, Weihua Hu, and Jure Leskovec. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *WWW*, pages 697–706. ACM, 2007.

Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, Beijing, China, July 2015. Association for Computational Linguistics.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2071–2080. JMLR.org, 2016.

Wenhan Xiong, Thien Hoang, and William Yang Wang. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *EMNLP*, pages 564–573. Association for Computational Linguistics, 2017.