Query answering over the polymorphic web of data

Cosimo Gregucci[⋆]

University of Stuttgart, Stuttgart, Germany
cosimo.gregucci@ipvs.uni-stuttgart.de

Abstract. Knowledge graphs are a versatile means to gather Semantic Web data and are typically stored and queried with the W3C standards, RDF and SPARQL. Despite the significant progress made in query processing, predicting plausible answers in presence of missing facts remains a challenge. This aspect has been tackled by proposing methods to predict links and solve queries in some reduced fragments of SPARQL. Thus far, I have explored two parallel directions for this thesis. First, I study how to use knowledge graph embedding methods to predict missing facts. In particular, I explore how to combine different knowledge graph embedding methods to improve the quality of the predictions. Second, I study how to connect techniques to query knowledge graphs from these two research areas, namely database query processing, and graph learning. The former techniques provide actual answers of a query, while the latter provide plausible ones. My hypothesis is that I can define a common query interface, based on SPARQL, to provide answers from these polymorphic data sources. To this end, I propose an extension for SPARQL, called polymorphic SPARQL.

Keywords: link prediction · ensemble · query answering · polymorphic

1 Introduction

The emergence of large knowledge graphs (KGs) stored and queried using the W3C standards RDF [5], and SPARQL [9], has encouraged many researchers to improve query processing over KGs [10]. SPARQL is a database query language and, as such, the result of a SPARQL query over a given dataset is defined based on the data that is in the dataset. Thus, given that knowledge graphs are known to be incomplete, in presence of missing links, some answers might not be retrieved by the SPARQL engine.

Such aspect has been addressed by the graph learning community, who identify two problems: 1) predicting missing links, which is known as *link prediction*, and 2) predicting plausible answers of first-order logic query, which is known as *complex query answering*.

According to the graph learning community, a KG is a triple $(\mathcal{V}, \mathcal{R}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{R}$ are two finite sets, whose elements are called *entities* and *relation*

[⋆] Category: Early Stage PhD

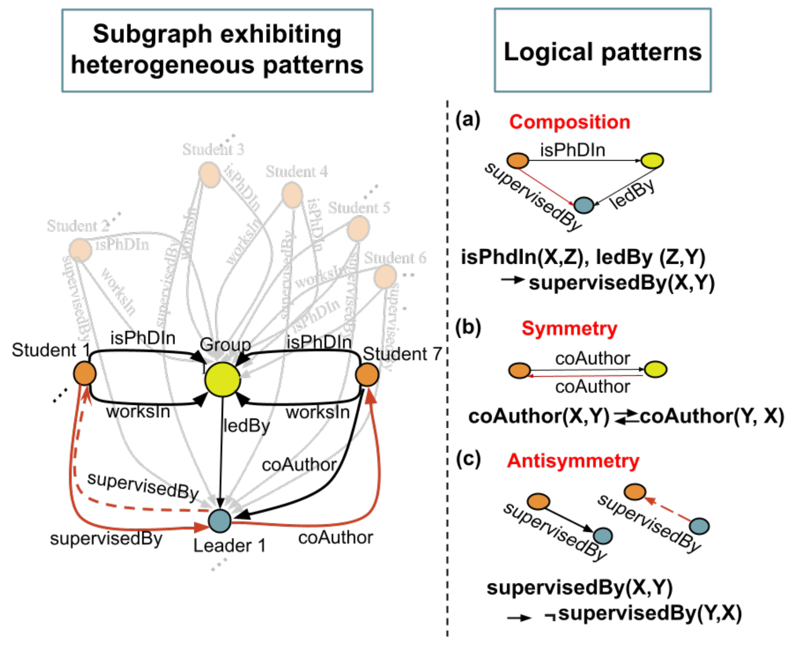


Fig. 1: A sub-graph that exhibits heterogeneous relational patterns [15].

names, and $\mathcal{E}$ is a subset of $\mathcal{V} \times \mathcal{R} \times \mathcal{V}$ that represent *relationships between entities*. Relationships are triples (h, r, t) where h and t are called the head and the tail entities.

Like complex query answering, link prediction can be seen as answering first-order logic queries. Indeed, the link prediction problems of finding sensible tails for a given pair of head and relation name, and finding sensible heads for a given pair of relation name and tail, denoted $(h, r, ?t)$ or $(?h, r, t)$, correspond to the respective first order queries $(x).r(h, x)$ and $(x).r(x, t)$. Since SPARQL has the same expressive power as relational calculus [1], a safe subset of first-order logic queries, link prediction and complex query answering methods can be used to return plausible SPARQL answers.

Knowledge graph embeddings (KGEs) are a prominent approach for both problems, link prediction [3, 18, 20], and complex query answering [11, 16, 17]. Relations in the graph may follow patterns (Figure 1) that can be learned. For example, some relations might be symmetric and others might be hierarchical. However, the existing approaches have two limitations:

L1. The learning capability of different knowledge graph embedding methods varies for each pattern and, so far, no single method can learn all patterns equally well (see Table 1).

L2. Both complex query answering and link prediction methods only support a limited subset of SPARQL. Indeed, existing link prediction methods are restricted to the atomic queries $(x).r(h, x)$ and $(x).r(x, t)$, and complex query

Table 1: Specification of query representation of baseline and state of the art KGE models and respective pattern modeling and inference abilities. $\circ$ is element-wise complex product together with relation normalization. S = Symmetry, A = Antisymmetry, I = Inversion, C = Composition, H = Hierarchy

Model	Query	Embeddings	S	A	I	C	H
TransE [3]	$\boldsymbol{q} = \boldsymbol{h} + \boldsymbol{r}$	$\boldsymbol{q}, \boldsymbol{h}, \boldsymbol{r} \in \mathbb{R}^d$	$\times$	$\checkmark - \mathbf{0}$	$\checkmark - \mathbf{0}$	$\checkmark - \mathbf{0}$	$\times$
RotatE [18]	$\boldsymbol{q} = \boldsymbol{h} \circ \boldsymbol{r}$	$\boldsymbol{q}, \boldsymbol{h}, \boldsymbol{r} \in \mathbb{C}^d$	$\checkmark - \mathbf{2}$	$\checkmark - \mathbf{2}$	$\checkmark - \mathbf{2}$	$\checkmark - \mathbf{2}$	$\times$
ComplEx [20]	$\boldsymbol{q} = \boldsymbol{h} \times \boldsymbol{r}$	$\boldsymbol{q}, \boldsymbol{h}, \boldsymbol{r} \in \mathbb{C}^d$	$\checkmark - \mathbf{2}$	$\checkmark - \mathbf{2}$	$\checkmark - \mathbf{2}$	$\times$	$\times$
DistMult [28]	$\boldsymbol{q} = \boldsymbol{h} \cdot \boldsymbol{r}$	$\boldsymbol{q}, \boldsymbol{h}, \boldsymbol{r} \in \mathbb{R}^d$	$\checkmark - \mathbf{0}$	$\times$	$\times$	$\times$	$\times$
RefH [4]	$\boldsymbol{q} = \boldsymbol{Ref(\theta_r)h}$	$\boldsymbol{q}, \boldsymbol{h} \in \mathbb{H}^d$	$\checkmark - \mathbf{0}$	$\times$	$\times$	$\times$	$\checkmark - \mathbf{0}$

answering methods are restricted to queries with constants (called *anchors*), returning a single variable, or having no cycles.

In this thesis, I address these two limitations. Regarding **L1**, I developed a method to combine different knowledge graph embedding methods using attention, and showed that the combined method can outperform the individual ones. I published a paper [7] showing these results, and I am currently working on improvements to this idea. Regarding **L2**, I am studying how to integrate SPARQL services with link prediction services to return both, actual answers from explicit links and plausible answers from predicted links. According to [16] completing the knowledge graph with the *predicted* links before executing queries would result in a graph that is too dense, thus I plan to retrieve the predictions at query runtime. Moreover, by completing the graph, the predicted links would be indistinguishable from the actual ones. To this end, I will define a common interface, I called *polymorphic SPARQL* (*p-SPARQL*), on top of both interfaces.

2 Related work

This section presents the work related to limitations **L1** and **L2**.

2.1 Combination of knowledge graph embeddings

Xu et al. [27] show that the combination of multiple runs of low-dimensional embedding models can outperform the corresponding individual high-dimensional embedding model. Unlike my approach [7], they do not combine different methods to combine the different learning capabilities.

Most methods combining different KGEs train models separately and then combine their scoring function [13, 21, 23, 24]. Unlike these approaches, my approach combines the multiple vector representations of a query and uses attention to select, for each query, the best suited representations.

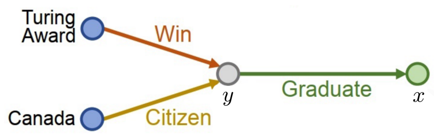
I combine multiple vector representations to increase the learning capabilities of individual methods. Another way to increase the learning capability of a method is to embed queries into spaces that combine different geometrical spaces. For example, Gu et al. [8] combines Hyperbolic, Spherical, and Euclidean spaces, and UltraE [25] uses an Ultra-hyperbolic manifold, which generalizes Hyperbolic and Spherical manifolds allowing to simultaneous embed multiple distinct hierarchical relations and non-hierarchical ones in a single *heterogeneous* geometric space. So far, I have studied how to combine query vectors rather than geometric spaces [7]. My proposed method combines vectors in Euclidean space, and then projects them to non-Euclidean manifolds to learn hierarchical relations; it is not limited to a single geometry, and I will adapt it to combine vector representations beyond the Euclidean space.

2.2 Including link prediction in SPARQL queries

Complex query answering Most approaches for complex query answering map queries into probability distributions [17], or regions [11, 16] in the vector space. They map first-order logic queries to directed acyclic graphs (DAGs). They are also called *dependency graphs*, and their nodes are the constants and variables in the query. For each atom $r(u, v)$ a directed edge between u and v with label r is in the dependency graph. The direction of each edge is chosen conveniently to guarantee that the dependency graph have a unique sink node(see Figure 2). The dependency graph is used to define the operations that are performed to get to the representation of the sink.

As we already mentioned, existing query answering methods do not support every first-order query, even for reduced first-order logic fragments, such as conjunctive queries.

The limitations are that queries must return a single variable (which is the sink), have no cycles, and all local sources (i.e., nodes with no entering edges) are constants. The following conjunctive query violates all these conditions, and hence cannot be solved by the existing methods: $(x, y).\exists z \exists u(p(x, y) \wedge q(y, z) \wedge r(z, x) \wedge s(u, x))$. Unlike the existing methods, the interface I will define, called p-SPARQL, will support all basic graph patterns by combining standard SPARQL interfaces with link-prediction services.



$(x).(\exists y.(\mathsf{Win}(\mathsf{TuringAdward}, y) \wedge \mathsf{Citizen}(\mathsf{Canada}, y) \wedge \mathsf{Graduate}(y, x))$

Fig. 2: Dependency graph of a first-order query asking *"Where did Canadian citizens with Turing Award graduate?"* [16].

Including similarity metrics in SPARQL query Several works propose extending SPARQL with similarity functions [12, 14].

However, the aforementioned approaches compute similarity among entities, while in link prediction a similarity metric is computed to match a query to a candidate answer.

3 Problem statement and contributions

In this thesis I will study 1) how to combine knowledge graph embeddings, and 2) how to include link prediction in SPARQL queries.

How to combine knowledge graph embeddings. In this thesis, I study how to improve the link-prediction task by combining query vectors, computed with different KGE methods.

Let $\mathbb{M}$ be a set of several existing KGE methods such as TransE [3], RotatE [18], ComplEx [20], DistMult [28], AttE [4]. For each query $q = (h,r,?)$, let the *query representation set* be $\mathcal{Q} = \{\boldsymbol{q}_m \mid \boldsymbol{q}_m = g_r^m(\boldsymbol{h}), m \in \mathbb{M}\}$ where $\boldsymbol{q}_m = g_r^m(\boldsymbol{h})$ is the query representation for the method m. For example, TransE defines $\boldsymbol{q}_{\text{TransE}} = g_r^{\text{TransE}}(\boldsymbol{h}) = \boldsymbol{h} + \boldsymbol{r}$. Our problem is thus to find a function g_Θ such that the combined method m_C performs better than each method $m \in \mathbb{M}$ separately. I can define a combined query $\boldsymbol{q}_C$, as a function of the query vectors $\boldsymbol{q}_m$, $\boldsymbol{q}_C = g_\Theta(\boldsymbol{q}_{\text{TransE}}, \dots, \boldsymbol{q}_{\text{AttE}})$, where g_Θ is a combination function and Θ is a vector of parameters for the function.

Hypothesis 1. The query vectors can be combined using a relation-specific attention mechanism, where the best-performing models should have the most importance.

The related research questions are:

RQ1. Is attention able to give more importance to the best-performing models?
RQ2. Is the combined model able to outperform the single models in the link prediction task?

How to include link prediction in SPARQL query. In this thesis, we will address the problem of how to compute plausible answers for SPARQL having, as input, a SPARQL service s and a link-prediction service m, both defined over the same knowledge graph G.

For convenience, I will focus on the SPARQL fragment consisting of basic graph patterns whose entities and relation names occur in G, and no variables are allowed in predicate position. This SPARQL fragment corresponds to conjunctive queries, that is, first-order logic queries $(x_1, \dots, x_j).(\exists y_1 \cdots \exists y_z.\varphi)$ where φ is a conjunction of atoms $(p_1(u_1, v_1) \wedge \cdots \wedge p_n(u_n, v_n))$, variables $x_1, \dots, x_j, y_1, \dots, y_z$ are the variables occurring in φ, and sets $\{x_1, \dots, x_j\}$ and $\{y_1, \dots, y_z\}$ are disjoint. As we already pointed out, link-prediction and complex query answering methods can be used to provide plausible answers to a restricted set of these queries.

Given a conjunctive query q^C, we can evaluate q^C in service s to obtain a set, $[\![q^C]\!]_s$, containing all mappings $\mu = (x_1 \mapsto a_1, \ldots, x_n \mapsto a_n)$ that are answers to q^C in G. We will write $\mu_{x \mapsto a}$ to denote a mapping $\{x \mapsto a\}$. On the other hand, a query q^C cannot be evaluated at the link prediction service m.

However, if we consider q^C to be composed by a set of atomic queries q of the form $(x).r(h, x)$ and $(x).r(x, t)$, then, when considering each atomic query q separately, it is possible to directly evaluate q in m. In this case, the result is a set of pairs $(\mu_{x \mapsto u}, \tau_{\mu_{x \mapsto u}})$ where u is an entity of graph G, and τ_u is a score given to mapping $\mu_{x \mapsto u}$.

If we now want to evaluate the atomic query q in both services, m, and s, we will notice a difference in the form of the answers: the service s returns set of answers, while the service m returns a set of pairs.

In a high level, our problem is how to define a sensible semantics that extends the SPARQL semantics to incorporate scores to the answers. To define such extended semantics, called p-SPARQL, desiderata must consider, for example, that every answer $\mu \in [\![q]\!]_s$ must have the higher possible score, since we know that μ is an actual answer. Otherwise, if $\mu \notin [\![q]\!]_s$, it should have a lower score that accounts for the plausibility of the answer.

Returning scores for all possible mappings is impractical in SPARQL because datasets are huge. Instead, given a number $k \geq 0$, we can define a service, called $s+k$, that returns a set, called $[\![q]\!]_{s+k}$, of pairs (μ, τ_μ) where τ_μ is the score given to mapping μ, and $[\![q]\!]_{s+k}$ contains all mappings $\mu \in [\![q]\!]_s$ plus the top k scored answers according to the p-SPARQL semantics I will propose.

When it is possible to evaluate q directly at both services, the definition of $[\![q]\!]_{s+k}$ is trivial: return all answers of $[\![q]\!]_s$ (assigning them the highest score) plus the top k results given by m. In the other cases, the problem is more complex. For example, if query q is $(x, y).p(x, y)$ then we cannot use directly service m because m does not support query q. Instead, we can iterate over every entity h in graph G to answer query $(y).p(h, y)$ to get the scores of solutions $\mu_{y \mapsto t} \in [\![(y).p(h, y)]\!]_{s+k}$. Similarly, we can compute the scores of solutions $\mu_{x \mapsto h} \in [\![(x).p(x, t)]\!]_{s+k}$. This observation leads to the following hypothesis:

Hypothesis 2. The score for a mapping μ can be computed as the combination of the scores of its individual mappings $\mu_{variable \mapsto entity}$.

Hypothesis 3. We can define an efficient algorithm that does not require computing the score of every mapping $\mu_{variable \mapsto entity}$.

For example, given a query $(x, y).(r(x, y) \wedge p(y, c))$, the score of a mapping $\mu = \{x \mapsto a, y \mapsto b\}$ depend on the scores given by m to mappings $\mu_{x \mapsto a}$ and $\mu_{y \mapsto b}$ for the respective queries $(x).r(x, b)$, $(y).p(a, y)$, and $(y).p(y, c)$. If mapping $\mu_{x \mapsto a}$ has a low score, it may not be worth to compute the score of mapping $\mu_{y \mapsto b}$ because the score of the combined mapping μ will anyway be low.

RQ3. How can we combine the scores given to mappings by the link prediction service to obtain sensible scores for the combined mappings?

RQ4. How can we return the answers $\mu \in [\![q]\!]_{s+k}$ efficiently?

4 Research Methodology and Approach

In this section, I describe the approach I am following for solving the problems identified in the previous section.

Combination of query vectors I already mention that a KG can follow different patterns, as shown in Figure 1, and that different KGE methods cannot learn all patterns equally well. Table 1 shows the patterns that a KGE method can learn, along with the number of constraints that it has to impose to do so, for each dimension. For example, RotatE defines transformations as rotations $g_r^{\text{RotatE}}(\boldsymbol{h}) = \boldsymbol{h} \circ \boldsymbol{r}$ in Complex space. In this way, RotatE can enforce both $\boldsymbol{h} \circ \boldsymbol{r} = \boldsymbol{t}, \boldsymbol{t} \circ \boldsymbol{r} = \boldsymbol{h}$ if $\boldsymbol{r^2 = 1}$, and thus, requires two constraints $\boldsymbol{r \neq -1}$ and $\boldsymbol{r \neq 1}$ to express antisymmetrical relations. However, TransE can model antisymmetrical patterns naturally, without imposing any constraint, but it is not able to learn symmetrical patterns.

In [7], we showed that an attention-based combination of KGEs can exploit the advantages of each KGE method used in the combination. The combined query representation is defined as $\boldsymbol{q}_C = \sum \alpha_i \mathbf{q}_i$, where each weight α_i can be computed by using an attention mechanism [4]:

$$\alpha_i = \frac{\exp(g(\boldsymbol{w}\boldsymbol{q}_i)}{\sum_j \exp(g(\boldsymbol{w}\boldsymbol{q}_j)},$$

where i and j identify models in $\mathbb{M}$, and $g(x) = \boldsymbol{wx}$ is a function with trainable parameter $\boldsymbol{w}$. This version of our method is called Spherical Embedding with Attention (SEA).

A variant of this method, called Spherical Embedding with Pointcaré Attention (SEPA), projects the combined query $\boldsymbol{q}_C$ to a non-Euclidean manifold, i.e. the Poincaré ball, via the exponential map $\boldsymbol{q}_C^{\mathcal{M}} = \exp_{\boldsymbol{0}}(\boldsymbol{q}_C)$. The score function is $score(q,a) = d(\boldsymbol{q}_C^{\mathcal{M}} \oplus \boldsymbol{r}, \boldsymbol{a})$, where $\boldsymbol{q}_C^{\mathcal{M}}, \boldsymbol{r}, \boldsymbol{a}$ are points on a manifold $\mathcal{M}$, $\exp_0(.)$ is the exponential map from origin, and $\oplus$ is Mobius addition.

In future work, we plan to combine various manifolds besides combining the query vectors in knowledge graph embedding.

Including link prediction in SPARQL Regarding **RQ3**, if a query involves more than one literal, then it may involve the scores given to more than one fact. If the score of a mapping μ reflects a probability, then we should consider this as the joint probability given to the facts involved into this mapping μ. For example, if $\mu = \{x \mapsto a, y \mapsto b\}$, and the query q is $(x,y).(p(x,y) \wedge r(y,c))$, then the score of μ for query q must reflect the probability of $p(a,b) \wedge p(b,c)$, which is the joint probability of the links $p(a,b)$ and $p(b,c)$. The computation of this joint probability from the marginal probabilities of these links depends on the conditional probabilities between these links, which are related to the scores of mappings μ and $\mu_{y \mapsto b}$ for the queries $(x,y).p(x,y)$ and $(y).r(y,c)$. Assuming independence between the links, we can simplify the score definition. However, this assumption may not capture some patterns satisfied by relations in the graph. Moreover, it could also be possible to use other frameworks for combining scores, such as using t-norm fuzzy logics.

Regarding **RQ4**, if we know that the top k predicted answers have a score over a certain threshold, then we can discard mappings μ such that we can infer that the score of μ is below the threshold. Indeed, if scores are probabilities (where a score 1 is given to actual links), and a marginal probability is below the threshold, then we can infer that the joint probability is also below the threshold. To use this idea, we can index the top answers for queries of the form $(x).\exists y.p(x,y)$ and $(y).\exists x.p(x,y)$ for each relation name p in graph G.

5 Evaluation plan

In this section, I will describe the benchmarks and the metrics that I will use for evaluating the proposed solutions.

Combination of query vectors In [7], I used the following standard benchmarks for the evaluation:

- **Wordnet**: WN18RR [6] is a subset of WN18, which contains a mixture of symmetric and antisymmetric as relational patterns, and hierarchical structural patterns. WN18RR contains 11 relations, 86,835 training triples, and 40,943 entities. Compared to the other datasets in KGE literature, WN18RR is considered sparse;
- **FreeBase**: FB15k-237 [19] is the subset of FB15k from removing leakage of inverse relations [6]. FB15k-237 is less sparse than WN18RR and mainly contains composition patterns. It contains 237 relations, 272,115 triples, and 14,541 entities.
- **NELL**: NELL-995 [26] contains 75,492 entities and 200 relations, having $\sim$ 22% hierarchical relations. I use a subset of NELL-995 with 100% hierarchy, created in [2].

I use the popular ranking metrics [22] namely Mean Reciprocal Rank (MRR), and Hits@k, k = 1,3,10. For future works, we plan to use a similar set of datasets and metrics.

Including link prediction in SPARQL queries Regarding **RQ3**, I plan to evaluate our method using the same datasets used by existing complex query answering methods [11,16], which are created from the ones showed in Section 5 and using ranking-based metrics. Besides, I plan to create an additional dataset composed of general BGP queries, which cannot be answered by existing complex query answering methods. I also plan to evaluate our method with SPARQL query performance benchmarks, after adapting them to the link prediction setting. Regarding **RQ4**, we have two metrics: the execution time of queries, and the space and memory used by p-SPARQL.

6 Results

Our work [7] showed that the approaches described in Section 2.1 are able to outperform the individual models used in the combination. Specifically, the hyperbolic version of our combined model (SEPA) outperforms all baselines in low dimensions while the Euclidean one (SEA) is the best model in high dimensions (**RQ2**). It also showed that the attention mechanism can give more importance to the best models for the specific kind of relation involved in the query (**RQ1**).

7 Conclusions

In this thesis, I want to exploit existing knowledge graph embedding methods, to improve their performance in the link prediction task and enhance their usability.

As a first step, I studied how to combine existing KGE methods to improve their performance in link prediction task, which was shown in [7]. Our approach facilitates the combination of the query representations from a wide range of popular knowledge graph embedding models. In future work, we will combine various manifolds besides combining query embeddings. Additionally, the proposed approach could be applied to other tasks, e.g., use an attention mechanism to combine different multi-hop queries [16, 17].

The second step of this thesis will be to bridge between the two research areas of database query processing and machine learning, by proposing p-SPARQL, which integrates link prediction within SPARQL queries. This would allow using link prediction within more complex queries, enhancing their usability.

Acknowledgments This PhD is part of the KnowGraphs project, receiving funding in the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant (agreement No: 860801). Furthermore, I would like to thank my supervisor Prof. Dr. Steffen Staab, and my co-supervisors Dr. Daniel Hernández and Mojtaba Nayyeri for their continuous support.

References

1. Angles, R., Gutierrez, C.: The expressive power of SPARQL. In: ISWC 2008
2. Balazevic, I., Allen, C., Hospedales, T.: Multi-relational poincaré graph embeddings. NeurIPS **32**, 4463–4473 (2019)
3. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. Advances in neural information processing systems **26** (2013)
4. Chami, I., Wolf, A., Juan, D.C., Sala, F., Ravi, S., Ré, C.: Low-dimensional hyperbolic knowledge graph embeddings. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 6901–6914 (2020)
5. Cyganiak, R., Wood, D., Lanthaler, M.: RDF 1.1 Concepts and Abstract Syntax. Tech. rep., W3C Recommendation (02 2014)
6. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2d knowledge graph embeddings. In: AAAI (2018)

7. Gregucci, C., Nayyeri, M., Hernández, D., Staab, S.: Link prediction with attention applied on multiple knowledge graph embedding models. In: ACM WebConf (2023)
8. Gu, A., Sala, F., Gunel, B., Ré, C.: Learning mixed-curvature representations in product spaces. In: International Conference on Learning Representations (2018)
9. Harris, S., Seaborne, A.: SPARQL 1.1 Query Language. Tech. rep., W3C Recommendation (03 2013)
10. Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., Melo, G.D., Gutierrez, C., Kirrane, S., Gayo, J.E.L., Navigli, R., Neumaier, S., et al.: Knowledge graphs. ACM Computing Surveys (CSUR) **54**(4), 1–37 (2021)
11. Huang, Z., Chiang, M.F., Lee, W.C.: Line: Logical query reasoning over hierarchical knowledge graphs. In: Proceedings of the 28th ACM SIGKDD. pp. 615–625 (2022)
12. Kiefer, C., Bernstein, A., Stocker, M.: The fundamentals of isparql: A virtual triple approach for similarity-based semantic web tasks. In: The Semantic Web, pp. 295–309. Springer (2007)
13. Krompaß, D., Tresp, V.: Ensemble solutions for link-prediction in knowledge graphs. In: Proceedings of the 2nd Workshop on Linked Data for Knowledge Discovery, Porto, Portugal. pp. 1–10 (2015)
14. Kulmanov, M., Kafkas, S., Karwath, A., Malic, A., Gkoutos, G.V., Dumontier, M., Hoehndorf, R.: Vec2sparql: integrating sparql queries and knowledge graph embeddings. bioRxiv p. 463778 (2018)
15. Nayyeri, M., Vahdati, S., Sallinger, E., Alam, M.M., Yazdi, H.S., Lehmann, J.: Pattern-aware and noise-resilient embedding models. In: ECIR (2021)
16. Ren, H., Hu, W., Leskovec, J.: Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In: ICLR 2020 (2020)
17. Ren, H., Leskovec, J.: Beta embeddings for multi-hop logical reasoning in knowledge graphs. NeurIPS **33**, 19716–19726 (2020)
18. Sun, Z., Deng, Z., Nie, J., Tang, J.: Rotate: Knowledge graph embedding by relational rotation in complex space. In: ICLR 2019, New Orleans, LA, USA (2019)
19. Toutanova, K., Chen, D.: 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 (2015)
20. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: ICML (2016)
21. Wang, K., Liu, Y., Ma, Q., Sheng, Q.Z.: Mulde: Multi-teacher knowledge distillation for low-dimensional knowledge graph embeddings. In: ACM WebConf (2021)
22. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: A survey of approaches and applications. IEEE Transactions on Knowledge and Data Engineering **29**(12), 2724–2743 (2017)
23. Wang, Y., Gemulla, R., Li, H.: On multi-relational link prediction with bilinear models. In: AAAI (2018)
24. Wang, Y., Chen, Y., Zhang, Z., Wang, T.: A probabilistic ensemble approach for knowledge graph embedding. Neurocomputing (2022)
25. Xiong, B., Zhu, S., Nayyeri, M., Xu, C., Pan, S., Zhou, C., Staab, S.: Ultrahyperbolic knowledge graph embeddings. In: ACM SIGKDD (2022)
26. Xiong, W., Hoang, T., Wang, W.Y.: Deeppath: A reinforcement learning method for knowledge graph reasoning. In: EMNLP (2017)
27. Xu, C., Nayyeri, M., Vahdati, S., Lehmann, J.: Multiple run ensemble learning with low-dimensional knowledge graph embeddings. In: IJCNN (2021)
28. Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: ICLR (2015)