

Schema Injection in the Wild for Link Prediction: Lessons Learned from Incomplete Schema Cases

Nicolas Robert^{1,*}, Pierre Monnin¹ and Catherine Faron¹

¹Université Côte d'Azur, Inria, CNRS, I3S, Sophia-Antipolis, France

Abstract

Even though knowledge graph embedding models for link prediction (LP) often disregard schemas of knowledge graphs, recent research increasingly studies their impact within the learning process. Although yielding promising results, these approaches are often evaluated on datasets providing complete schemas, i.e., all entities are typed and all relations are associated with both a domain and a range. This setting limits the assessment of the effectiveness of the approaches, motivating our present work where we re-evaluate a schema injection method on datasets broadly accepted in LP with incomplete schemas. Our results show a more nuanced performance in such complex settings, with a trade-off between rank-based performance (MRR, Hits@K) and semantic capture (Sem@K).

Keywords

Link Prediction, Knowledge Graph Embedding, Loss functions, Schema injection, Neurosymbolic AI

1. Introduction

Knowledge Graphs (KGs) are inherently incomplete due to their manual or semi-automatic construction approaches [1], which gives rise to various refinement tasks. Among them, *Link Prediction* (LP) consists in predicting the missing tail or head of an incomplete triple $(h, r, ?)$ or $(?, r, t)$, respectively. *Knowledge Graph Embedding Models* (KGEMs) have been extensively studied for LP [2], where these models learn vectors for entities and relations to compute a plausibility score for triples (h, r, t) .

While early KGEM approaches disregarded the schemas of KGs (e.g., subclass, domain, and range axioms), an increasing number of research work study their impact [3, 4, 5, 6, 7]. In particular, Hubert et al. [6] proposed to inject domain and range information in the three main loss functions adopted in LP by sampling and processing differently negative triples that are schema-coherent (e.g., (Paris, capitalOf, Italy)) and schema-incoherent (e.g., (Paris, capitalOf, BarackObama)). Their experiments showed promising results, both in schema capture and rank performance. However, their setting is limited to datasets where all entities are typed and all relations are associated with both a domain and a range. This leaves open the question of the impact of schema injection in a more realistic scenario, with some entities untyped and some relations without a domain or a range, motivating the following research questions:

- RQ1.** How do schema-driven loss functions affect link prediction performance of KGEMs on knowledge graphs with incomplete schemas?
- RQ2.** How does the effect of schema-driven loss functions vary depending on the completeness of relation signatures?

In this paper, we evaluate the schema-injection method of Hubert et al. [6] on more realistic scenarios with incomplete schemas and assess which triples of the dataset benefit from it. Results show that this method tends to trade off rank-based performance with schema capture. This highlights the need for more robust schema-injection approaches that can jointly improve rank-based and semantic metrics.

Posters, Demos, Blue Sky, and Tutorials at SEMANTiCS 2026, Sep 2026, Ghent, Belgium

*Corresponding author.

✉ nicolas.robert@inria.fr (N. Robert); pierre.monnin@inria.fr (P. Monnin); catherine.faron@inria.fr (C. Faron)

🌐 <https://github.com/NicoRobertIn> (N. Robert); <https://pmonnin.github.io/> (P. Monnin); <https://webusers.i3s.unice.fr/~faron/> (C. Faron)

🆔 0009-0009-2595-6168 (N. Robert); 0000-0002-2017-8426 (P. Monnin); 0000-0001-5959-5561 (C. Faron)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

2. Preliminaries

Families of KGEMs. There are three main families of KGEMs, differing in the way they score the plausibility of a triple. *Geometric models* such as *TransE* [8] and *TransH* [9] use a distance function and consider the relation embedding as a geometric transformation applied on the head entity to obtain the tail entity. *Semantic matching models* such as *DistMult* [10], *ComplEx* [11], and *TuckER* [12] use a bilinear scoring function. *Neural network-based models* such as *ConvE* [13] rely on artificial neural networks to compute plausibility scores.

Families of loss functions. KGEMs are trained to assign high scores to positive triples and low scores to negative triples that are generated by corrupting positive ones. To this aim, there are three main families of loss functions, differing in the way they drive models to assign high scores to positive triples and low scores to negative triples. *Pointwise* loss functions such as the *Pointwise Logistic Loss* (PLL) and the *Binary Cross Entropy Loss* (BCEL) compare the predicted score and the ground truth of each triple whereas *pairwise* loss functions such as the *Pairwise Hinge Loss* (PHL) compare scores of positive and negative triples to maximize their differences.

Schema-driven PHL, BCEL and PLL as defined in [6]. Hubert et al. consider the schema-coherence of each negative triple, using a *semantic factor* hyperparameter ϵ . Their PHL function applies a margin of $\gamma \cdot \epsilon$, instead of standard γ , between positive and schema-coherent negative triples. Their BCEL function uses ϵ as label value for schema-coherent negative triples. Finally, their PLL function uses ϵ as a probability, in each batch, of considering schema-coherent negative triples as positive.

3. Experiments

To re-evaluate Hubert et al. [6] approach in more realistic settings, we choose datasets that are broadly accepted by the LP community, namely *NELL-995* [14], *DB100k* [15], and *YAGO3-10* [16]. These datasets were originally introduced without a schema, and thus, we consider their semantic extensions proposed by Robert et al. [17], *NELL-995+*, *DB100k+*, and *YAGO3-10+*, which include the schema associated to their respective original KGs. Note that not all entities are typed and not all relations have a domain and a range. These datasets are summarized in Table 1.

To compare vanilla training with schema-injection, we train each considered KGEM twice with the same random seed. For each KGEM, we perform two grid searches: one varying embedding dimension, batch size, learning weight, and regularizer weight; and another varying all the previous hyperparameters as well as the ϵ semantic factor.¹ To ensure that differences between loss functions are not related to the negative sampling, we use a fixed negative sampling procedure. For every positive triple in the training set, we generate schema-coherent and schema-incoherent negative triples through head corruption and tail corruption. For each of these four corruption types, up to 500 negative triples are generated. Then, at each batch computation, for every positive triple in a batch, one schema-coherent and one schema-incoherent negative triple are picked from the generated negatives. Note that in the absence of schema information, schema-coherent negative triples cannot be generated. In this case, we randomly generate 500 negative triples via head corruption and 500 negative triples via tail corruption. At each batch computation, we sample two of these generated negatives. Each model is trained for 400 epochs, with a checkpoint saved every 10 epochs. The best checkpoint is selected on the MRR against the validation set and evaluated on the test set. Metrics are computed on the whole test set and on subsets where relations are fully signed, with domain only, range only and without signature.

Table 2 presents the performances obtained on *NELL-995+*, *DB100k+*, and *YAGO3-10+*, allowing to answer **RQ1**. To answer **RQ2**, we further analyze KGEM performance according to whether the relations in test triples have both domain and range, only one of them, or neither. For this, we only

¹Loss functions, datasets, hyperparameter spaces and code are available on GitHub: <https://github.com/Wimmics/schema-injection-in-the-wild>

Table 1

Statistics of NELL-995+, DB100k+, and YAGO3-10+.

Dataset	NELL-995+	DB100k+	YAGO3-10+
# Train triples	149,678	597,572	1,079,040
# Valid triples	543	50,000	5,000
# Test triples	3,992	50,000	5,000
# Entities	74,492	99,604	123,182
# Typed entities	74,492	92,558	121,009
# Types	267	289	93,937
# Relations	200	470	37
# Domains	190	354	37
# Ranges	190	361	28
# Train triples per relation	748.39 $\pm$ 1,362.18	1,271.43 $\pm$ 4,845.48	29,163.24 $\pm$ 78,634.38
# Triple with domain & range	107,790	229,831	1,067,137
# Triple with domain only	6,545	48,220	21,903
# Triple with range only	25,779	346,945	0
# Triple without signature	14,099	72,576	0

consider DB100k+ as it is the only dataset of the three with thousands triples of each category. The other two datasets contain too few triples of each category to provide statistically significant results. Results of this detailed analysis are provided in Table 3.

4. Discussion

Schema-injection trades off MRR and Hits@10 with Sem@10. Contrary to the experiments in [6] on datasets with complete schemas, our setting on datasets with incomplete schemas show in most cases an improvement in Sem@10 and slightly reduced MRR and Hits@10. This observation underscores that incomplete schemas result in a more difficult learning setting, hindering rank-based performance. MRR, Hits@10, and Sem@10 improve jointly only for DistMult on NELL-995+ and DB100k+, and for ConvE and TuckER on YAGO3-10+. Regarding DistMult, it is the only semantic matching model combined with a pairwise loss function. This combination of model and loss might learn more easily the constraints on the embedding space entailed by schema injection. Regarding ConvE and TuckER, they are both 1-N scorers, where each triple is evaluated against all entities in the graph. This leads to an additional training signal that could explain their different performances (see below).

Rank performance degradation increases with the number of training triples per relation.

On average (Table 1), a relation has 748 train triples in NELL-995+, 1,271 in DB100k+, and 29,163 in YAGO3-10+. For each KGEM, comparing its performance across the three datasets, we observe a higher rank performance degradation when the number of training triples per relation increases. This can be explained as the more a KGEM has training triples for a relation, the more it is likely to learn its underlying patterns, such as its signature. When a KGEM is already well trained for a relation in a KG, guiding it by schema-injection to further learn the schema may lead to overfit the schema capture. This, in turn, may mislead the model and degrade rank performance since some true triples can be schema-incoherent. ConvE and TuckER appear less impacted by this issue. Their 1-N scoring mechanism provides additional training signals from all possible negatives that may compensate the schema-driven signal. To address schema capture overfitting, a future work lies in adapting the value of the semantic factor to the number of train triples of relations.

Detailed performance results are inconclusive. One might expect a schema-injection method to particularly benefit triples whose relations have known signatures, but our results (Table 3) do not

Table 2

MRR, Hits@10, and Sem@10 of KGEMs using standard loss functions (STD) and the schema-driven loss functions from [6]. In bold is the best score obtained by a given model on a given dataset. † indicates a value that is significantly higher for a given metric, a given model, and a given dataset, considering a 95% confidence interval (difference of 3.17% in NELL-995+, 0.89% in DB100k+, and 2.83% in YAGO3-10+).

Loss	KGEM	Model	NELL-995+			DB100k+			YAGO3-10+		
			MRR	Hits@10	Sem@10	MRR	Hits@10	Sem@10	MRR	Hits@10	Sem@10
$\mathcal{L}_{\text{PHL}}$	TransE	STD	0.334	0.444	0.540	† 0.267	† 0.510	0.607	† 0.227	† 0.415	0.919
		[6]	0.325	0.419	† 0.886	0.247	0.457	† 0.836	0.155	0.279	† 0.998
	TransH	STD	0.354	0.457	0.485	0.260	† 0.504	0.600	† 0.237	† 0.429	0.921
		[6]	0.348	0.454	† 0.871	0.258	0.471	† 0.840	0.162	0.291	† 0.994
$\mathcal{L}_{\text{BCEL}}$	DistMult	STD	0.247	0.330	0.375	0.246	0.415	0.391	0.190	† 0.342	0.767
		[6]	0.255	0.342	† 0.512	† 0.257	† 0.434	† 0.649	0.174	0.298	† 0.965
	ConvE	STD	0.416	0.527	0.855	† 0.407	† 0.562	† 0.799	0.457	0.631	0.974
		[6]	0.404	0.502	† 0.953	0.364	0.502	0.779	† 0.533	† 0.671	0.974
$\mathcal{L}_{\text{PLL}}$	TuckER	STD	† 0.429	† 0.537	0.823	† 0.367	† 0.547	0.829	0.271	0.437	0.990
		[6]	0.364	0.466	† 0.893	0.214	0.347	0.829	† 0.364	† 0.531	0.994
	ComplEx	STD	† 0.323	† 0.407	0.261	† 0.342	0.447	0.468	† 0.324	† 0.495	0.811
		[6]	0.261	0.350	† 0.389	0.239	0.449	† 0.507	0.255	0.429	† 0.876

support this assumption. Except the increase of Sem@10, we do not identify any pattern in a model family or a loss function in any subset of triples. This highlights the need to investigate other approaches of schema injection that would lead to less noisy and more predictable improvements in performance.

Toward improving schema-injection. The observed shortcomings of the schema-injection of [6] could also be linked to some theoretical flaws. Recall that a KGEM is trained to respectively predict high scores to positive triples, and low scores to negatives. Intuitively, a schema-driven KGEM should predict higher scores to schema-coherent negative triples than to schema-incoherent negative triples. However, the definition of the schema-driven PHL of [6] only applies a loose constraint where schema-coherent negative triples can have scores closer to positives than schema-incoherent negatives, which can create a confusion between schema-coherent and schema-incoherent negatives. Regarding their schema-driven PLL, it stochastically considers schema-coherent negatives in batches as positives during training. This behavior can constrain a KGEM to predict different scores to a same triple at different epochs, potentially causing convergence issues and preventing the KGEM to reach its optimal performance. These two methodological issues affect non-only schema-coherent negative triples but other triples that involve their entities and relations. Hence, they constitute promising avenues to improve schema-injection at the theoretical level.

5. Conclusion

In this paper, we further investigated the approach of Hubert et al. [6] injecting domain and range in loss functions to train KGEMs for link prediction. When evaluated on more realistic datasets with incomplete schema, results show a degradation of MRR and Hits@10 in return of an increased Sem@10, contrary to observations on datasets with complete schema reported in [6]. These inconclusive results and the theoretical limits of the approach we raised motivate the extensive evaluation of other injection approaches [4, 18] as well as the development of theoretically sound injection methods capable of facing realistic scenarios, which we leave for future work.

Table 3

Performance on DB100k+ according to whether the relations in test triples have both domain and range, only one of them, or neither. n denotes the number of triples of each subset. The best results between STD and [6] are in bold. $\dagger$ indicates a significant difference w.r.t. confidence intervals of 95% (1.56% for domain & range, 3.37% for domain only, 1.27% for range only, and 2.79% for no signature). *NA* means Sem@10 cannot be computed in the absence of relation signature.

Loss	Model	Metric	Domain & Range ($n=16,517$)				Domain Only ($n=3,527$)				Range Only ($n=24,827$)				No Signature ($n=5,129$)			
			STD	Head [6]	STD	Tail [6]	STD	Head [6]	STD	Tail [6]	STD	Head [6]	STD	Tail [6]	STD	Head [6]	STD	Tail [6]
$\mathcal{L}_{PHL}$	TransE	MRR	0.151	0.141	$\dagger$ 0.326	0.250	0.149	0.147	0.627	0.654	0.152	0.140	0.370	0.360	0.160	0.171	0.427	0.421
		Hits@10	$\dagger$ 0.326	0.269	$\dagger$ 0.586	0.471	0.285	0.264	0.803	0.803	$\dagger$ 0.370	0.328	$\dagger$ 0.699	0.655	0.338	0.353	0.706	0.689
		Sem@10	0.687	$\dagger$ 0.890	0.643	$\dagger$ 0.846	0.645	$\dagger$ 0.845	NA	NA	NA	NA	0.465	$\dagger$ 0.754	NA	NA	NA	NA
	TransH	MRR	0.148	0.151	$\dagger$ 0.323	0.273	0.141	0.156	0.611	$\dagger$ 0.674	0.154	0.145	0.363	0.366	0.145	$\dagger$ 0.183	0.395	$\dagger$ 0.430
		Hits@10	$\dagger$ 0.320	0.277	$\dagger$ 0.583	0.492	0.293	0.302	0.803	0.818	$\dagger$ 0.371	0.341	$\dagger$ 0.689	0.662	0.328	$\dagger$ 0.374	0.695	0.707
		Sem@10	0.664	$\dagger$ 0.899	0.644	$\dagger$ 0.849	0.666	$\dagger$ 0.836	NA	NA	NA	NA	0.463	$\dagger$ 0.756	NA	NA	NA	NA
	DistMult	MRR	0.106	$\dagger$ 0.138	0.302	0.299	0.067	0.099	0.637	0.650	0.146	$\dagger$ 0.165	0.387	0.378	0.068	0.093	0.356	0.365
		Hits@10	0.216	$\dagger$ 0.253	0.498	0.499	0.126	$\dagger$ 0.199	0.768	$\dagger$ 0.795	0.295	$\dagger$ 0.314	0.629	0.620	0.162	$\dagger$ 0.213	0.577	$\dagger$ 0.616
		Sem@10	0.318	$\dagger$ 0.674	0.512	$\dagger$ 0.681	0.345	$\dagger$ 0.715	NA	NA	NA	NA	0.422	$\dagger$ 0.575	NA	NA	NA	NA
	ConvE	MRR	$\dagger$ 0.242	0.216	$\dagger$ 0.431	0.344	$\dagger$ 0.296	0.182	$\dagger$ 0.733	0.643	0.312	0.328	$\dagger$ 0.558	0.488	0.288	0.273	$\dagger$ 0.568	0.495
		Hits@10	$\dagger$ 0.367	0.333	$\dagger$ 0.620	0.531	$\dagger$ 0.469	0.324	$\dagger$ 0.855	0.795	$\dagger$ 0.423	0.406	$\dagger$ 0.745	0.658	$\dagger$ 0.447	0.396	$\dagger$ 0.769	0.690
		Sem@10	0.858	0.857	0.769	0.757	$\dagger$ 0.819	0.785	NA	NA	NA	NA	$\dagger$ 0.732	0.686	NA	NA	NA	NA
$\mathcal{L}_{BCEL}$	TuckER	MRR	$\dagger$ 0.228	0.142	$\dagger$ 0.427	0.243	0.282	0.264	$\dagger$ 0.745	0.630	$\dagger$ 0.241	0.117	$\dagger$ 0.490	0.261	$\dagger$ 0.245	0.190	$\dagger$ 0.559	0.302
		Hits@10	$\dagger$ 0.370	0.260	$\dagger$ 0.629	0.381	$\dagger$ 0.478	0.434	$\dagger$ 0.870	0.760	$\dagger$ 0.382	0.228	$\dagger$ 0.712	0.404	$\dagger$ 0.446	0.335	$\dagger$ 0.775	0.487
		Sem@10	0.907	0.906	$\dagger$ 0.821	0.789	$\dagger$ 0.829	0.859	NA	NA	NA	NA	$\dagger$ 0.748	0.720	NA	NA	NA	NA
	ComplEx	MRR	$\dagger$ 0.187	0.132	$\dagger$ 0.322	0.296	$\dagger$ 0.155	0.107	$\dagger$ 0.606	0.566	$\dagger$ 0.323	0.153	$\dagger$ 0.465	0.338	$\dagger$ 0.208	0.107	$\dagger$ 0.483	0.335
		Hits@10	$\dagger$ 0.277	0.266	0.451	$\dagger$ 0.505	$\dagger$ 0.276	0.210	0.679	$\dagger$ 0.731	$\dagger$ 0.394	0.341	0.578	$\dagger$ 0.646	$\dagger$ 0.335	0.244	$\dagger$ 0.680	0.599
		Sem@10	$\dagger$ 0.597	0.530	0.383	$\dagger$ 0.483	0.590	0.586	NA	NA	NA	NA	0.313	$\dagger$ 0.468	NA	NA	NA	NA

Acknowledgments

Experiments presented in this paper were carried out using the Grid’5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see <https://www.grid5000.fr>). This work was supported by the French government through the France 2030 investment plan managed by the National Research Agency (ANR), as part of the Initiative of Excellence Université Côte d’Azur under reference number ANR-15-IDEX-01. The authors are grateful to the Université Côte d’Azur’s Center for High-Performance Computing (OPAL infrastructure) for providing resources and support. This publication is based upon work from COST Action CA23147 GOBLIN - Global Network on Large-Scale, Cross-domain and Multilingual Open Knowledge Graphs, supported by COST (European Cooperation in Science and Technology, <https://www.cost.eu>).

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

- [1] A. Hogan, et al., Knowledge Graphs, Synthesis Lectures on Data, Semantics, and Knowledge, 2021.
- [2] S. Ji, et al., A survey on knowledge graphs: Representation, acquisition, and applications, IEEE Trans. Neural Networks Learn. Syst. 33 (2022).
- [3] P. Monnin, et al., Discovering alignment relations with graph convolutional networks: A biomedical case study, Semantic Web 13 (2022).
- [4] A. Iana, H. Paulheim, More is not always better: The negative impact of a-box materialization on rdf2vec knowledge graph embeddings, in: Workshop on Combining Symbolic and Sub-symbolic Methods and their Applications, CSSA, 2020.
- [5] N. Hubert, et al., Sem@K: Is my knowledge graph embedding model semantic-aware?, Semantic Web 14 (2023).
- [6] N. Hubert, et al., Treat different negatives differently: Enriching loss functions with domain and range constraints for link prediction, in: 21st Extended Semantic Web Conf., ESWC, 2024.

- [7] C. d’Amato, N. F. Quatraro, N. Fanizzi, Injecting background knowledge into embedding models for predictive tasks on knowledge graphs, in: 18th Extended Semantic Web Conf., ESWC, 2021.
- [8] A. Bordes, et al., Translating embeddings for modeling multi-relational data, in: Advances in Neural Information Processing Systems 26, 2013.
- [9] Z. Wang, et al., Knowledge graph embedding by translating on hyperplanes, in: 28th AAAI Conf. on Artificial Intelligence, 2014.
- [10] B. Yang, et al., Embedding entities and relations for learning and inference in knowledge bases, in: 3rd Int. Conf. on Learning Representations, ICLR, 2015.
- [11] T. Trouillon, et al., Complex embeddings for simple link prediction, in: 33nd Int. Conf. on Machine Learning, ICML, 2016.
- [12] I. Balazevic, C. Allen, T. M. Hospedales, Tucker: Tensor factorization for knowledge graph completion, in: Conf. on Empirical Methods in Natural Language Processing and the 9th Int. Joint Conf. on Natural Language Processing, EMNLP-IJCNLP, 2019.
- [13] T. Dettmers, et al., Convolutional 2d knowledge graph embeddings, in: 32nd AAAI Conf. on Artificial Intelligence, AAAI, 2018.
- [14] W. Xiong, T. Hoang, W. Y. Wang, Deeppath: A reinforcement learning method for knowledge graph reasoning, in: Conf. on Empirical Methods in Natural Language Processing, EMNLP, 2017.
- [15] B. Ding, et al., Improving knowledge graph embedding using simple constraints, in: 56th Annual Meeting of the Association for Computational Linguistics, ACL, 2018.
- [16] F. Mahdisoltani, J. Biega, F. M. Suchanek, YAGO3: A knowledge base from multilingual wikipedias, in: 7th Biennial Conf. on Innovative Data Systems Research, CIDR, 2015.
- [17] N. Robert, P. Monnin, C. Faron, Semantically enriched datasets for link prediction: DB100k+, NELL-995+ and YAGO3-10+, in: Advanced Neuro-Symbolic Applications workshop, ANSyA, 2025.
- [18] N. Hubert, H. Paulheim, P. Monnin, A. Brun, D. Monticolo, Schema first! learn versatile knowledge graph embeddings by capturing semantics with machine, in: Proceedings of the 12th Knowledge Capture Conference 2023, K-CAP 2023, Pensacola, FL, USA, December 5-7, 2023, ACM, 2023, pp. 188–196.