KGE-PATH: INTERPRETABLE LINK PREDICTION BASED ON KNOWLEDGE GRAPH EMBEDDING

Anonymous authors
Paper under double-blind review

ABSTRACT

Explaining why a prediction is made is essential for the prediction to be trusted and acted upon. Most explainers of link predictions made by graph and knowledge-graph models are post-hoc: they treat the prediction model as a black box and perturb its input to obtain an explanation. This forces the model to run on inputs whose distribution differs from the one it was trained on, so the faithfulness of the explanation is assumed rather than guaranteed. It is also time-consuming, since an additional model has to be trained or optimized to produce the perturbation. To address both issues, we propose **KGE-Path**, a transparent knowledge graph embedding (KGE) model whose prediction relies only on the geometric relationship between the embeddings of the two nodes, together with an explanation method that is derived from this relationship. We exploit this transparency and score each path connecting the two query nodes using only the embeddings of the model and the topological structure of the input graph; the resulting paths, together with their scores, form an explanation that is read off directly from the model and therefore faithfully reflects how the prediction is made. To make the model better suited to this explanation, we also adapt the standard KGE training: an *edge penalty* that performs negative sampling over the relation types, a *node penalty* that exploits the information of the neighborhood, and a regularization term that further aligns the explanation with the prediction. On six datasets, KGE-Path explains better than GNN- and KGE-based explainers, and it explains a query pair faster than the baselines on four of the six datasets.

1 INTRODUCTION

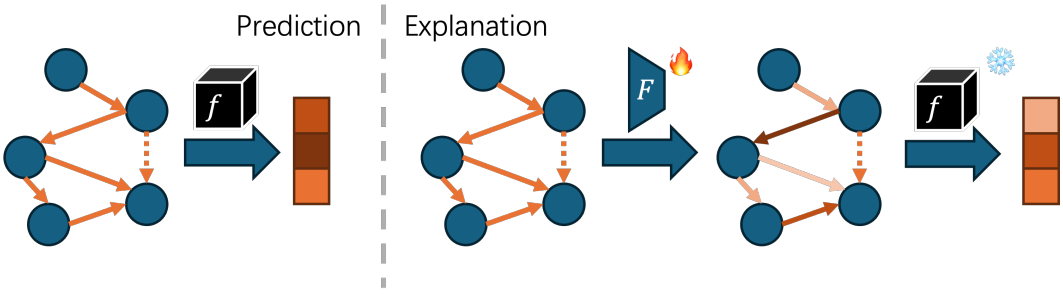


Figure 1: Limitation of current post-hoc explanations. The explanation is produced by an additional module F that is trained on a frozen prediction model f, instead of being read off from f itself. Since the model f is treated as a black box, this method has to perturb the input of f to obtain the explanation. Since F makes f run on inputs that are no longer drawn from the data on which f was trained, the predictions of the two panels are obtained under different distributions, and the faithfulness of the explanation relies on the assumption that the behavior of f on these inputs still reflects the original prediction.

Link prediction is one of the most fundamental tasks on graphs. Given nodes and edges that have already been observed, link prediction aims to infer the properties of the edges that have not been

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

observed yet, such as whether an edge exists between two nodes and which type this edge belongs to. This capability is valuable in many real-world applications: in recommender systems it is used to predict which items a user may be interested in, in social network analysis it is used to infer the type of an unobserved social relation, and in knowledge base completion it is used to fill in the missing relation between two entities. This task is also known as relation prediction in some contexts.

Two families of models are commonly used for this purpose: graph neural networks (GNNs) Kipf & Welling (2016); Veličković et al. (2017) and knowledge graph embedding (KGE) models. GNNs aggregate the messages of neighboring nodes to obtain a node embedding, and then predict the property of an edge with an additional scoring function that takes the embeddings of its two endpoints as input; relational variants such as RGCN Schlichtkrull et al. (2018), RGAT Busbridge et al. (2019) and CompGCN Vashishth et al. (2020) make the aggregation depend on the relation types. KGE models such as TransE Bordes et al. (2013), DistMult Yang et al. (2015), ComplEx Trouillon et al. (2016), ConvE Dettmers et al. (2017) and RotatE Sun et al. (2019) learn an embedding for every entity and every relation type and score a triple by a function of the three embeddings.

Beyond prediction accuracy, explanation is a central requirement in many applications, since a prediction can hardly be trusted or acted upon if it cannot be justified. The explainers built for their link predictions, such as GNNExplainer Ying et al. (2019), PGExplainer Luo et al. (2020), ExplaiNE Kang et al. (2019), PaGE-Link Zhang et al. (2023), Power-Link Chang et al. (2024), and KGExplainer Ma et al. (2025), fall short as shown in Figure 1: they are mostly post-hoc, i.e., the prediction model is treated as a black box and they have to perturb its input to obtain the explanation. The prediction model is therefore run on a distribution that differs from the one it was trained on, so whether the explanation is faithful to how the prediction is made is assumed rather than guaranteed. The whole procedure is also time-consuming, since an additional model has to be trained or optimized to produce the perturbation.

We propose **KGE-Path**, an interpretable model built on the geometric relationship of KGE embeddings. When making a prediction, this geometric model expresses the relation between two nodes purely through the geometric relationship between the embeddings of the two nodes,which makes the model transparent. We exploit this transparency for explanation: given a query pair, we first enumerate the paths of bounded length between the two endpoints, and then score each path by how faithfully the composition of its relation embeddings reproduces the geometric relationship between the embeddings of the two endpoints. Because the explanation is read off directly from the trained embeddings and the topological structure of the input graph, it requires no additional training and faithfully reflects how the prediction is made. At test time, its cost therefore consists only of path enumeration and score computation. A path connecting the two query nodes is also easy for a human to read and matches the intuition behind link prediction. To make the model better suited to this path-based explanation, we also adapt the standard KGE training: we use an edge penalty and a node penalty on the geometric relationship, and a regularization term that further aligns the explanation with the prediction during training.

Our contributions can be summarized as follows:

- We propose a path-oriented explanation method for link prediction. It reads the explanation directly from the geometric relationship of the KGE model, so that the explanation requires no additional training and is consistent with the way in which the prediction is made.
- We further propose two penalties: an edge penalty that performs negative sampling over the relation types and a node penalty that makes full use of the relations of the neighboring nodes. These two losses improve prediction accuracy and help produce a better explanation. We also introduce a regularization term that further aligns the explanation with the prediction.
- Extensive experiments on six datasets show that KGE-Path explains better than GNN- and KGE-based explainers, and further analyses demonstrate its efficiency and the importance of each component.

A detailed discussion of the related work is given in Appendix A.1.

2 PRELIMINARIES

In this section, we define the notation and the problem.

Graphs. Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ denote a simple directed graph, where $\mathcal{N}$ is a set of nodes and $\mathcal{E}$ is a set of edges. Each edge $e = (n, m) \in \mathcal{E}$ is an ordered pair of nodes $n, m \in \mathcal{N}$, indicating a directed connection from n to m, and carries a relation type r from a set $\mathcal{R}$ of relation types, where $|\mathcal{R}|$ denotes the number of relation types.

Knowledge Graph. A knowledge graph is a set of triples $\mathcal{K} = \{(h, r, t)\} \subseteq \mathcal{N} \times \mathcal{R} \times \mathcal{N}$, where $\mathcal{N}$ is an entity set and $\mathcal{R}$ is a relation set. Each triple $(h, r, t) \in \mathcal{K}$ can be viewed as a directed edge labeled with the relation type r, pointing from the head entity h to the tail entity t. For simplicity, in what follows we treat $\mathcal{K}$ as a simple directed graph with relation types on its edges; this requires that there is at most one relation type for each ordered pair of entities, i.e., the relation assignment is a single-valued function $\mathcal{N} \times \mathcal{N} \to \mathcal{R}$.

Notation. In the rest of this paper, we use x_i to denote the i-th node and $\boldsymbol{x}_i$ its embedding, r_i to denote the i-th relation type and $\boldsymbol{r}_i$ its embedding, and $\boldsymbol{e}_{ij}$ to denote the embedding of the relation type on the edge from the i-th node to the j-th node, so that all the edges of the same relation type share it, i.e., $\boldsymbol{e}_{ij} = \boldsymbol{r}_r$ for the relation type r of the edge. We also use $(\boldsymbol{x})_d$ to denote the d-th component of a vector $\boldsymbol{x}$. All the embeddings are complex vectors of the same dimension D, and all the operations on them are component-wise: $\odot$ denotes the Hadamard product and $/$ the component-wise division.

Problem Definition. Given a knowledge graph $\mathcal{K}$, our trained model performs the following two tasks:

1. It maps an ordered pair of existing nodes to a relation type, i.e., a mapping $\mathcal{N} \times \mathcal{N} \to \mathcal{R}$; we refer to this task as link prediction in the rest of the paper.
2. For each link prediction, it produces an explanation, i.e., a collection of paths, where each path is a simple path of at most M edges connecting the head and the tail entity in the knowledge graph and carries a score, with M the maximum path length. Formally, writing $\mathcal{E}^N$ for the N-fold Cartesian product $\mathcal{E} \times \cdots \times \mathcal{E}$ of the edge set with itself, an explanation is a mapping $\mathcal{N} \times \mathcal{N} \to \mathcal{P}\Big(\bigcup_{N=1}^{M} \left(\mathcal{E}^N \times (0, 1)\right)\Big)$, where the scores lie in $(0, 1) \subseteq \mathbb{R}$ and $\mathcal{P}(S)$ denotes the power set of S.

3 METHODOLOGY

Figure 2 gives an overview of KGE-Path, which consists of three parts. In the *training* stage, we update the node and the relation embeddings with the two penalties and the regularization term described below. In the *prediction* stage, we infer the relation of a query pair from the geometric relationship between its two node embeddings, and match it with the closest relation embedding, i.e., a match based on distance (Eq. equation 7). In the *explanation* stage, we enumerate the paths between the two query nodes and score each of them by how faithfully it reproduces that geometric relationship.

3.1 TRAINING AND PREDICTION

To make KGE models focus more on the link prediction task, we move away from the standard KGE training, which relies on negative sampling by corrupting the two query entities, and instead perform negative sampling over the relation types, i.e., we corrupt the relation type of an edge. Since the number of relation types is much smaller than the number of entities, the space of such edge-level negative sampling is small. To compensate for this, we introduce a node penalty that exploits the information of the neighborhood.

RotatE represents a relation as a rotation in the complex plane and scores a triple (h, r, t) by $-\|\boldsymbol{x}_h \odot \boldsymbol{r} - \boldsymbol{x}_t\|_2$, where $\|(\boldsymbol{r})_d\|_2 = 1$ for every component d. A well-trained RotatE model therefore satisfies the following equation for every observed triple (x_i, r, x_j) in the knowledge graph:

$$\boldsymbol{x}_i \odot \boldsymbol{e}_{ij} = \boldsymbol{x}_j \tag{1}$$

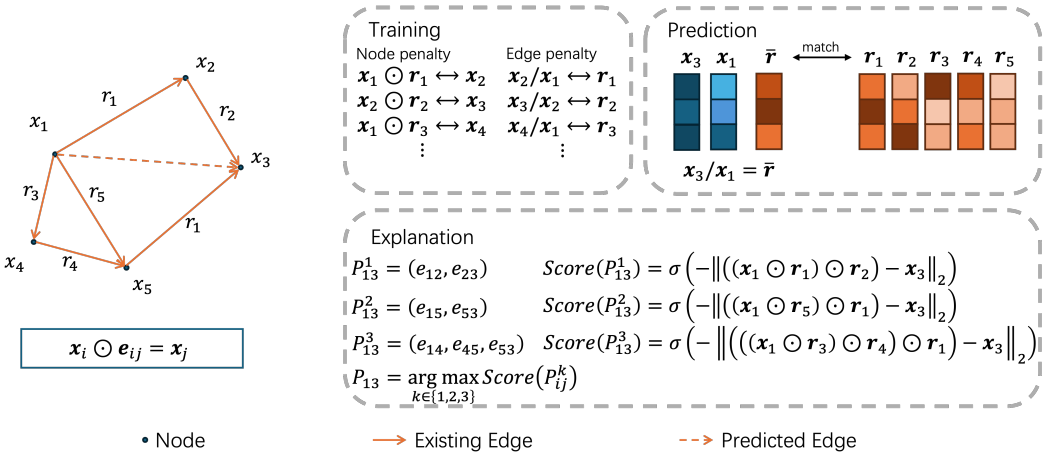


Figure 2: Overview of the proposed KGE-Path. On the left, we show the node embeddings trained by our model; they should approximately satisfy Eq. equation 1, which is enforced during training by the node penalty and the edge penalty. During prediction, we only perform a match based on distance between the relation embedding inferred from the query node embeddings and the learned relation embeddings. During explanation, we enumerate all candidate paths and compute a score for each of them. For clarity of the figure, we omit the projection function.

Intuitively, Eq. equation 1 allows us to recover any one of the three quantities given the other two, and we exploit this property in two places: to infer the relation of a node pair from the two node embeddings, and to propagate a node embedding along a path. For the former we need the result to be a valid relation embedding, i.e., to lie on the unit circle in every component, so we project relation embeddings $\boldsymbol{z}$ by:

$$\text{Project}(\boldsymbol{z}) = ((\boldsymbol{z})_1/\|(\boldsymbol{z})_1\|_2, \ldots, (\boldsymbol{z})_D/\|(\boldsymbol{z})_D\|_2), \tag{2}$$

To exploit the information of the whole neighborhood and to update the embeddings of all the nodes at once, we iterate this propagation inside each training epoch: we run L rounds, which we also call layers, and in the l-th round each node is updated from $(l-1)$-th round node embeddings of its in-neighbors, so that after L rounds the update of a node depends on the embeddings of the nodes in its L-hop in-neighborhood. For each node x_j in the l-th round, we first compute its embedding $\boldsymbol{x}_j^{(l)}$ via the following update:

$$\boldsymbol{x}_j^{(l)} = \frac{\epsilon}{\sum_{i \in \text{Neigh}(j)} w_{ij}^{(l)}} \sum_{i \in \text{Neigh}(j)} w_{ij}^{(l)} \left(\boldsymbol{x}_i^{(l-1)} \odot \boldsymbol{e}_{ij}\right) + (1-\epsilon)\boldsymbol{x}_j^{(l-1)} \tag{3}$$

where $\epsilon \in (0, 1]$ is the update ratio, $w_{ij}^{(l)} = w_r^{(l)} \in (0, 1]$ is a weight that depends only on the relation type r of the edge (x_i, x_j), and $\text{Neigh}(j)$ is the set of indices of the in-neighbors of node j (if this set is empty, we set $\boldsymbol{x}_j^{(l)} = \boldsymbol{x}_j^{(l-1)}$). Since $\sum_i w_{ij}^{(l)}$ normalizes the weights, Eq. equation 3 is a convex combination: it moves $\boldsymbol{x}_j$ towards the weighted average of the propagated in-neighbor embeddings $\boldsymbol{x}_i^{(l-1)} \odot \boldsymbol{e}_{ij}$ by a fraction ϵ. The update aggregates the in-neighbors only; we discuss the reverse direction of an edge in the appendix. We then use the distance between this new embedding and the original one as the node penalty:

$$\mathcal{L}_{\text{node}}^{(l)} = \frac{1}{|\mathcal{N}|} \sum_{j=1}^{|\mathcal{N}|} \left\| \boldsymbol{x}_j^{(l)} - \boldsymbol{x}_j^{(l-1)} \right\|_2, \qquad \mathcal{L}_{\text{node}} = \frac{1}{L} \sum_{l=1}^{L} \mathcal{L}_{\text{node}}^{(l)}. \tag{4}$$

The updated embeddings $\boldsymbol{x}_j^{(l)}$ are used only to define the node penalty; the edge penalty below is computed on the embeddings $\boldsymbol{x}_j^{(0)}$ of the current epoch, from which the prediction and the explanation are read off. In the rest of the paper we therefore write $\boldsymbol{x}_j$ for $\boldsymbol{x}_j^{(0)}$ and do not distinguish the layers.

For each edge (x_i, x_j), we minimize the distance between the relation embedding inferred from the two node embeddings and the relation embedding of the ground-truth edge:

$$\mathcal{L}_{\text{edge}} = -\log \frac{\exp(-\|\text{Project}(\boldsymbol{x}_j/\boldsymbol{x}_i) - \boldsymbol{e}_{ij}\|_2)}{\sum_{r_q \in \mathcal{R}} \exp(-\|\text{Project}(\boldsymbol{x}_j/\boldsymbol{x}_i) - \boldsymbol{r}_q\|_2)}. \tag{5}$$

Eq. equation 5 takes the form of a cross-entropy loss over relation types: the softmax normalization implicitly contrasts the ground-truth relation with all the other relations, playing a role similar to negative sampling. The edge penalty is averaged over all the edges of the graph in the same way as the node penalty over the nodes. To sum up, the overall objective is:

$$\mathcal{L} = \mathcal{L}_{\text{node}} + \mathcal{L}_{\text{edge}} + C\mathcal{L}_{\text{reg}} \tag{6}$$

where C is a weighting hyperparameter and $\mathcal{L}_{\text{reg}}$ is a regularization term introduced in the Explanations subsection below.

To make a prediction, we pick the relation type whose embedding is closest to the relation embedding inferred from the query pair:

$$\hat{r} = \arg\min_{r_q \in \mathcal{R}} \|\text{Project}(\boldsymbol{x}_j/\boldsymbol{x}_i) - \boldsymbol{r}_q\|_2. \tag{7}$$

3.2 Explanations

For a query pair (x_i, x_j), our explanation is built from the paths that connect x_i to x_j, together with their contribution to the prediction.

To keep the search tractable, we proceed as follows. First, we extract the union of the 2-hop subgraphs of the two query nodes to narrow down the search space. Second, we further reduce it by applying the k-core pruning strategy used in PaGE-Link Zhang et al. (2023). Then, on the remaining subgraph, we enumerate all simple paths of length at most M, i.e., paths in which no node is repeated, denoted as $\Pi_{ij} = \{P_{ij}^1, \ldots, P_{ij}^K\}$, where the k-th path $P_{ij}^k = (e_1^k, e_2^k, \ldots, e_N^k)$ consists of $N \leq M$ edges e_n^k. Finally, we score each path by the distance between the tail embedding and the embedding obtained by successively applying Eq. equation 1 along the path, starting from the head embedding:

$$\text{Score}(P_{ij}^k) = \sigma(-\|\boldsymbol{x}_i \odot \boldsymbol{e}_1^k \odot \cdots \odot \boldsymbol{e}_N^k - \boldsymbol{x}_j\|_2) \tag{8}$$

where $\boldsymbol{e}_n^k$ is the relation embedding of the n-th edge e_n^k of P_{ij}^k, and σ is the sigmoid function.

The score admits two interpretations. Geometrically, it reflects how faithfully the edges along the path reproduce the geometric relationship between the two query node embeddings $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$: the closer the propagated embedding lies to the tail embedding, the more reliable the path is. Algebraically, it measures the agreement between the relation embeddings composed along the path and the relation embedding that the prediction itself relies on. Indeed, the quantity inside the norm can be rewritten as $\boldsymbol{x}_i \odot \boldsymbol{e}_1^k \odot \cdots \odot \boldsymbol{e}_N^k - \boldsymbol{x}_j = \boldsymbol{x}_j \odot \left(\boldsymbol{e}_1^k \odot \cdots \odot \boldsymbol{e}_N^k / (\boldsymbol{x}_j/\boldsymbol{x}_i) - \mathbf{1}\right)$, so the score compares the composed relation embedding $\boldsymbol{e}_1^k \odot \cdots \odot \boldsymbol{e}_N^k$ with the embedding ratio $\boldsymbol{x}_j/\boldsymbol{x}_i$, from which Eq. equation 7 infers the relation of the query pair. When the squared modulus is the same for every component of $\boldsymbol{x}_j$, i.e., $\|(\boldsymbol{x}_j)_d\|_2^2 = \mu$ for all d (a property that the regularization loss below enforces), this common modulus factors out of the norm and taking the norm on both sides gives

$$\left\|\boldsymbol{x}_i \odot \boldsymbol{e}_1^k \odot \cdots \odot \boldsymbol{e}_N^k - \boldsymbol{x}_j\right\|_2 = \sqrt{\mu} \left\| \frac{\boldsymbol{e}_1^k \odot \cdots \odot \boldsymbol{e}_N^k}{\boldsymbol{x}_j/\boldsymbol{x}_i} - \mathbf{1} \right\|_2, \tag{9}$$

so the two distances differ only by the positive factor $\sqrt{\mu}$, and the ordering of the candidate paths of a query pair is exactly their ordering by agreement with $\boldsymbol{x}_j/\boldsymbol{x}_i$. Since the prediction is driven by the same agreement, the explanation is aligned with the prediction by construction. Our goal is well defined under the assumption below, which the training of the model is designed to support.

Assumption 1 *Let (x_i, x_j) be a query pair and let $E(P) = \|\boldsymbol{x}_i \odot \boldsymbol{e}_1 \odot \cdots \odot \boldsymbol{e}_N - \boldsymbol{x}_j\|_2$ be the composition error of a path P from x_i to x_j, whose score is $\sigma(-E(P))$ (Eq. equation 8). The strength of the evidence that P provides for the prediction decreases approximately monotonically with $E(P)$, so ranking the candidate paths of a query pair by their score is approximately the same as ranking them by the strength of their evidence.*

The edge penalty enforces Eq. equation 1 on every observed edge, and the node penalty extends the same constraint along the neighborhood, so the composition of the relation embeddings along a path that connects the query pair is trained towards $\boldsymbol{x}_j/\boldsymbol{x}_i$, i.e., towards the same quantity from which Eq. equation 7 reads off the prediction. Under Assumption 1 this makes the score of a path a measure of the evidence it carries: among the candidate paths, the ones whose composition agrees more closely with $\boldsymbol{x}_j/\boldsymbol{x}_i$ receive the higher scores.

The factorization in Eq equation 9, however, requires the squared modulus to be the same for every component of $\boldsymbol{x}_j$, i.e., $\|(\boldsymbol{x}_j)_d\|_2^2 = \mu$ for all d, which the training above does not enforce. When the moduli differ, the quantity inside the norm is a weighted sum of the per-component disagreements, where the weight of component d is $|(\boldsymbol{x}_j)_d|_2^2$, so a path may be ranked high because it agrees with $\boldsymbol{x}_j/\boldsymbol{x}_i$ on the components that carry a large modulus, rather than because it reproduces the relation that the prediction relies on. We therefore add a regularization term that makes the modulus constant across the components of every node embedding:

$$\mathcal{L}_{\text{reg}} = \frac{1}{|\mathcal{N}|D} \sum_{j=1}^{|\mathcal{N}|} \sum_{d=1}^{D} \left(\|(\boldsymbol{x}_j)_d\|_2^2 - \mu \right)^2, \qquad \mu = \frac{1}{|\mathcal{N}|D} \sum_{j=1}^{|\mathcal{N}|} \sum_{d=1}^{D} \|(\boldsymbol{x}_j)_d\|_2^2. \tag{10}$$

This term does not interfere with the prediction: the $\mathrm{Project}$ of Eq. equation 2 discards the modulus of $\boldsymbol{x}_j/\boldsymbol{x}_i$, so Eq. equation 7 depends only on the phases of the node embeddings. What $\mathcal{L}_{\text{reg}}$ removes is the spurious scale in the explanation score. This agrees with the ablation in Figure 4: dropping it leaves the prediction almost unchanged while weakening the explanation.

We then define the explanation of a query pair as the set of its path–score pairs, and take the path with the highest score as the explanation path, which by Assumption 1 is the path that provides the strongest evidence for the prediction:

$$\left\{ \left(P_{ij}^k, \mathrm{Score}(P_{ij}^k) \right) \right\}_{k=1}^{K}, \qquad P_{ij} = \underset{k \in \{1,\ldots,K\}}{\arg\max}\ \mathrm{Score}(P_{ij}^k). \tag{11}$$

Some explanation metrics, namely AUC and fidelity, do not use a single path but require a score for every edge of the subgraph from which the candidate paths are enumerated, i.e., of the union of the 2-hop subgraphs of the two query nodes; we write $\mathcal{E}_c$ for the edge set of this subgraph. These metrics therefore require a mapping $\mathrm{Score} : \mathcal{E}_c \to (0, 1)$, which we obtain by assigning to each edge $e_{ij} \in \mathcal{E}_c$ the score of the best path it belongs to:

$$\mathrm{Score}(e_{ij}) = \max_{k:\, e_{ij} \in P_{ij}^k} \mathrm{Score}(P_{ij}^k), \tag{12}$$

and we set $\mathrm{Score}(e_{ij}) = 0$ if e_{ij} does not belong to any of the enumerated paths P_{ij}^k.

Algorithm 1 in the appendix summarizes the training and the inference procedure.

4 Experiments

4.1 Datasets and Experimental Setup

Datasets. Statistics of the six datasets are summarized in Table 4 in the Appendix A.3.

Evaluation Metrics. For explanation, we report F1-score and Jaccard on FrenchRoyalty-200k and EuropeanRoyalty, and AUC on AugCitation and UserItemAttr; the two families of metrics differ because the ground truths differ in form. On the two royal datasets the ground truth is a set of paths, with which we compare the single path returned by the model. On AugCitation and UserItemAttr, and for the fidelity of WN18RR and Amazon, the metric is computed from the edge scores of Eq. equation 12, i.e., the score of the best path that contains the edge. WN18RR and Amazon provide no ground truth, so we report the fidelity of the explanation with respect to a trained RGCN, which is the same for every explainer so that all of them are evaluated against the same model; its definition is given in Appendix A.4. For prediction, we use accuracy (ACC) and AUC on all datasets.

Parameter Settings. The maximum path length is $M = 2$ on the two royal datasets, whose ground-truth explanations consist of paths of length at most 2, and $M = 3$ on the other four; the loss weight

of $\mathcal{L}_{\text{reg}}$ is $C = 0.1$ and the embedding dimension is $D = 256$ on all datasets. The remaining hyperparameters, including the weights $w_{ij}^{(l)}$ in Eq. equation 3 and the number of training epochs, are listed in Table 5 in Appendix A.5, and the number of layers L and the update ratio ϵ are studied in Appendix A.8.

Baselines. For explanation, we compare our model with GNNExplainer Ying et al. (2019), PGExplainer Luo et al. (2020), ExplaiNE Kang et al. (2019), PaGE-Link Zhang et al. (2023), Power-Link Chang et al. (2024), and KGExplainer Ma et al. (2025); these explainers are post-hoc and are mostly built on perturbing an RGCN Schlichtkrull et al. (2018). Our model works differently: it derives the prediction and the explanation from the same geometric relationship between the embeddings of the two query nodes, so the explanation has to be read together with the accuracy of the prediction it is derived from. We therefore also report prediction results, against RGCN and against RGAT Busbridge et al. (2019), CompGCN Vashishth et al. (2020), CosMIG Rao et al. (2022) and HL-GNN Zhang et al. (2024). All the baselines are reproduced with their official implementations on the same data splits as our model, and the implementations and the provenance of every reported number are given in Appendix A.6.

4.2 Interpretability Experiments

Table 1: Interpretability performance

	FrenchRoyalty-200k		EuropeanRoyalty		AugCitation	UserItemAttr	WN18RR	Amazon
	F1-score	Jaccard	F1-score	Jaccard	AUC	AUC	Fidelity	Fidelity
GNNExplainer	0.114	0.110	0.214	0.178	0.829	0.608	0.016	0.271
PGExplainer	0.103	0.095	0.202	0.192	0.586	0.578	0.020	<u>0.293</u>
ExplaiNE	0.182	0.174	0.242	0.206	0.464	0.507	0.111	0.097
PaGE-Link	<u>0.364</u>	<u>0.474</u>	<u>0.344</u>	<u>0.384</u>	0.966	0.956	0.120	0.134
Power-Link	0.296	0.309	0.239	0.251	<u>0.971</u>	<u>0.961</u>	<u>0.420</u>	0.188
KGExplainer	0.245	0.218	0.236	0.207	0.034	0.060	0.005	0.0007
Ours	**0.482**	**0.488**	**0.587**	**0.609**	**0.999**	**0.983**	**0.547**	**0.741**

The **bold** font highlights the best method. The <u>underlined</u> highlights the second-best method.

Table 1 reports the interpretability results. On the two royal datasets, which provide ground-truth explanations, KGE-Path clearly outperforms both the GNN-based explainers and the path-based baselines on F1-score and Jaccard. It also attains the best AUC on AugCitation and UserItemAttr and the best fidelity on Amazon and WN18RR. These results are consistent with our design: the explanation path is exactly the path whose composed relation embeddings best reproduce the relation between the two node embeddings that the prediction relies on. We further study how the quality of the explanation depends on the degree of the query pair in Appendix A.7.

4.3 Prediction Experiments

Table 2: Prediction Performance

	FrenchRoyalty-200k		EuropeanRoyalty		AugCitation		UserItemAttr		WN18RR		Amazon	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
RGCN	0.542	0.748	0.525	0.750	0.804	0.960	**0.925**	0.943	<u>0.688</u>	0.890	0.969	0.901
RGAT	0.554	0.798	0.546	0.796	0.846	0.979	<u>0.920</u>	0.985	0.678	<u>0.904</u>	0.907	0.968
CompGCN	0.559	0.714	0.500	0.767	0.846	0.980	0.881	**0.989**	0.437	0.871	0.968	<u>0.998</u>
CosMIG	<u>0.851</u>	<u>0.946</u>	<u>0.813</u>	<u>0.960</u>	<u>0.859</u>	<u>0.985</u>	0.911	<u>0.988</u>	0.504	0.886	0.900	0.969
HL-GNN	0.522	0.840	0.408	0.668	0.502	0.877	0.342	0.781	0.289	0.779	<u>0.975</u>	0.995
Ours	**0.972**	**0.981**	**0.982**	**0.991**	**0.865**	**0.988**	0.904	0.976	**0.705**	**0.930**	**0.989**	**0.999**

The **bold** font highlights the best method. The <u>underlined</u> highlights the second-best method.

Table 2 reports the prediction performance. KGE-Path achieves the best ACC and the best AUC on five of the six datasets. The only dataset on which our model is not the best is UserItemAttr (ACC 0.904, AUC 0.976).One explanation might be that UserItemAttr contains only 250 nodes and $1{,}800$ links, so the relational structure is very limited and the parameter-heavy GNN baselines can largely memorize the observed relation patterns from the training triples, whereas our model deliberately

relies on the geometric structure rather than on fitting capacity. Overall, the comparison shows that explaining does not come at the price of a weaker predictor.

4.4 Case Study

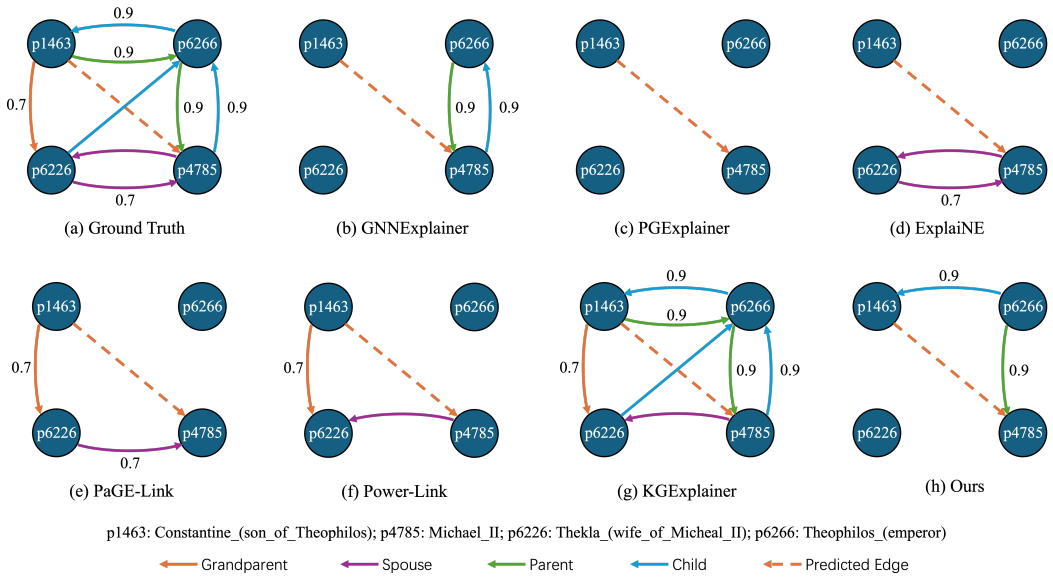


Figure 3: Case study on EuropeanRoyalty. The dashed edge is the query pair (Constantine, p_{1463}) $\rightarrow$ (Michael II, p_{4785}); the number next to an edge is the ground-truth explanation score of that edge, the colours encode the relation types, and the method of every panel is labelled in the panel. The ground-truth explanation of this pair consists of the two paths p_{1463}–p_{6266}–p_{4785} and p_{1463}–p_{6226}–p_{4785}; for the two mask-based explainers we show the two edges with the highest mask scores.

Figure 3 shows the explanation of a query pair of EuropeanRoyalty, the relation between Constantine and his grandfather Michael II, together with those of six baselines. Its ground-truth explanation consists of the two paths through Theophilos, whose edges carry the highest score 0.9, and through Thekla, whose edges carry the score 0.7, and KGE-Path returns the two edges of the stronger one. The baselines return instead a weaker route (ExplaiNE, PaGE-Link and Power-Link), individual edges that do not connect the query pair (GNNExplainer and PGExplainer), or an enclosing subgraph too coarse to separate the 0.9 evidence from the 0.7 evidence (KGExplainer). This example indicates that our path score is well aligned with the ground-truth importance of the paths.

4.5 Ablation Study

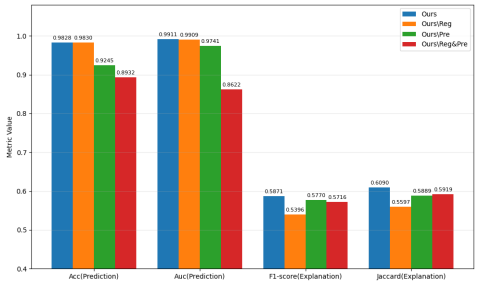


Figure 4: Ablation study for prediction and explanation on EuropeanRoyalty.

To investigate the effect of each component of our training, we conduct ablation studies on the EuropeanRoyalty dataset:

- Ours\Reg: The regularization term $\mathcal{L}_{\text{reg}}$ is removed.
- Ours\Pre: The adaptation for link prediction is removed, i.e., we train RotatE in the traditional way but keep the regularization term.
- Ours\Reg&Pre: Both components are removed, i.e., we purely train RotatE in the traditional way.

Figure 4 reports the result, where the two components play complementary roles. Removing $\mathcal{L}_{\text{reg}}$ leaves the prediction metrics almost unchanged but clearly weakens the explanation metrics: the term shapes the node embeddings so that the path score stays aligned with the relation embedding used by the prediction. Removing the training adaptation has the opposite effect, since the node penalty and the edge penalty are what steer the training towards link prediction: the explanation metrics stay close to those of the full model, whereas both prediction metrics drop noticeably. Removing both components gives the weakest prediction performance, and the explanation metrics also fall below those of the full model.

4.6 Efficiency Analysis

Table 3: Average time cost of KGE-Path and relevant methods per query pair, in seconds.

	FrenchRoyalty-200k	EuropeanRoyalty	AugCitation	UserItemAttr	WN18RR	Amazon
PaGE-Link	0.740	1.206	0.137	0.626	0.662	0.852
Power-Link	1.469	0.795	1.059	1.278	3.133	7.100
KGExplainer	5.851	17.065	78.541	12.564	0.164	2.439
Ours	0.034	0.041	0.472	0.062	0.013	1.481

We report the efficiency of our method in Table 3, which gives the average time cost of explaining one query pair. Our explanation is read off directly from the trained embeddings, so no additional training is needed at explanation time and the cost of a query pair is dominated by path enumeration on the pruned subgraph. Our method is the fastest on four of the six datasets; on AugCitation and Amazon, where both the computation graph and the enumeration space are large, it is still the second fastest after PaGE-Link, and clearly cheaper than Power-Link and KGExplainer. The gain over the baselines is largest on the datasets whose 2-hop subgraph is sparse, such as the two royal datasets and WN18RR, where the baselines still have to optimize a mask or a surrogate model for every explained instance. Our cost is governed by the density of the 2-hop subgraph and by the path-length limit M rather than by the size of the whole graph, which makes it predictable; the number of candidate paths, the time complexity and the effect of M are discussed in Appendix A.9.

5 Conclusion

In this paper, we proposed KGE-Path, an interpretable link predictor whose prediction and explanation are both read off the same geometric relationship between the embeddings of the two query nodes. Because the explanation is derived from the topological structure of the input graph and from the geometric relationship that the prediction itself relies on, it requires no additional training and faithfully reflects how the prediction is made. We further adapted the standard KGE training with an edge penalty and a node penalty on the geometric relationship, and with a regularization term that aligns the explanation with the prediction. Experiments on six datasets show that KGE-Path explains better than GNN- and KGE-based explainers and explains a query pair faster than all the baselines on four of the six datasets.

AI use statement

In this work, we used generative AI tools for the generation of code in the non-critical parts of our implementation, and for suggesting optimizations of a few critical parts of the code. We have not used generative AI tools for the model architecture design, for the running logic of the critical parts of the code, for the experimental design, or for the main ideas of the paper, and the remaining tasks that require disclosure are not applicable to this work. Additionally, we used generative AI

tools to assist authors in producing fluent English expressions. We have reviewed all AI-assisted work: all AI-generated code was executed and checked for correctness by the authors, and every AI-suggested optimization was verified to preserve the behaviour of the implementation. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility statement

Our code is available at `https://anonymous.4open.science/r/KGE-Path-114514`. The hyperparameter settings used on each dataset are listed in Table 5 in the appendix, which also indicates, for every dataset, the number of training epochs, whether the reversed edges are used, and how the training set is constructed. The datasets and the preprocessing steps are described in the Experiments section, and the fidelity metric is defined in Appendix A.4.

References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

Dan Busbridge, Dane Sherburn, Pietro Cavallo, and Nils Y Hammerla. Relational graph attention networks. *arXiv preprint arXiv:1904.05811*, 2019.

Heng Chang, Jiangnan Ye, Alejo Lopez Avila, Jinhua Du, and Jia Li. Path-based explanation for knowledge graph completion. *arXiv preprint arXiv:2401.02290*, 2024.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. *CoRR*, abs/1707.01476, 2017. URL `http://arxiv.org/abs/1707.01476`.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Francesco Ferrini, Veronica Lachi, Antonio Longa, Bruno Lepri, and Andrea Passerini. Gnns meet sequence models along the shortest-path: an expressive method for link prediction, 2025. URL `https://arxiv.org/abs/2507.07138`.

Jie Gao, Jia Hu, Geyong Min, and Fei Hao. Sgexplainer: Balanced path-based signed graph neural network explanation for link sign prediction. In *Proceedings of the ACM Web Conference 2026*, pp. 638–649, 2026.

Wenbin Guo, Xin Wang, Jiaoyan Chen, Lingbing Guo, Zhao Li, and Zirui Chen. Realm: Residual quantization bridging knowledge graph embeddings and large language models, 2025. URL `https://arxiv.org/abs/2510.09711`.

Nicholas Halliwell, Fabien Gandon, and Freddy Lecue. A simplified benchmark for ambiguous explanations of knowledge graph link prediction using relational graph convolutional networks. In *AAAI 2022-36th AAAI Conference on Artificial Intelligence*, 2022.

Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pp. 507–517, 2016.

Bo Kang, Jefrey Lijffijt, and Tijl De Bie. Explaine: An approach for explaining network embedding-based link predictions. *arXiv preprint arXiv:1904.12694*, 2019.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. Modeling relation paths for representation learning of knowledge bases, 2015. URL `https://arxiv.org/abs/1506.00379`.

Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *Advances in Neural Information Processing Systems*, 33:19620–19631, 2020.

Tengfei Ma, Wen Tao, Mufei Li, Jiani Zhang, Xiaoqin Pan, Yijun Wang, Bosheng Song, Xiangxiang Zeng, et al. Towards synergistic path-based explanations for knowledge graph completion: Exploration and evaluation. In *International Conference on Learning Representations*, volume 2025, pp. 55473–55493, 2025.

Jiahua Rao, Shuangjia Zheng, Sijie Mai, and Yuedong Yang. Communicative subgraph representation learning for multi-relational inductive drug-gene interaction prediction. *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, 2022. doi: http://dx.doi.org/10.24963/ijcai.2022/544.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, pp. 593–607. Springer, 2018.

Yong-Min Shin, Sun-Woo Kim, and Won-Yong Shin. Page: Prototype-based model-level explanations for graph neural networks, 2024. URL https://arxiv.org/abs/2210.17159.

Ye Sun, Lei Shi, and Yongxin Tong. expath: Explaining knowledge graph link prediction with ontological closed path rules. *Proceedings of the VLDB Endowment*, 18(9):2818–2830, 2025. URL https://arxiv.org/abs/2412.04846.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.

Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: Extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 990–998, 2008.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. *CoRR*, abs/1606.06357, 2016. URL http://arxiv.org/abs/1606.06357.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations, ICLR*, 2020.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases, 2015. URL https://arxiv.org/abs/1412.6575.

Zidu Yin, Yuankai Qi, Dong Gong, Ehsan Abbasnejad, Kun Yue, and Javen Qinfeng Shi. Link prediction on multi-relational graphs from an influence propagation perspective. *Pattern Recognition*, 180:114039, 2026. doi: 10.1016/j.patcog.2026.114039.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Juzheng Zhang, Lanning Wei, Zhen Xu, and Quanming Yao. Heuristic learning with graph neural networks: a unified framework for link prediction. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 4223–4231, 2024.

Shichang Zhang, Jiani Zhang, Xiang Song, Soji Adeshina, Da Zheng, Christos Faloutsos, and Yizhou Sun. Page-link: Path-based graph neural network explanation for heterogeneous link prediction. In *Proceedings of the ACM Web Conference 2023*, pp. 3784–3793, 2023.

A Appendix

A.1 Related Works

GNNs for link prediction. Early graph neural networks, such as GCN Kipf & Welling (2016) and GAT Veličković et al. (2017), were designed for semi-supervised node classification on graphs whose edges carry no type information. They propagate messages along edges without considering the relation types, so that two edges of different types that connect the same pair of nodes remain indistinguishable, and the information they can use for a prediction is bounded by the number of message-passing layers. To handle multi-relational graphs, RGCN Schlichtkrull et al. (2018) assigns a relation-specific transformation to every relation type, and RGAT Busbridge et al. (2019) makes the attention coefficients depend on the relation types. CompGCN Vashishth et al. (2020) composes entity and relation embeddings in a single message-passing update, CosMIG Rao et al. (2022) learns communicative representations of subgraphs for multi-relational prediction, and HL-GNN Zhang et al. (2024) unifies several heuristic link-prediction rules in a single GNN framework. These models make better use of the relation types. More recent work continues in this direction: SP4LP Ferrini et al. (2025) points out that node-centric message passing overlooks the subgraph connecting a pair of nodes, and runs a sequence model along the shortest path between a candidate pair, which is provably more expressive than message-passing GNNs; IGNP Yin et al. (2026) models the relation between two nodes as influence propagation, compresses the subgraphs that carry this propagation into virtual edges, and uses the result to guide the prediction on multi-relational graphs. Both build on information that lies on the routes connecting the two endpoints, as our model does, but they feed it into a learned predictor, whereas we read both the prediction and the explanation off the same embeddings.

KGE models. KGE models learn an embedding for every entity and every relation type and score a triple by a function of the three embeddings. Most of them use a bilinear or a neural score function: DistMult Yang et al. (2015) uses a trilinear diagonal product, ComplEx Trouillon et al. (2016) extends it to complex-valued embeddings in order to model asymmetric relations, and ConvE Dettmers et al. (2017) applies a two-dimensional convolution over the reshaped embeddings. These models are trained by corrupting the head or the tail entity of a triple, and their score functions do not define an explicit geometric relationship between entities and relations that could be reused beyond training. Lin et al. Lin et al. (2015) represent a multi-hop relation path by the composition of the relation embeddings along it, and use the composed representation to improve link prediction; in this work, we use the same composition to explain a prediction rather than to make one. TransE Bordes et al. (2013) and RotatE Sun et al. (2019) are different in this respect: they interpret a relation as a translation or as a rotation in the embedding space and use the distance between the transformed query embeddings as the score function directly, so that an explicit geometric relationship is preserved throughout the whole prediction process. Our model builds on this property and adopts RotatE as its base model. More recently, KGE models have also been combined with large language models: ReaLM Guo et al. (2025) discretizes the embeddings of a pretrained KGE model into code sequences and integrates them into the vocabulary of an LLM, so that the model can reason with both structured and textual knowledge. Such hybrids improve link prediction, but how to explain what they predict is left open.

Explanations. Existing explainers for graph and knowledge-graph predictions are mostly post-hoc: they explain a fixed, already trained model through an additional optimization or training stage. GNNExplainer Ying et al. (2019) optimizes a soft mask over the edges (and node features) of the computation graph for each explained instance, while PGExplainer Luo et al. (2020) parameterizes the mask generator so that a mask can be produced without per-instance optimization. ExplaiNE Kang et al. (2019) explains the link predictions of network embedding methods by a set of edges obtained from a linear program, and PaGE-Link Zhang et al. (2023) searches for paths that support a link prediction on a heterogeneous graph and learns a weight mask for every test triple. For knowledge graphs, Power-Link Chang et al. (2024) and KGExplainer Ma et al. (2025) explain the predictions of KGE models by paths. Besides being post-hoc, these methods are designed for entity prediction or for the existence of a link rather than for the relation type of a given pair of nodes, so their explanation is decoupled from the way in which the prediction itself is made. Recently prototype-based explainers such as PAGE Shin et al. (2024) do couple prediction and explanation by predicting on prototypes, but the prototype projection incurs considerable time overhead. eXpath Sun et al. (2025) also explains the link predictions of KGE models with paths from the head to

Algorithm 1 KGE-Path: Training and Inference

Input: training knowledge graph $\mathcal{K}_{\text{train}}$, number of layers L, update ratio ϵ, path-length limit M, regularization weight C, total training epochs T, randomly initialized node embeddings $\{\boldsymbol{x}_i\}$, relation embeddings $\{\boldsymbol{r}_i\}$, and edge weights $\{w_{ij}^{(l)}\}$.

Output: trained node embeddings $\{\boldsymbol{x}_i\}$, relation embeddings $\{\boldsymbol{r}_i\}$, and edge weights $\{w_{ij}^{(l)}\}$.

Training:
for epoch $t = 1, \ldots, T$ **do**
  Set $\boldsymbol{x}_j^{(0)} \leftarrow \boldsymbol{x}_j$ for all j (the current trained embeddings).
  for layer $l = 1, \ldots, L$ **do**
    Update node embeddings $\boldsymbol{x}_j^{(l)}$ for all j by Eq. equation 3.
  end for
  Compute the node penalty $\mathcal{L}_{\text{node}}$ averaged over all nodes in $\mathcal{K}_{\text{train}}$ and layers.
  Compute the edge penalty $\mathcal{L}_{\text{edge}}$ on the layer-0 embeddings, averaged over all edges in $\mathcal{K}_{\text{train}}$.
  Compute the regularization term $\mathcal{L}_{\text{reg}}$ over all nodes in $\mathcal{K}_{\text{train}}$.
  Set $\mathcal{L} \leftarrow \mathcal{L}_{\text{node}} + \mathcal{L}_{\text{edge}} + C\,\mathcal{L}_{\text{reg}}$.
  Update $\{\boldsymbol{x}_i\}$, $\{\boldsymbol{r}_i\}$, and $\{w_{ij}^{(l)}\}$ via backpropagation.
end for
Inference: given a query node pair (x_i, x_j):
Extract the union of the 2-hop subgraphs of x_i and x_j, and apply k-core pruning to obtain the search subgraph $\mathcal{G}_{ij}$.
Enumerate all simple paths of length at most M in $\mathcal{G}_{ij}$, i.e., $\{P_{ij}^1, \ldots, P_{ij}^K\}$.
for $k = 1, \ldots, K$ **do**
  Compute the path score $\text{Score}(P_{ij}^k)$ by Eq. equation 8.
end for
Select the explanation path $P_{ij} \leftarrow \arg\max_k \text{Score}(P_{ij}^k)$.
Predict the relation $\hat{r}$ by Eq. equation 7.

the tail, but it obtains and evaluates these paths with ontological closed path rules rather than from the embeddings of the explained model itself, so it is different from our work. SGExplainer Gao et al. (2026) targets datasets whose edges are either positive or negative, so it does not fit our setting either.

A.2 KGE-PATH

Our model's algorithm is in Algorithm 1.

A.3 DATASETS

The statistics of our datasets are summarized in Table 4. FrenchRoyalty-200k and EuropeanRoyalty describe royal family relations, and their ground-truth explanations are generated by a logical reasoner Halliwell et al. (2022). AugCitation and UserItemAttr are the datasets used by PaGE-Link Zhang et al. (2023), where AugCitation is derived from ArnetMiner Tang et al. (2008). WN18RR is adapted by Dettmers et al. (2018) from the classical WN18 dataset Bordes et al. (2013), and Amazon is taken from He & McAuley (2016). In Table 4, the *#Link Test* column gives the number of test links, i.e., the number of query pairs, which we use to normalize the time cost in the efficiency analysis, and the last column indicates whether a dataset provides ground-truth explanations for its links, which determines the evaluation metrics we use. Some of our datasets contain multi-edges, i.e., a pair of nodes may be related by more than one relation type. Prediction still works in their presence. In the explanation stage, we allow two paths with the same node sequence but different relation type sequences when enumerating paths.

Table 4: Statistical information of the datasets.

DATASET	#NODE	#LINK	#LINK TEST	#LINK TYPE	W/ EXP
FRENCHROYALTY-200K	2,125	12,717	2,471	6	✓
EUROPEANROYALTY	6,959	38,691	3,952	4	✓
AUGCITATION	45,961	243,228	7,203	7	✓
USERITEMATTR	250	3,352	273	5	✓
WN18RR	40,943	93,003	3,134	11	
AMAZON	10,166	148,659	14,746	2	

W/ EXP DENOTES WHETHER THE DATASET PROVIDES GROUND-TRUTH EXPLANATIONS FOR LINKS.

A.4 FIDELITY

For the two datasets that do not provide ground-truth explanations, we evaluate an explanation by its fidelity with respect to an RGCN Schlichtkrull et al. (2018). Let Φ denote the trained RGCN, $\mathcal{G}_c$ the computation graph of a query pair, $\hat{r}$ the relation that Φ predicts for that pair on $\mathcal{G}_c$, and $\Phi(\mathcal{G})_r$ the score that Φ assigns to a relation r when it runs on a computation graph $\mathcal{G}$. We convert the explanation into a score for every edge of $\mathcal{G}_c$ by Eq. equation 12 and use that score as an edge mask: we write $\mathcal{G}_{c\setminus\exp}$ for the computation graph in which every edge $e_{ij} \in \mathcal{E}_c$ carries the weight

$$\mathrm{Weight}(e_{ij}) = 1 - \mathrm{Score}(e_{ij}), \tag{13}$$

so that an edge that the explanation deems important, i.e., an edge with a high score, is suppressed. The fidelity of the query pair is the drop of the score that Φ assigns to the predicted relation, which we average over all the query pairs of the dataset:

$$\mathrm{Fidelity} = \Phi(\mathcal{G}_c)_{\hat{r}} - \Phi(\mathcal{G}_{c\setminus\exp})_{\hat{r}}. \tag{14}$$

A larger value means that the explanation covers the evidence on which the prediction of the RGCN relies.

A.5 HYPERPARAMETER SETTINGS

Table 5: Hyperparameter Settings of our model.

	FrenchRoyalty-200k	EuropeanRoyalty	AugCitation	UserItemAttr	WN18RR	Amazon
Dimension D	256	256	256	256	256	256
Layer	2	2	2	2	4	4
Epsilon	0.7	0.9	0.5	0.5	0.9	0.5
Reverse	True	True	False	False	False	False
Negative Ratio	0.16	0.16	0.0	0.0	0.0	0.0
Train Epochs	10,000	10,000	10,000	10,000	20,000	10,000
Train Full Data	True	True	True	False	True	True
Rel Weight	Learnable	Learnable	Ones	Ones	Ones	Ones

Table 5 lists the hyperparameters of KGE-Path on each dataset. *Dimension* is the embedding dimension D, which we fix to 256 on all datasets. *Layer* and *Epsilon* are the number of layers L and the update ratio ϵ in Eq. equation 3, which are selected according to the parameter study in Figures 6 and 7. *Reverse* indicates whether we additionally include the reversed counterpart of every edge. To fully exploit the information of an edge in the inverse direction, we additionally include the reversed edge (x_j, x_i) for every edge (x_i, x_j) on the two royal datasets. The reversed edge is not a free parameter: it carries the conjugate of the same relation embedding, $\overline{\boldsymbol{e}_{ij}}$, which is exactly the embedding of the inverse relation. Since $\|\boldsymbol{e}_{ij}\|_2 = 1$ component-wise, we have

$$\boldsymbol{x}_j \odot \overline{\boldsymbol{e}_{ij}} = \boldsymbol{x}_j / \boldsymbol{e}_{ij} = \boldsymbol{x}_i \tag{15}$$

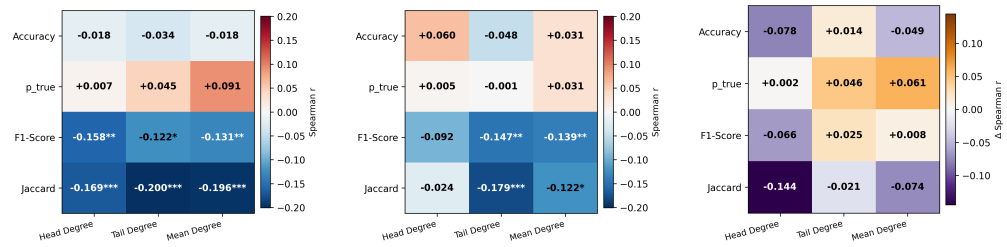
whenever Eq. equation 1 holds, so the propagation above already covers the inverse direction of an edge. *Negative Ratio* is the ratio between the number of negative triples and the number of positive triples used during training (for reference, the negative ratio of RotatE is 128.0). The negative triples are obtained by corrupting only the head or the tail entity of a positive triple, and this ratio is

non-zero only on the two royal datasets. *Train Epochs* is the number of training epochs, and *Train Full Data* indicates whether all triples other than the test triples are used as the training set: when it is true we fully exploit every non-test triple for training, whereas when it is false we keep the original training/test split, in which case the two splits do not cover all the edges and a few edges are left unused. Finally, *Rel Weight* denotes the weights $w_{ij}^{(l)}$ that the node penalty assigns to the different relation types in each layer: *learnable* means that they are trained, while *ones* means that they are fixed to 1 and are not trainable, in which case the neighbor aggregation in Eq. equation 3 is unweighted.

A.6 Reproducibility Details

The prediction baselines are reproduced with their official implementations, RGCN from DGL 2.4 and RGAT, CompGCN and CosMIG from PyG 2.3, on the same data splits as our model, and these four models use two layers and a 128-dimensional embedding. For the explainers, the results of PaGE-Link on AugCitation and UserItemAttr and those of Power-Link on AugCitation, UserItem-Attr and WN18RR are taken from the original papers, and the remaining explanation results are produced by running the code released by the original authors.

A.7 Degree Analysis



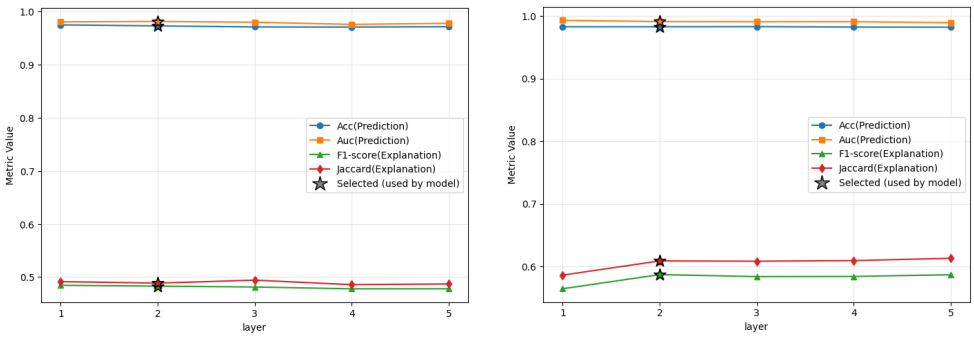
(a) Ours (b) Random initialized (c) Difference

Figure 5: Spearman correlation between the degree of the test triples and the quality of the prediction and of the explanation, on EuropeanRoyalty. The rows are the prediction metrics (accuracy, and the probability p_{true} that the model assigns to the ground-truth relation) and the explanation metrics (F1-score and Jaccard); the columns are the degree of the head node, the degree of the tail node, and their mean. (a) Our trained model. (b) A model with randomly initialized embeddings. (c) The difference between (a) and (b), which isolates the effect of training. Stars denote the significance of a correlation: $*$ for $p < 0.05$, $**$ for $p < 0.01$ and $***$ for $p < 0.001$.

Figure 5 relates the quality of the explanation to the degree of the query pair. For the trained model (a), neither the accuracy nor the probability p_{true} of the ground-truth relation correlates with the degree of either endpoint ($|r| \leq 0.1$), so the prediction is not systematically easier on well-connected pairs, whereas the F1-score and the Jaccard index are weakly but consistently negatively correlated with the degree ($-0.12 \leq r \leq -0.20$ with $p < 0.05$). This is consistent with the denser candidate set of a well-connected pair, in which more candidate paths compete with the correct one, and with the ground-truth explanations of such pairs containing more paths. Panel (b) shows that the negative correlation with the tail degree and with the mean degree is already present with randomly initialized embeddings, so this part of the dependence is a property of the neighborhood rather than an effect of training, whereas the dependence on the head degree becomes significant only after training; the differences in (c) are at most 0.15 in every cell. In other words, the degree of a query pair does affect the quality of the explanation, mildly but consistently ($|r| \leq 0.2$ with $p < 0.05$), whereas the prediction itself remains insensitive to the degree.

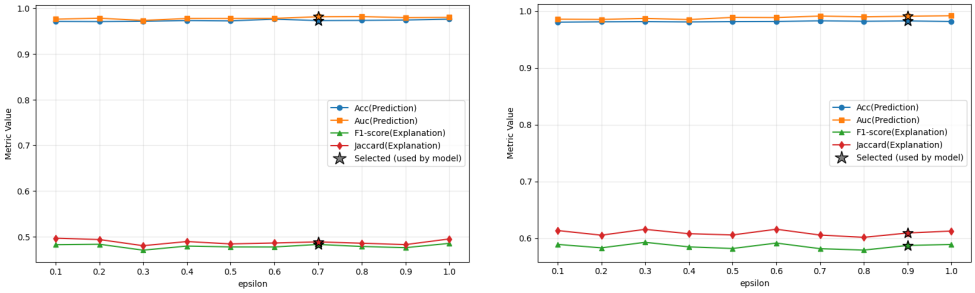
A.8 Parameter Experiments

We study the sensitivity of the model to the two hyperparameters it introduces, the number of layers L and the update ratio ϵ.



(a) FrenchRoyalty-200k (b) EuropeanRoyalty

Figure 6: The influence of the number of layers L on FrenchRoyalty-200k and EuropeanRoyalty.



(a) FrenchRoyalty-200k (b) EuropeanRoyalty

Figure 7: The influence of the update ratio ϵ on FrenchRoyalty-200k and EuropeanRoyalty.

Figure 6 shows the influence of the number of layers L. The prediction metrics are remarkably stable across $L \in \{1, \ldots, 5\}$ on both datasets, whereas the explanation metrics behave differently: they are almost flat on FrenchRoyalty-200k but increase from $L = 1$ to $L = 2$ on EuropeanRoyalty and remain nearly constant afterwards. This indicates that a small number of layers already provides sufficient information, and that stacking more layers does not bring further benefit while increasing the training cost. We therefore select $L = 2$ for both datasets, as marked by the stars in the figures.

Figure 7 shows the influence of the update ratio ϵ. Again, the prediction metrics are insensitive to ϵ and stay above 0.96 for all the values we tested, which suggests that the model does not rely on heavy neighbor aggregation and that the node penalty keeps the updated embeddings close to their original geometric positions in embedding space. The explanation metrics fluctuate mildly with ϵ and attain their best values at $\epsilon = 0.7$ on FrenchRoyalty-200k and at $\epsilon = 0.9$ on EuropeanRoyalty; we adopt these two values in the experiments.

A.9 Efficiency Details

Table 6 gives the average number of candidate paths per 2-hop subgraph after pruning. This number is small on the two royal datasets and on WN18RR, and much larger on AugCitation and Amazon, which matches the measured time in Table 3. Table 7 gives the time complexity of explaining one query pair, where $\mathcal{E}_c$ and $\mathcal{N}_c$ are the edges and nodes of the computation graph, i.e., the union of the 2-hop subgraphs of the query pair, and $\mathcal{E}_c^k$ and $\mathcal{N}_c^k$ are those after k-core pruning; M is the maximum path length, T the number of epochs for which a baseline trains its mask for a single query pair, and $|\Pi_{ij}|$ the number of candidate paths. Our cost comes from the two steps of the explanation: extracting the 2-hop subgraphs and pruning them with k-core costs $O(|\mathcal{E}_c|)$, and enumerating the paths and computing their scores costs $O(M|\Pi_{ij}|)$, since there are $|\Pi_{ij}|$ paths of at most M edges

each; adding the two gives $O(|\mathcal{E}_c|+M|\Pi_{ij}|)$, with no training loop. The three baselines are post-hoc, so their cost carries additionally the training factor T (Power-Link also the factor M, KGExplainer a term quadratic in $|\mathcal{N}_c|$), which does not appear in ours. This matches Table 3: our cost is at most 0.07 seconds per query pair on the two royal datasets, UserItemAttr and WN18RR, and rises to 0.47 and 1.48 seconds on AugCitation and Amazon, where both the computation graph and the candidate set are large, while UserItemAttr is the opposite case, with many candidate paths but a tiny computation graph.

Table 6: Average number of candidate paths per 2-hop subgraph after pruning.

	FrenchRoyalty-200k	EuropeanRoyalty	AugCitation	UserItemAttr	WN18RR	Amazon
AvgPath	9.4	9.8	124.2	200.1	8.0	1514.4

Table 7: Time complexity of explaining one query pair.

PAGE-LINK	POWER-LINK	KGEXPLAINER	OURS
$O(\|\mathcal{E}_c\| + \|\mathcal{E}_c^k\|T)$	$O(\|\mathcal{E}_c\| + \|\mathcal{E}_c^k\|MT)$	$O(\|\mathcal{N}_c\|(\log\|\mathcal{N}_c\| + \|\mathcal{N}_c\|T))$	$O(\|\mathcal{E}_c\| + M\|\Pi_{ij}\|)$