HyperKGR: Knowledge Graph Reasoning in Hyperbolic Space with Graph Neural Network Encoding Symbolic Path

Lihui Liu
Wayne State University
hw6926@wayne.edu

Abstract

Knowledge graphs (KGs) enable reasoning tasks such as link prediction, question answering, and knowledge discovery. However, real-world KGs are often incomplete, making link prediction both essential and challenging. Existing methods, including embedding-based and path-based approaches, rely on Euclidean embeddings, which struggle to capture hierarchical structures. GNN-based methods aggregate information through message passing in Euclidean space, but they struggle to effectively encode the recursive tree-like structures that emerge in multi-hop reasoning. To address these challenges, instead of learning static entity and relation embeddings, we propose a hyperbolic GNN framework (HYPERKGR) that embeds recursive learning trees in dynamic query-specific hyperbolic space. By incorporating hierarchical message passing, our method naturally aligns with reasoning paths and dynamically adapts to queries, improving prediction accuracy. Unlike static embedding-based approaches, our model learns context-aware embeddings tailored to each query. Experiments on multiple benchmark datasets show that our approach consistently outperforms state-of-the-art methods, demonstrating its effectiveness in KG reasoning. The code can be found in https://github.com/lihuiliullh/HyperKGR

1 Introduction

A knowledge graph (KG) is a structured representation of information that captures a collection of facts, where nodes correspond to real-world entities, events, or objects, and edges denote relationships between these nodes. Knowledge graphs have been widely applied in diverse domains, such as natural language understanding (Radford et al., 2018), question answering (Acharya and Adhikari, 2021) and so on, due to their ability to represent and query relational data effectively. Since their formal introduction in 2012,[1] various prominent knowledge graphs have been developed, including Freebase, Yago, and Wikidata, each offering rich resources for advancing research and applications in AI and data science.

Knowledge graph reasoning refers to the process of deriving new knowledge or insights from existing knowledge graphs by inferring missing facts or uncovering hidden patterns. This task is critical as real-world knowledge graphs are often large yet highly incomplete, presenting significant challenges in accurately predicting new facts. A key subtask within this domain is link prediction, which focuses on predicting the missing entity (tail or head) in a relational triple, such as $(u, p, ?)$ or $(?, p, v)$, where u and v are entities and p represents their relationship. The ability to infer such missing entities has vast implications for improving KG quality and enabling advanced reasoning in practical applications.

Over the past decade, numerous methods have been proposed to address knowledge graph reasoning. These include embedding-based approaches, which represent entities and relations as low-dimensional vectors in Euclidean space (Sun et al., 2019; Zhang et al., 2019); path reasoning methods, which explore multi-hop relational paths in the graph (Yang et al., 2017; Das et al., 2017); and, more recently, graph neural network (GNN)-based techniques, which aggregate information from local graph neighborhoods to model relational dependencies (Zhang and Yao, 2022; Zhang et al., 2023; Zhu et al., 2021). However, most existing methods focus on reasoning in embedding space without fully exploiting the hierarchical or structural characteristics inherent in many KGs. For hyperbolic knowledge graph link prediction tasks (Nickel and Kiela, 2017b; Chami et al., 2020; Balažević et al., 2019), while current methods embed entire KGs into hyperbolic space to capture hierarchical structures, their performance is hindered by the non-tree-like nature of most KGs and the presence of

[1] https://en.wikipedia.org/wiki/Knowledge_graph

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 25177–25188
November 4-9, 2025

weak or inconsistent hierarchies among entities. For instance, relations like `isFriend` are symmetric rather than hierarchical, which introduces noise into the learned embeddings.

Motivated by the strengths and limitations of existing approaches, we propose a novel method (HYPERKGR) that integrates path reasoning into the architecture of GNN-based KG reasoning models. Unlike previous hyperbolic embedding methods, which embed all entities and relations into a fixed hyperbolic space, our approach embeds the entire message-passing tree in hyperbolic space corresponding to a specific query $(u, p, ?)$. This design enables hierarchical message passing, which aligns naturally with the tree-like structures often observed in reasoning paths. Furthermore, instead of using static entity and relation embeddings, our model generates query-specific hyperbolic embeddings for entities and relations. This means that for a given query , the embeddings are dynamically computed, adapting to the specific reasoning context. Our extensive experiments on various datasets and scenarios demonstrate that the proposed method consistently outperforms state-of-the-art techniques, showcasing its effectiveness and robustness. More specifically, we make the following contributions:

1. We introduce a novel hyperbolic GNN-based framework that embeds hierarchical message-passing structure into hyperbolic space and dynamically generates query-specific hyperbolic embeddings for entities.

2. We conduct comprehensive experiments on multiple benchmark datasets, demonstrating significant performance improvements over existing methods.

2 Problem Definition and Preliminaries

Problem Definition. A *Knowledge Graph* (KG) is a structured representation of knowledge in the form of entities and relationships. Formally, a KG is defined as a directed multi-relational graph $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{F})$, where $\mathcal{V}$ is the set of entities, $\mathcal{R}$ is the set of relation types, and $\mathcal{F} \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$ is the set of facts represented as triples (u, r, v), where $u, v \in \mathcal{V}$ and $r \in \mathcal{R}$. Given a query in the form of a triple $(u, q, ?)$ or $(?, q, v)$, the goal of link prediction is to identify the most likely entity to complete the query. This task is central to improving the completeness and utility of knowledge graphs by inferring missing information. In this work, we aim to learn effective representations of entities and relations to improve the accuracy and efficiency of link prediction.

2.1 Preliminaries

Hyperbolic Space. The n-dimensional hyperbolic space, denoted by H^n, is a Riemannian manifold with constant negative curvature. It differs fundamentally from Euclidean space $\mathbb{R}^n$ (zero curvature) and the hypersphere S^n (positive curvature). Hyperbolic space exhibits unique geometric properties, such as exponential growth of volume and the absence of global compactness, making it suitable for modeling hierarchical and tree-like structures. Hyperbolic space can be represented using various models, each isometric and describing the same underlying geometry. In this work, we focus on the Poincaré ball model, which is particularly useful for computational purposes.

Poincaré Ball Model. The Poincaré ball model represents the n-dimensional hyperbolic space as:

$$D^n = \{x \in \mathbb{R}^n : \|x\| < 1\},$$

where $\|x\|$ is the Euclidean norm. The Riemannian metric is given by:

$$g^D(x) = \lambda_x^2 g^E, \quad \lambda_x = \frac{2}{1 - \|x\|^2},$$

where g^E is the Euclidean metric tensor. The geodesic distance between two points $x, y \in D^n$ is:

$$d_D(x, y) = \operatorname{arccosh}(1 + 2\frac{\|x - y\|^2}{(1 - \|x\|^2)(1 - \|y\|^2)})$$

This model is commonly used for hyperbolic embeddings in machine learning, as it allows for efficient computation and manipulation of hyperbolic distances.

Hyperbolic Embeddings. Hyperbolic embeddings are used to represent data points (such as entities or relationships in a knowledge graph) in hyperbolic space. This is particularly useful for tasks like link prediction, where the underlying graph exhibits hierarchical or multi-relational structures. Hyperbolic geometry is well-suited to capture such structures due to its properties, such as the exponential growth of volume and the ability to model hierarchical relationships naturally.

Möbius Addition. A fundamental operation in hyperbolic geometry is Möbius addition, defined

for points $x, y \in D_c^n$ as:

$$x \oplus_c y = \frac{(1 + 2c\langle x, y\rangle + c^2\|y\|^2)x + (1 - c\|x\|^2)y}{1 + 2c\langle x, y\rangle + c^2\|x\|^2\|y\|^2}.$$

This operation allows for the combination of points in hyperbolic space while respecting its curvature, and is an essential component of hyperbolic embedding techniques.

Exponential and Logarithmic Maps. To bridge Euclidean and hyperbolic spaces, the exponential and logarithmic maps are employed. The exponential map $\exp_c^x$ maps a tangent vector $v \in T_x D_c^n$ to a point in D_c^n:

$$\exp_c^x(v) = x \oplus_c \tanh\left(\sqrt{c}\frac{\|v\|}{\sqrt{c}}\right)\frac{v}{\|v\|}.$$

The logarithmic map $\log_c^x$ provides the inverse operation:

$$\log_c^x(y) = \frac{2\sqrt{c}\,\text{arctanh}(\sqrt{c}\|-x \oplus_c y\|)}{\|-x \oplus_c y\|}(-x \oplus_c y).$$

In practice, the maps $\exp_c^0$ and $\log_c^0$ are often used for transitions between Euclidean vectors and the Poincaré ball representation.

2.2 From Path Learning to GNN

Link prediction is aimed at predicting the existence of a query relation q between a head entity u and a tail entity v. Path-based methods (Yang et al., 2017; Das et al., 2017) solve knowledge graph reasoning by looking at the paths between a pair of entities in a knowledge graph. The idea is that a local structure is encoded by counting different types of random walks (paths) from u to v. From a mathematical perspective, path-based methods aim to learn a function $f_q(u, v)$ to predict the triplet (u, q, v) based on all paths from entity u to entity v, which could be summarized as

$$f_q(u, v) = \bigoplus_{P \in P_{u,v}} f_q(P) \tag{1}$$

$$f_q(P = (e_1, r_1, e_2 ... e_{|P+1|})) = \bigotimes_{i=1}^{|P|} w_q(e_i, r_i, e_{i+1})$$

where $P_{u,v}$ denotes the set of paths from u to v, and $w_q(e_i, r_i, e_{i+1})$ is the learning function on triplet (e_i, r_i, e_{i+1}) respect to the relation q. The summation operator $\bigoplus$ and multiplication operator $\bigotimes$ are generalized operators. They could denote any reasonable operations or functions, such as summation, aggregation, matrix multiplication, element wise product and so on.

However, the problem of path based reasoning is that given any pair of entities (u, v), we need to find all paths connected to these two entities, which is very computationaly expensive. An alternative approach is to use dynamic programming to accelerate the process. The observation is that when evaluating (e_i, r_i, e_{i+1}) with different $e_i \in V$ but the same query $(u, r_q, ?)$, the neighboring edges $\hat{E}_u^\ell, \ell = 1, \ldots, L$ of u are shared. Thus, instead of enumerating each possible path for each $(u, q, e_?)$, the dynamic programming iteratively propagates the representations of i hops to compute the representations of $i + 1$ hops, which achieves a polynomial time complexity. Formally, let $f_q^{(i)}(u, v)$ be the representation of i-hops. the $i + 1$ hop representation can be written as:

$$f_q^{i+1}(u, v) = \bigoplus_{(e_i, r_i, v) \in \hat{E}_u^{i+1}} f_q^i(u, e_i) \bigotimes w_q(e_i, r_i, v) \tag{2}$$

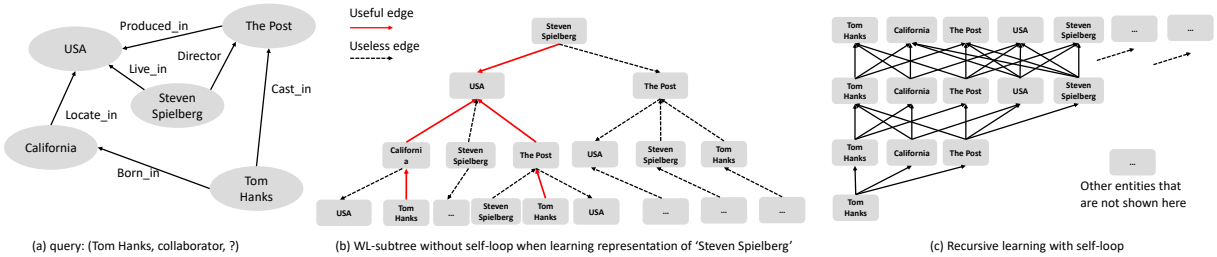
Once the representations of $f_q^i(u, e_i)$ for all the entities $e_i \in V_u^i$ in the i-th layer are ready, we can encode $f_q^{i+1}(u, v)$ by combining $f_q^i(u, e_i)$ with the edges $(e_i, r_i, v) \in \hat{E}_u^{i+1}$ in the $(i + 1)$-th layer.

Existing methods (e.g., (Zhang and Yao, 2022; Zhu et al., 2021)) interpret Equation 2 as a graph neural network (GNN) aggregation layer, where the general summation $\bigoplus$ is approximated by the GNN aggregation function, and the general multiplication $\bigotimes$ is approximated by the GNN message-passing function.

3 Methodology

In the previous section, we introduced the connection between path-based methods and graph neural networks. In this section, we present our method based on this connection. We show here that Equation 2 is equivalent to a GNN only when the aggregation and message-passing functions satisfy certain specific properties. Otherwise, substituting Equation 2 with a GNN introduces noise, which can degrade the model's performance or cause it to fail. This theory inspires the design of our method.

Theorem 1. *The dynamic programming update function in Equation 2 is equivalent to a graph neural network (GNN) aggregation layer only if the node e_i at the i-th layer receives and aggregates*



(a) query: (Tom Hanks, collaborator, ?) (b) WL-subtree without self-loop when learning representation of 'Steven Spielberg' (c) Recursive learning with self-loop

Figure 1: An example of answering the query (Tom Hanks, collaborator, ?). (a) The knowledge graph used; (b) The WL-subtree of the entity 'Steven Spielberg' without self-loops; (c) The recursive learning process with self-loops, where the number of entities in the i-th iteration grows dramatically based on the (i-1)-th layer.

information solely from nodes within the $(i-1)$*-hop neighborhood of the source node* u*, rather than from all neighbors of* e_i*.*

Proof. When a graph neural network (GNN) update function learns the embedding for a node v, the process can be represented as a Weisfeiler-Lehman (WL) subtree structure, as illustrated in Figure 1 (b). The message-passing function aggregates information from the $(i-1)$-th layer to the i-th layer. The original path function in Equation 1 describes all paths between the source node u and the target node v, represented by the thick red edges in Figure 1 (b). To ensure that Equation 2 functions equivalently to the GNN update function, the GNN must disregard all paths in the WL-subtree that are not part of the path set $P_{u,v}$, which are denoted as dashed edges. This requirement implies that when learning the embedding for node e_i at the i-th layer, the GNN must aggregate information exclusively from the thick paths while ignoring the dashed edges. Nodes on these thick paths belong to the intersection of the $(i-1)$-hop neighbors of u and the 1-hop neighbors of e_i. Thus, the GNN aggregation layer must ensure that information is propagated only through the relevant paths to maintain equivalence with Equation 2. □

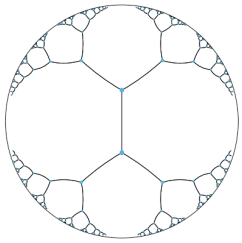


Figure 2: An example of hyperbolic tree embedding (image adapted from (Balažević et al., 2019)).

Theorem 3 demonstrates the connection between dynamic programming and graph neural networks. This suggests that dynamic programming in hyperbolic space is also equivalent to graph neural networks in hyperbolic space. More specifically, dynamic programming is used to recursively compute embeddings for the i-th neighborhood of the source node u before proceeding to compute embeddings for the $(i+1)$-th neighborhood of u, as illustrated in Figure 1 (c). However, when the graph becomes dense, the number of paths between nodes increases significantly. Consequently, the i-th layer will involve a large number of results, causing the number of nodes considered at the i-th layer to grow exponentially with their distance from the root of the tree. According to (Silva, 2022), the expected number of nodes within i hops from a source node u is approximately $(n-1)k^i p^i$, where n is the total number of nodes, k denotes the average degree of nodes, and p is the connection probability. [2]

With the explosion of the number of nodes needed to learn or update as the layer depth increases, naively using Euclidean embedding could not effectively distinguish these nodes, since its ability to model complex patterns is inherently bounded by the dimensionality of the embedding space (Nickel and Kiela, 2017a). On the other hand, hyperbolic space can be thought of as a continuous version of trees and as such it is naturally equipped to model hierarchical structures. For instance, it has been shown that any finite tree can be embedded into a finite hyperbolic space such that distances are preserved approximately (Nickel and Kiela, 2017a). See Figure 2 for an example. This inspires to utilize hyperbolic embedding, instead of Euclidean embedding, to learn the recursive tree representation. Thus, we generalize Equation 2 to hyperbolic space, where $\boxplus$ denotes the aggregation

[2]For additional details, see https://cs.rice.edu/~al110/teaching/mlg_s21_notes/lecture_5_network_science.pdf.

in the hyperbolic space, and $\boxtimes$ denotes the message passing in the hyperbolic space.

$$f_q^{i+1}(u,v) = \boxplus_{(e_i,r_i,v)\in\hat{E}_u^{i+1}} f_q^i(u,e_i) \boxtimes w_q(e_i,r_i,v) \tag{3}$$

3.1 Hyperbolic Embedding With Learnable Curvature

To effectively address the exponential growth of nodes with increasing path length or aggregation depth while simultaneously capturing hierarchical information, we employ hyperbolic embeddings to better encode the propagation tree structure. In hyperbolic geometry, the curvature $c < 0$ fundamentally influences the rate of expansion, making it particularly suitable for representing hierarchical or tree-like structures. By introducing a learnable curvature c, the embedding space can dynamically adjust to the graph's complexity, capturing varying hierarchical and relational structures. Let $h_u \in \mathbb{H}_c^d$ denote the embedding of a node in d-dimensional hyperbolic space with curvature c, and let $r \in \mathbb{H}_c^d$ represent a relation vector. The relation transition $(t = h + r)$ in Euclidean dynamic programming corresponds to a message-passing process in hyperbolic space, where the updated embedding h_v is computed as follows:

$$h_v = \text{proj}_{\mathbb{H}_c}\left(h_u \oplus_c r\right)$$

where $\oplus_c$ denotes Möbius addition parameterized by c, and $\text{proj}_{\mathbb{H}_c}$ ensures that the resulting embedding remains within the hyperbolic manifold. Specifically, in the Poincaré ball model, this projection is given by:

$$\text{proj}_{\mathbb{H}_c}(x) = \begin{cases} x, & \text{if } \|x\| < \frac{1}{\sqrt{-c}}, \\ \frac{x}{\|x\|} \cdot \frac{1}{\sqrt{-c}}, & \text{otherwise.} \end{cases}$$

The curvature c adapts the local geometry of the hyperbolic space to better fit the graph's structure. During training, the learnable curvature interacts with both the node embeddings and the relational parameters, allowing the model to optimize the space's geometry. This results in better modeling of complex hierarchical relationships and dynamic dependencies within the graph.

3.2 Hyperbolic Attention Aggregation

Given a query relation and a pair of entities, only some of the paths between the entities are important for answering the query. Consider the example in Figure 1 (a), the path `Tom Hanks` $\xrightarrow{\text{Born\_in}}$ `California` $\xrightarrow{\text{Located\_in}}$ `USA` $\xleftarrow{\text{Live\_in}}$ `Steven Spielberg` cannot determine whether `Steven Spielberg` is an answer to (`Tom Hanks`, `collaborator`, ?). On the other hand, path like `Tom Hanks` $\xrightarrow{\text{Cast\_in}}$ `The Post` $\xleftarrow{\text{Director}}$ `Steven Spielberg` is able to predict that (`Tom Hanks`, `collaborator`, `Steven Spielberg`) is true. To achieve this, we introduce attention in hyperbolic space to weight the importance of each path in the tree. Let h_s and h_r represent the embeddings of the subject entity and the relation in the hyperbolic space, respectively, and let h_{qr} denote the embedding of the query relation in the hyperbolic space. These embeddings are transformed into a shared tangent space followed by a linear transformation:

$$s_{\text{T}} = W_s log_c^0(h_s), r_{\text{T}} = W_r log_c^0(h_r)$$

$$qr_{\text{T}} = W_{qr} log_c^0(h_{qr})$$

where W_s, W_r, and W_{qr} are the respective linear transformation. The combined context embedding is computed as:

$$C_{\text{T}} = \text{ReLU}(s_{\text{T}} + r_{\text{T}} + qr_{\text{T}})$$

This embedding is then passed through a linear transformation with a sigmoid activation to obtain the attention coefficient α: $\alpha = \sigma(w_c^\top C_{\text{T}})$, where w_c is a learnable weight vector, and σ is the sigmoid function. The attention coefficient $\alpha \in (0, 1)$ reflects the relevance of the path relation to the query relation. Finally, the attention coefficient is used to scale the message from the relation path:

$$\text{message}_{u,v} = \alpha \cdot \text{proj}_{\mathbb{H}_c}\left(h_u \oplus_c r\right)$$

By weighting the message in this way, the model prioritizes relation paths that are more relevant to the query relation, improving its ability to capture meaningful interactions in reasoning tasks.

3.3 Multi-layer Updating

After obtaining the hyperbolic entity embeddings for each node in the l-th layer of the learning tree, the embeddings for nodes in the $(l+1)$-th layer are computed recursively using the same mechanism. However, a critical challenge in multi-layer propagation is the risk of information loss due to successive transformations, which may lead to forgetting previously learned representations. To mitigate this issue and maintain historical information along the

propagation path, we incorporate a gated update mechanism using a Gated Recurrent Unit (GRU). Specifically, after learning the hyperbolic embedding at each layer, the updated entity embeddings are processed through a GRU-based gating function. The update process is formulated as follows:

$$h_q(u,v)^{(l+1)} = \text{GRU}(h_q^{(l)}(u,v), h_0^{(l)})$$

where $h_q^{(l)}(u,v)$ denotes the entity embedding from the previous layer, and $h_0^{(l)}$ represents the memory vector. By leveraging this recurrent mechanism, the model effectively preserves long-range dependencies and prevents gradient vanishing, ensuring a more stable and expressive representation learning process across layers.

3.4 Link Prediction

After L layers' aggregation, the scoring function $f_q^L(u,v)$ can be calculated by

$$f_q^L(u,v) = \mathbf{w}^\top h_q^L(u,v)$$

During the training process, we utilize multi-class log-loss to optimize the parameters of the model. The loss function is defined as:

$$\mathcal{L} = -\sum_{(u,v)} \log \left(\frac{\exp(f_q^L(u,v))}{\sum_{(u,v'),v' \in \mathcal{V}} \exp(f_q^L(u,v'))} \right)$$

where the sum runs over all possible (u,v) pairs in the training set, and the objective is to maximize the score for correct triples while minimizing the score for incorrect ones.

3.5 Efficiency Improvement through Sampling

As shown in Figure 1(C), the recursive tree grows exponentially with the number of layers. However, for a given query $(u,q,?)$, only a subset of entities in the tree are relevant. To speed up the training process, an effective approach is to design a sampling strategy that prunes unnecessary branches early, improving efficiency. In (Zhang et al., 2023), a Connection-Preserving Incremental Sampling method is proposed, as illustrated in Figure 3. We can incorporate this sampling strategy into HYPERKGR to further enhance efficiency.

3.6 Time Complexity

One advantage of HYPERKGR is that it has a relatively low time complexity during inference. Consider a scenario where a model is required to infer the score of all possible triplets $f_q(u,v)$. We group triplets with the same condition u,q together, where each group contains $|V|$ triplets. For all node v at each group, we can learn their embeddings at the same time. Since a small constant number of iterations L is sufficient for HYPERKGR to converge, the algorithm has time complexity of: $\mathcal{O}(|\mathcal{F}|d + |\mathcal{V}|d^2)$, where d is the dimension of representations. Therefore, the amortized time complexity for a single triplet is: $\mathcal{O}\left(\frac{|\mathcal{F}|d+d^2}{|\mathcal{V}|}\right)$

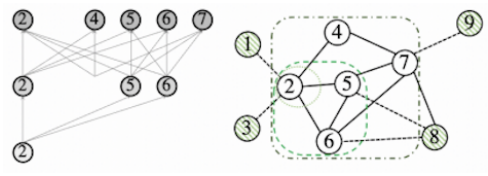


Figure 3: Connection-Preserving Incremental Sampling (image adapted from (Zhang et al., 2023)). This sampling method, originally proposed by AdaProp (Zhang et al., 2023), can be utilized to further enhance the performance of our model.

3.7 Theoretical Proof

Theorem 2. *The proposed* HYPERKGR *consistently outperforms or matches models based on dynamic programming or GNN with Euclidean embeddings. (Check Appendix for the formal proof)*

Proof. Euclidean embeddings correspond to a special case of HYPERKGR where the learned curvature approaches zero. As HYPERKGR generalizes Euclidean embeddings by allowing variable curvature, it is capable of capturing a broader range of geometric structures. Consequently, HYPERKGR consistently matches or surpasses the performance of models using dynamic programming or GNN with Euclidean embeddings. □

4 Experiments

We evaluate HYPERKGR in two different settings: the transductive setting and the inductive setting. In the transductive setting, we use the knowledge graphs FB15k-237 (Toutanova and Chen, 2015), WN18RR (Dettmers et al., 2018a), NELL (Xiong et al., 2017), Family (Kok and Domingos, 2007) and UMLS (Kok and Domingos, 2007). Following the experiment setting of (Zhang and Yao, 2022; Zhu et al., 2021; Zhang et al., 2023), for the inductive setting, we use the knowledge graphs FB15k-237, WN18RR, and NELL. We employ the standard transductive and inductive splits of these datasets. Detailed statistics of the datasets can be found in the Appendix.

Model	Family			UMLS			WN18RR			FB15k-237			NELL-995		
	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10
ConvE	-	-	-	.94	92.	96.	.43	39.	49.	.325	23.7	50.1	-	-	-
RotatE	.921	86.6	98.8	.925	86.3	99.3	.477	42.8	57.1	.337	24.1	53.3	.508	44.8	60.8
QuatE	.941	89.6	99.1	.944	90.5	99.3	.480	44.0	55.1	.350	25.6	53.8	.533	46.6	64.3
MuRE	-	-	-	-	-	-	.475	43.6	55.4	.336	24.5	52.1	-	-	-
MuRP	-	-	-	-	-	-	.481	44.0	56.6	.335	24.3	51.8	-	-	-
ATTH	-	-	-	-	-	-	.466	42.8	55.3	.324	23.6	50.1	-	-	-
UltraE	-	-	-	-	-	-	.488	44.0	55.8	.338	24.7	51.4	-	-	-
LorentzKG	-	-	-	-	-	-	.502	45.6	58.9	.384	28.7	57.9	-	-	-
MINERVA	.885	82.5	96.1	.825	72.8	96.8	.448	41.3	51.3	.293	21.7	45.6	.513	41.3	63.7
Neural LP	.924	87.1	99.4	.745	62.7	91.8	.435	37.1	56.6	.252	18.9	37.5	OOM	OOM	OOM
DRUM	.934	88.1	99.6	.813	67.4	97.6	.486	42.5	58.6	.344	25.2	51.6	OOM	OOM	OOM
RNNLogic	-	-	-	.842	77.2	96.5	.483	44.6	55.8	.344	25.2	53.0	-	-	-
CompGCN	.933	88.3	99.1	.927	86.7	99.4	.479	44.3	54.6	.355	26.4	53.5	OOM	OOM	OOM
NBFNet	.989	98.8	98.9	.948	92.0	99.5	.551	49.7	66.6	.415	32.1	59.9	.525	45.1	63.9
RED-GNN	.992	98.8	99.7	.964	94.6	99.0	.533	48.5	62.4	.374	28.3	55.8	.540	47.0	64.9
AdaProp	.988	98.6	99.0	.969	95.6	99.5	.562	49.9	66.1	.417	33.1	58.5	.542	47.9	64.4
HYPERKGR	**.992**	98.8	**99.7**	.970	95.7	99.1	.541	49.4	63.1	.366	27.3	55.3	.543	47.6	**65.1**
HYPERKGR + Sample	.990	**98.9**	99.1	**.978**	**96.5**	**99.5**	**.565**	**51.2**	**66.6**	**.417**	**33.1**	**58.5**	**.547**	**48.2**	64.8

Table 1: Transductive reasoning. '-' means unavailable results.

Methods	WN18RR				FB15k-237				NELL-995			
	V1	V2	V3	V4	V1	V2	V3	V4	V1	V2	V3	V4
RuleN	73.0	69.4	40.7	68.1	44.6	59.9	60.0	60.5	76.0	51.4	53.1	48.4
Neural LP	77.2	74.9	47.6	70.6	46.8	58.6	57.1	59.3	87.1	56.4	57.6	53.9
DRUM	77.7	74.7	47.7	70.2	47.4	59.5	57.1	59.3	87.3	54.0	57.7	53.1
GraIL	76.0	77.6	40.9	68.7	42.9	42.4	42.4	38.9	56.5	49.6	51.8	50.6
NBFNet	82.7	79.9	56.3	70.2	51.7	63.9	58.8	55.9	79.5	63.5	60.6	59.1
RED-GNN	79.9	78.0	52.4	72.1	48.1	61.9	60.3	62.1	84.6	60.1	59.4	55.2
AdaProp	85.3	83.6	61.0	74.3	54.6	65.0	62.3	63.4	88.6	62.3	59.7	**61.3**
HYPERKGR	79.5	78.2	53.1	71.6	48.2	62.1	60.5	61.8	77.6	55.2	58.8	55.4
HYPERKGR + Sample	**86.3**	**84.6**	**61.1**	**74.8**	**54.7**	**65.8**	**62.9**	**63.6**	**89.3**	**64.6**	**61.0**	60.8

Table 2: Inductive setting (evaluated with Hit@10).

Baselines. We compare HYPERKGR with path-based, embedding-based, and GNN-based methods. In the inductive setting, only path-based methods and GNNs are considered, as embedding methods cannot generalize to unseen entities. For the transductive setting, we compare HYPERKGR with (i) non-GNN methods, including ConvE (Dettmers et al., 2018b), QuatE (Zhang et al., 2019), RotatE (Sun et al., 2019), MINERVA (Das et al., 2017), DRUM (Sadeghian et al., 2019), and RNN-Logic (Qu et al., 2020); and (ii) GNN-based methods, including CompGCN (Vashishth et al., 2020), NBFNet (Zhu et al., 2021), RED-GNN (Zhang and Yao, 2022), and AdaProp (Zhang et al., 2023) and (iii) hyperbolic embedding methods (Balažević et al., 2019; Chami et al., 2020; Fan et al., 2024; **?**). For the inductive setting, we evaluate against RuleN (Meilicke et al., 2018), Neural LP (Yang et al., 2017), DRUM, GraIL (Teru et al., 2020), NBFNet, RED-GNN, and AdaProp. Results for RED-GNN and AdaProp are obtained using their official source code, while others are taken from original papers or reproduced with official implementations. Following prior work (Zhang et al., 2023), we evaluate performance using mean reciprocal rank (MRR), Hit@1, and Hit@10, where higher values indicate better performance.

Hyper-parameters. We tune the learning rate in $[10^{-4}, 10^{-2}]$, weight decay in $[10^{-5}, 10^{-2}]$, dropout rate in $[0, 0.3]$, batch size in $\{5, 10, 20, 50, 100\}$, embedding dimension d in $\{32, 48, 64, 96\}$, attention dimension d_α in $\{3, 5\}$, number of layers L in $\{3, 4, 5\}$, and activation function δ in $\{\text{identity}, \tanh, \text{ReLU}\}$. Adam (Kingma and Ba, 2017) is used as the optimizer. The best hyperparameter settings are selected based on the MRR metric on T_{val}, with a maximum of 50 training epochs.

4.1 Transductive Reasoning

Table 1 presents the results for transductive reasoning across five benchmark datasets. Among traditional embedding-based methods, QuatE and RotatE perform well, particularly on Family and UMLS, benefiting from their ability to capture relational patterns. Path-based approaches, such as MINERVA and Neural LP, generally lag behind, struggling with long-range dependencies, while DRUM achieves moderate performance. GNN-

based models, including CompGCN and NBFNet, demonstrate stronger results by leveraging message passing, with NBFNet excelling on WN18RR and FB15k-237 due to its more effective propagation mechanism. RED-GNN and AdaProp further improve performance, with AdaProp achieving the best results among baselines, particularly on WN18RR. Our proposed HYPERKGR outperforms all baselines on NELL-995 and performs competitively on other datasets. When enhanced with sampling (HYPERKGR + Sample), it achieves the best overall performance, surpassing all baselines on UMLS, WN18RR, and FB15k-237.

4.2 Inductive Reasoning

Table 2 presents the Hit@10 results for the inductive setting. Among traditional rule-based methods, DRUM and Neural LP achieve slightly higher scores than RuleN, demonstrating their effectiveness in capturing logical patterns. However, these approaches generally underperform compared to GNN-based methods, which better leverage structural information. GraIL, while effective in some cases, struggles on FB15k-237 and NELL-995 due to its limited expressiveness in capturing multi-relational dependencies. NBFNet, RED-GNN, and AdaProp achieve strong performance across all datasets, with AdaProp leading among existing baselines, particularly on WN18RR and NELL-995. Our proposed HYPERKGR performs competitively but lags slightly behind AdaProp, highlighting the challenge of fully leveraging hyperbolic space in the inductive setting. However, when combined with the sampling strategy (HYPERKGR + Sample), our method achieves the best overall results, outperforming all baselines on every dataset variant except for V4 in NELL-995.

4.3 Ablation Study

Table 3 presents the performance of different variants. We examine the effect of removing q_r from the attention mechanism (denoted as Attn-w.o.-q_r). Specifically, we eliminate the attention weight qr_T from the attention process. Since attention is responsible for identifying important edges, removing q_r reduces the informativeness of the learned structure, leading to weaker performance.

Methods	WN18RR		FB15k-237		NELL	
	MRR	H@10	MRR	H@10	MRR	H@10
Attn-w.o.-q_r	.673	79.1	.279	39.0	.530	74.6
HYPERKGR	.701	79.5	.340	48.2	.610	77.6
HYPERKGR+Sp	.722	86.3	.317	54.7	.661	89.3

Table 3: Performance comparison of different methods.

5 Related work

Embedding-based methods represent entities and relations as low-dimensional vectors, enabling efficient reasoning through similarity computations. Early models, such as TransE (Bordes et al., 2013) and DistMult (Yang et al., 2014), introduced translational and bilinear scoring functions, while more advanced methods like ComplEx (Trouillon et al., 2016) and TransR (Lin et al., 2015) improved multi-relational modeling by refining entity-relation interactions. Recent work has explored hyperbolic embeddings to better capture hierarchical structures. Unlike Euclidean spaces, hyperbolic spaces naturally encode tree-like relationships due to their exponential volume growth. Poincaré embeddings (Nickel and Kiela, 2017b) first demonstrated the effectiveness of hyperbolic geometry, which was later extended through models like Hyperbolic Graph Neural Networks (HGNNs)(Liu et al., 2019b). In knowledge graphs, MuRP(Balažević et al., 2019) and AttH (Chami et al., 2020) have leveraged hyperbolic spaces for improved link prediction, particularly in hierarchical settings.

In contrast, multi-hop reasoning approaches learn inference rules from relational paths within the knowledge graph, enabling explicit reasoning over multi-hop connections. Methods like Neural LP (Yang et al., 2017), DRUM (Sadeghian et al., 2019), and path-based techniques (Qu et al., 2020) generate logical rules from observed patterns. More recently, graph neural networks (GNNs) have been applied to multi-hop reasoning, allowing models like GraIL (Teru et al., 2020) and NBFNet (Zhu et al., 2021) to leverage structural information for improved reasoning.

6 Conclusion

In this paper, we introduced a novel hyperbolic GNN-based framework for knowledge graph reasoning, which embeds hierarchical message-passing structures into hyperbolic space and generates query-specific embeddings. Our approach incorporates dynamic, query-driven representations, which adapt to the specific reasoning context. Extensive experiments demonstrated that our method consistently outperforms state-of-the-art techniques in link prediction tasks, showing its effectiveness in capturing hierarchical dependencies and improving reasoning accuracy.

7 Ethical Considerations

We have carefully assessed the potential risks associated with our work and do not foresee any significant ethical concerns. Our framework is designed with a strong emphasis on usability and ease of implementation, lowering adoption barriers while minimizing operational complexities. Additionally, our research is built upon an open-source dataset, ensuring transparency, fostering collaboration, and promoting ethical integrity by providing accessible and reproducible data.

8 Limitations

Despite its effectiveness, our approach has certain limitations. First, our training data is limited in scope, primarily covering specific domains rather than offering broad generalization across diverse topics. This constraint may impact the model's performance when handling queries beyond these predefined areas. Furthermore, while our knowledge graph provides valuable contextual information, it remains inherently incomplete. Certain regions may lack sufficient data or relational links, potentially leading to gaps in the model's reasoning and inference capabilities.

References

A Acharya and S Adhikari. 2021. Alexa conversations: An extensible data-driven approach for building task-oriented dialogue systems.

Ivana Balažević, Carl Allen, and Timothy Hospedales. 2019. *Multi-relational poincaré graph embeddings*. Curran Associates Inc., Red Hook, NY, USA.

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

Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. 2020. Low-dimensional hyperbolic knowledge graph embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online. Association for Computational Linguistics.

R Das, S Dhuliawala, and M Zaheer. 2017. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018a. Convolutional 2d knowledge graph embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018b. Convolutional 2d knowledge graph embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

Xiran Fan, Minghua Xu, Huiyuan Chen, Yuzhong Chen, Mahashweta Das, and Hao Yang. 2024. Enhancing hyperbolic knowledge graph embeddings via lorentz transformations. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 4575–4589, Bangkok, Thailand. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Stanley Kok and Pedro Domingos. 2007. Statistical predicate invention. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 433–440, New York, NY, USA. Association for Computing Machinery.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29.

Lihui Liu, Yuzhong Chen, Mahashweta Das, Hao Yang, and Hanghang Tong. 2023. Knowledge graph question answering with ambiguous query. In *Proceedings of the ACM Web Conference 2023*.

Lihui Liu, Boxin Du, Yi Ren Fung, Heng Ji, Jiejun Xu, and Hanghang Tong. 2021. Kompare: A knowledge graph comparative reasoning system. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3308–3318.

Lihui Liu, Boxin Du, Jiejun xu, and Hanghang Tong. 2019a. G-finder: Approximate attributed subgraph matching. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 513–522.

Lihui Liu, Boxin Du, Jiejun Xu, Yinglong Xia, and Hanghang Tong. 2022. Joint knowledge graph completion and question answering. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 1098–1108, New York, NY, USA. Association for Computing Machinery.

Lihui Liu, Blaine Hill, Boxin Du, Fei Wang, and Hanghang Tong. 2024. Conversational question answering with language models generated reformulations over knowledge graph. In *Findings of the Association for*

Computational Linguistics: ACL 2024, pages 839–850, Bangkok, Thailand. Association for Computational Linguistics.

Lihui Liu, Zihao Wang, and Hanghang Tong. 2025. Neural-symbolic reasoning over knowledge graphs: A survey from a query perspective. *ACM SIGKDD Explorations Newsletter*, 27(1):124–136.

Qi Liu, Maximilian Nickel, and Douwe Kiela. 2019b. Hyperbolic graph neural networks.

Christian Meilicke, Manuel Fink, Yanjie Wang, Daniel Ruffinelli, Rainer Gemulla, and Heiner Stuckenschmidt. 2018. Fine-grained evaluation of rule- and embedding-based systems for knowledge graph completion. In *The Semantic Web – ISWC 2018: 17th International Semantic Web Conference, Monterey, CA, USA, October 8–12, 2018, Proceedings, Part I*, page 3–20, Berlin, Heidelberg. Springer-Verlag.

Maximilian Nickel and Douwe Kiela. 2017a. Poincaré embeddings for learning hierarchical representations.

Maximillian Nickel and Douwe Kiela. 2017b. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Meng Qu, Junkun Chen, Louis-Pascal Xhonneux, Yoshua Bengio, and Jian Tang. 2020. Rnnlogic: Learning logic rules for reasoning on knowledge graphs. *arXiv preprint arXiv:2010.04029*.

A Radford, J Wu, and R Child. 2018. Language models are unsupervised multitask learners.

Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. 2019. *DRUM: end-to-end differentiable rule mining on knowledge graphs*. Curran Associates Inc., Red Hook, NY, USA.

Arlei Silva. 2022. Machine learning with graphs.

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

Komal Teru, Etienne Denis, and Will Hamilton. 2020. Inductive relation prediction by subgraph reasoning. In *International Conference on Machine Learning*, pages 9448–9457. PMLR.

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

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR.

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

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. DeepPath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 564–573, Copenhagen, Denmark. Association for Computational Linguistics.

Yuchen Yan, Yongyi Hu, Qinghai Zhou, Lihui Liu, Zhichen Zeng, Yuzhong Chen, Menghai Pan, Huiyuan Chen, Mahashweta Das, and Hanghang Tong. 2024. Pacer: Network embedding from positional to structural. In *Proceedings of the ACM Web Conference 2024*, pages 2485–2496.

Yuchen Yan, Baoyu Jing, Lihui Liu, Ruijie Wang, Jinning Li, Tarek Abdelzaher, and Hanghang Tong. 2023. Reconciling competing sampling strategies of network embedding. *Advances in Neural Information Processing Systems*, 36:6844–6861.

Yuchen Yan, Lihui Liu, Yikun Ban, Baoyu Jing, and Hanghang Tong. 2021. Dynamic knowledge graph alignment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 4564–4572.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Fan Yang, Zhilin Yang, and William W Cohen. 2017. Differentiable learning of logical rules for knowledge base completion. *CoRR, abs/1702.08367*.

Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019. Quaternion knowledge graph embeddings. *Advances in neural information processing systems*, 32.

Yongqi Zhang and Quanming Yao. 2022. Knowledge graph reasoning with relational digraph. In *Proceedings of the ACM Web Conference 2022*, pages 912–924.

Yongqi Zhang, Zhanke Zhou, Quanming Yao, Xiaowen Chu, and Bo Han. 2023. Adaprop: Learning adaptive propagation for graph neural network based knowledge graph reasoning. In *KDD*.

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. 2021. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Advances in Neural Information Processing Systems*, 34.

9 Formal Proofs of Theorems

9.1 Theoretical Proof of Theorem 1

Below is the formal proof of Theorem 1.

Theorem 3. *The dynamic programming update function in Equation 2 is equivalent to a graph neural network (GNN) aggregation layer if and only if, at the i-th layer, the embedding of node e_i is updated by aggregating information exclusively from nodes in the set $\mathcal{N}(e_i) \cap \mathcal{N}^{(i-1)}(u)$, where $\mathcal{N}^{(i-1)}(u)$ denotes the $(i-1)$-hop neighborhood of the source node u.*

Proof. Let $G = (V, E)$ be a graph with source node $u \in V$. Define the following:

- $\mathcal{N}(v)$: the set of 1-hop neighbors of node $v \in V$,
- $\mathcal{N}^{(k)}(u)$: the set of nodes at exactly k-hop distance from u,
- $h_v^{(l)}$: the embedding of node v at layer l,
- $\mathcal{P}_{u,v}^{(i)}$: the set of paths of length i from u to v.

Dynamic Programming Update (Equation 2). The dynamic programming update computes the embedding of node e_i at layer i as:

$$h_{e_i}^{(i)} = \Psi\left(\bigoplus_{w \in \mathcal{R}_{u,e_i}^{(i)}} \Phi\left(h_w^{(i-1)}, h_{e_i}^{(i-1)}\right)\right), \quad (4)$$

where $\mathcal{R}_{u,e_i}^{(i)} = \mathcal{N}(e_i) \cap \mathcal{N}^{(i-1)}(u)$, Φ is a path-composition function, Ψ is a transformation function, and $\bigoplus$ is a permutation-invariant aggregator.

Standard GNN Aggregation. In contrast, a standard GNN update takes the form:

$$h_{e_i}^{(i)} = \phi\left(h_{e_i}^{(i-1)}, \bigoplus_{w \in \mathcal{N}(e_i)} \psi\left(h_{e_i}^{(i-1)}, h_w^{(i-1)}, e_{we_i}\right)\right), \quad (5)$$

where ϕ and ψ are learnable functions, and e_{we_i} denotes edge features.

Equivalence Condition. For Equation 4 and Equation 5 to be equivalent, the GNN's aggregation must be restricted to the same set $\mathcal{R}_{u,e_i}^{(i)}$. Furthermore, the function ψ, in combination with the edge features, must be able to simulate the behavior of Φ. That is,

$$\bigoplus_{w \in \mathcal{N}(e_i)} \psi\left(h_{e_i}^{(i-1)}, h_w^{(i-1)}, e_{we_i}\right) = \bigoplus_{w \in \mathcal{R}_{u,e_i}^{(i)}} \Phi'\left(h_w^{(i-1)}, h_{e_i}^{(i-1)}\right), \quad (6)$$

where Φ' is a reformulation of Φ using ψ and edge features.

Necessity. Assume there exists a node $w' \in \mathcal{N}(e_i) \setminus \mathcal{N}^{(i-1)}(u)$, i.e., a neighbor of e_i that is not reachable from u within $i-1$ steps. Then:

$$d(u, w') \neq i - 1 \Rightarrow d(u, e_i) \leq d(u, w') + 1 \neq i,$$

which implies that w' does not lie on any valid path of length i from u to e_i. Aggregating from such a node violates the fixed-length path structure enforced by the dynamic programming formulation, making the GNN update incompatible with Equation 4.

Sufficiency. Now assume aggregation is restricted to $\mathcal{R}_{u,e_i}^{(i)} = \mathcal{N}(e_i) \cap \mathcal{N}^{(i-1)}(u)$. Then for each $w \in \mathcal{R}_{u,e_i}^{(i)}$:

$$d(u, w) = i - 1 \text{and} \{w, e_i\} \in E \Rightarrow d(u, e_i) = i,$$

meaning that each w lies on a valid path of length i from u to e_i. Hence, the GNN update over $\mathcal{R}_{u,e_i}^{(i)}$ correctly mirrors the dynamic programming recurrence in Equation 4.

Conclusion. The update function in Equation 2 is equivalent to a GNN aggregation layer if and only if node e_i aggregates information solely from $\mathcal{N}(e_i) \cap \mathcal{N}^{(i-1)}(u)$. This constraint ensures that the GNN captures exactly the same path-based semantics as the dynamic programming formulation. □

9.2 Theoretical Proof of Theorem 2

Below is the formal proof of Theorem 2.

Theorem 4. *Let $\mathcal{G}$ be a class of query-answering problems defined over graphs, and let $\mathcal{M}_{Euc}$ denote models that perform reasoning using Euclidean embeddings, including GNN-based and dynamic programming-based methods in Euclidean space. Let* HYPERKGR *denote a model operating in a Riemannian manifold with learnable curvature c, such that $\mathbb{H}^d$ (hyperbolic space), $\mathbb{E}^d$ (Euclidean space), and $\mathbb{S}^d$ (spherical space) are all special cases.*

If HYPERKGR *is trained end-to-end with sufficient capacity and optimization, then for all tasks $T \in \mathcal{G}$, the expected generalization performance of* HYPERKGR *satisfies:*

$$\mathbb{E}[\mathcal{L}_T(\text{HYPERKGR})] \leq \mathbb{E}[\mathcal{L}_T(\mathcal{M}_{Euc})],$$

with equality if and only if the optimal embedding space for T is Euclidean.

Proof. The key idea is that HYPERKGR includes Euclidean models as a special case by allowing the curvature parameter $c \to 0$. More formally: (1) Any Euclidean embedding-based method corresponds to setting curvature $c = 0$ in a Riemannian manifold. (2) HYPERKGR optimizes c jointly with the embedding function, allowing it to adapt to the geometric structure of each task in $\mathcal{G}$.

Hence, HYPERKGR spans a superset of representational hypotheses compared to $\mathcal{M}_{\text{Euc}}$. Under the assumption of sufficient model capacity and optimization (e.g., universal approximation in Riemannian manifolds), HYPERKGR will recover or exceed the optimal performance achievable by $\mathcal{M}_{\text{Euc}}$, achieving strict improvement unless the target task inherently favors Euclidean geometry.

Thus, $\mathbb{E}[\mathcal{L}_T(\text{HYPERKGR})] \leq \mathbb{E}[\mathcal{L}_T(\mathcal{M}_{\text{Euc}})]$ holds, concluding the proof. □

10 Dataset

This paper focus on studying knowledge graph reasoning which is a special topic in AI (Liu et al., 2022, 2019a, 2021, 2023, 2024, 2025; Yan et al., 2021, 2023, 2024). We provide the statistics of entities, relations and split of triples in Table 4.

Table 4: Dataset statistics, where $|V|$ is the number of entities, $|R|$ is the number of relations, $|F|$ is the number of facts, $|T_{\text{val}}|$ is the number of validation triples, and $|T_{\text{tst}}|$ is the number of test triples.

Dataset	$\lvert V\rvert$	$\lvert R\rvert$	$\lvert F\rvert$	$\lvert T_{\text{val}}\rvert$	$\lvert T_{\text{tst}}\rvert$
Family	3,007	12	23,483	2,038	2,835
UMLS	135	46	5,327	569	633
WN18RR	40,943	11	86,835	3,034	3,134
FB15k-237	14,541	237	272,115	17,535	20,466
NELL-995*	74,536	200	149,678	543	2,818

11 Inference Time

Below is a preliminary experiment conducted to evaluate the inference time. Additional experimental results will be provided later.

Table 5: Inference running time (in seconds)

Dataset	**Max**	**Min**	**Average**
WN18RR	0.17	0.03	0.05
FB15k-237	0.28	0.04	0.17
NELL-995	0.21	0.04	0.11
YAGO3-10	0.47	0.15	0.37