Logical Query Reasoning over Hierarchical Knowledge Graph

Zijian Huang

School of Computer Science
The University of Auckland

Supervisor: Meng-Fen Chiang, Wang-Chien Lee

A thesis submitted in partial fulfilment of the requirements for the degree of MSc in Computer Science, The University of Auckland, 2021.

Abstract

Logical reasoning over Knowledge Graphs (KGs) for first-order logic (FOL) queries performs the query inference over KGs with logical operators, including conjunction ($\wedge$), disjunction ($\vee$), existential quantification ($\exists$) and negation ($\neg$), in order to closely approach true answers in embedding spaces. However, most existing work imposes strong distributional assumptions (e.g., Beta distribution) to represent entities and queries into presumed distributional shape, which clearly limits their expressive power. Moreover, query embeddings are challenging due to the relational complexities in multi-relational KGs, such as symmetry, anti-symmetry and transitivity. To bridge the gap, we propose a logical queries reasoning framework, **Lin**e **E**mbedding (**LinE**), for FOL queries. First, to relax the distributional assumptions, we introduce the logic space transformation layer, which is a generic neural function that converts embeddings from probabilistic distribution space to LinE embeddings space. Second, to tackle multi-relational and logical complexities, we formulate neural relation-specific projections and individual logical operators to truthfully ground LinE query embeddings on logical regularities and KG factoids. Lastly, to verify the LinE embedding quality, we generate a FOL query dataset from WordNet, which richly encompasses hierarchical relations. Extensive experiments show that LinE achieves considerable performance gain via both generalization reasoning and logical entailment settings on three benchmarks (Freebase, NELL, and WordNet) against existing strong baselines, particularly for multi-hop relational queries and negation-related queries.

Acknowledgements

I would like to first thank my thesis supervisor Meng-Fen Chiang. She always gives me useful suggestions when I ran into a trouble spot or had a question about my research or writing. Her insightful feedback pushed me to sharpen my thinking and brought my work to a higher level. I would also like to thank Wang-Chien Lee for helpful feedback and discussions. I am grateful to Michael Witbrock for providing the machine to run experiments. I am also grateful to the School of Computer Science at the University of Auckland for remote support. Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis.

Contents

List of Figures

List of Tables

Chapter 1

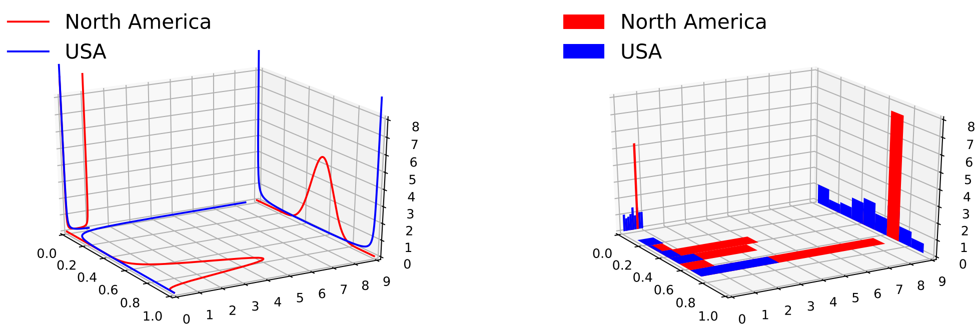
Introduction

Thanks to the availability of large-scale knowledge graphs (KGs), such as Freebase [8], WordNet [28], NELL [9], and YAGO [52], recent advances in knowledge graph representation learning have sparked significant research interests in logical query reasoning over multi-relational KGs. Understanding the relational properties in a collection of structured knowledge facts plays a pivotal role in the rapidly growing field of answering complex logical queries. Numerous efforts have been devoted to modeling logical operators or introducing new operators for first-order logical (FOL) queries on incomplete KGs [15, 19, 36, 37]. For instance, GQE [19] models both relational projections and set intersection operators by training neural networks as transformation functions. Apart from that, Query2Box [36] and BetaE [37] propose *box embeddings* and *beta embeddings*, two novel knowledge representations, to better formalize entities and logical queries in their respective representation spaces, while proposing a sophisticated attention mechanism to accurately capture set intersection behavior.

1.1 Research Challenges

Despite the recent progress, learning robust knowledge representations to better capture both relational and logical behavior, however, remains a challenging problem, with open issues in the following aspects: (i) expressive power on knowledge representations, (ii) preservation of closure properties under relational/logical operations, and (iii) support for both hierarchical and non-hierarchical logical query reasoning.

Firstly, most recent logical query reasoning models rely on some crucial assumptions on knowledge representations to enhance the *expressive power* of logical operators. GQE [19] formalizes entities and queries into a vector space, assuming that logical and relational behavior can be captured by a single value at each dimension. Query2Box [36] proposes box embeddings with centre and distance to box border to improve representation quality. BetaE [37] explores Beta distribution to improve representation quality, assuming that logical and relational



(a) Beta Representation Space

(b) LinE Representation Space

Figure 1.1: Comparison of Representation Space.

behaviors can be captured by a Beta distribution at each dimension in a distributional representation space. However, these strong assumptions lead to limitations in the expressiveness and violation tolerance of logical and relational behavior due to their relatively fixed shapes of assumed distribution. For example, the relational hierarchy in the triple ("North America", *contains*, "USA") is not truthfully reflected in the Beta space as shown in Figure 1.1(a), where there is no clear indication of one probability distribution ("North America" in red) encompasses the other ("USA" in blue). To obtain generic yet versatile representations to accurately capture logical and relational behavior, it is essential to equip the representations with better fault tolerance in logical and relational operations.

Secondly, preserving the *closure property* under relational and logical operations in knowledge representations is critical for enabling compositional computation of complex logical queries in the reasoning process. Most prior works only support a subset of logical operations. For example, region-based representations [36] do not preserve closure property under negation operation. Therefore, we need a knowledge representation that can comprehensively preserve the closure property under the logical and relation operations.

Lastly, most prior works tackle logical queries without considering relation hierarchy in KGs. Hierarchical relations, with anti-symmetry and partial-order transitivity, are intrinsic in KGs and thus are natural targets of logical queries. Different relational properties require different operations in the representation space to answer both logical queries with and without hierarchical relations. Nonetheless, relational annotations are usually not available to explicitly indicate a relation is hierarchical or not. We thus cannot trivially resort to supervised learning approaches to support multi-relational logical query reasoning.

1.2 Research Contributions

To overcome these challenges, we propose a novel KG reasoning framework based on **Lin**e **E**mbedding (**LinE**), for answering FOL queries over KGs. Specifically, to relax the distributional assumptions we propose to transform logical embedding from the Beta distribution space into a novel logic space, referred to as the LinE space, where we design competitive logical functions for logical operators while maintaining the closure property in the LinE space. To tackle multi-relational and logical complexities, we design an unsupervised learning approach to regulate query and KG entity embeddings in the LinE space, inspired by [10], using both curvature estimate and Krackhardt score to estimate the hierarchical relations. In addition, we rigorously generate a benchmark based on a hierarchical KG (WN18RR) for comprehensive study on reasoning ability of LinE framework. We conduct experiments over (i) the generalization reasoning and the logical entailment settings over 14 types of logical queries, and (ii) three benchmark datasets, including the Freebase, NELL, and WordNet which encompasses logical queries rich in hierarchical relations. In addition, we explore multiple formulations of logical operators and adopt the most competitive formulations. The main contributions are as follows.

- We propose a logical query reasoning framework (LinE) to preserve multi-relational complexities and logical regularities in the proposed LinE space. In particular, we propose logical space transformation and logical query inference to ground LinE embeddings to mixed relational and logical regularities.

- We design neural relation-specific projections to sensitively capture hierarchical and non-hierarchical relational properties in the KG guided by transitivity (*curvature* estimate) and anti-symmetry estimates (*Krackhardt* score).

- We generate a dataset of first-order logical queries, which heavily involve hierarchical relations from the benchmark KG WN18RR.

- In extensive experiments on three benchmark KGs, LinE shows better reasoning sensitivity to answer logical queries with and without hierarchical relations against dominant baselines.

1.3 Structure of Thesis

This thesis is structured into the following chapters.

- Chapter 2 - Related Work: We review key literature relevant to our research area. We discuss different existing logical reasoning models. We also present relevant work on multi-relational graph representation learning and knowledge graph question answering.

- Chapter 3 - Background: We introduce the problem of logical query reasoning and review key concepts relevant to our research area. We detail the notations and their description used in this thesis. Lastly, we formalize and define the research problem of logical query reasoning.
- Chapter 4 - Methodology: We propose a novel logical query reasoning framework, Line Embedding (LinE) model, to tackle the relational and logical complexities. Our framework transforms logical queries and entities between logical spaces, and performs logical reasoning on the proposed LinE representation space.
- Chapter 5 - We compare our framework with several dominant logical query reasoning methods and show that our framework outperforms these approaches across 14 logical query structures on three benchmarks.
- Chapter 6 - Conclusions: We conclude the thesis, discuss the limitations of our work and possible future directions.

Chapter 2

Literature Review

This chapter reviews current work on logical query reasoning and relevant research areas. Section 2.1 introduces the area of logical query reasoning. Section 2.2 reviews current knowledge graph representation learning approaches. Section 2.3 presents the most popular question answering methods over knowledge graphs.

2.1 Logical Query Reasoning

Section 2.1.1 presents the current popular logical representation space. Section 2.1.2 summarizes the current reasoning approaches that support various relational and logical operators. Section 2.1.3 reviews recent work on neural formulations for logical operators.

2.1.1 Representation Space

Logical query reasoning has been recently received growing interest, in particular, the class of existential first-order logical queries (EPFO) which includes the logical and existential operator. To answer complex logical queries over the KGs, some attempts have been made to formalize entities and queries as points [2, 19], or as regions [15, 36], or as distributions [37] in high-dimensional representation space.

GQE [19] embedded logical queries and entities in vector space, nonetheless GQE only supported conjunctive queries with $\exists$ and $\wedge$. CQD [2] formalized entities and queries using end-to-end differentiable objectives, with a pre-trained neural link predictor to compute the truth value of atoms. CQD supported queries with $\exists$, $\wedge$ and $\vee$ operators. Q2B [36] formalized entities and logical queries in the box representation space, supporting $\exists$, $\wedge$ and $\vee$ operators. Particularly, queries can be embedded as boxes. Likewise, a set of answer entities of the query refer to a set of points inside a box. Q2B handles $\exists$, $\wedge$ and $\vee$ operators by transforming queries into Disjunctive Normal Form (DNF). Q2B achieves 25% relatively improvement over prior work (GQE). HypE [15] formalizes entities as hyperboloids into Poincaré ball to better supports FOL queries except the negation operator. HypE defined queries as geometrical translation, intersection, and union. HypE explored hierarchical relationships and demonstrated its reasoning capability on e-commerce taxonomy. HypE outperforms the prior works such as GQE and Q2B on the logical query reasoning task. BetaE [37] formalizes entities and queries as Beta distributions. The closure property of Beta distribution enables BetaE to tackle FOL queries with $\exists$, $\wedge$, $\vee$, and $\neg$. BetaE tackled the negation operator which was not supported in the majority of prior work. BetaE shows its superior performance over all prior logical reasoning models. Recently, SMORE [35] reduces the training complexity to enable multi-hop reasoning over KGs [19, 36, 37] with faster training speed by reducing the complexity of training data via the bidirectional rejection sampling.

2.1.2 Relational and Logical Operators

Several attempts have been made to formalize entities and queries using different estimators [12,14,27,42,44,56]. GraIL [44] is a graph neural network based relation prediction framework that can generalize to unseen entities and graphs after the training process. EmQL [42] represented entity sets as centroid-sketch to support relational following, relational filtering, set intersection and union. EmQL tackled union operator via a normal form without changes in query structure. LogicE [27] combined query embeddings with the inductive bias of real-valued logic and also supports $\exists$, $\wedge$, $\vee$ and $\neg$. FuzzQE [12] provides a fuzzy logic based query embedding approach for answering FOL queries. The logical operators are defined in a principled and learning free manner. ConE [56] is a novel geometry-based embedding approach. ConE uses Cone Embedding to support all FOL operators, where the entities and queries are represented by Cartesian products of two-dimensional cones. PERM [14] embeds entity as a Multivariate Gaussian density to support $\exists$, $\wedge$ and $\vee$, by capturing its semantic position and smooth decision boundary.

2.1.3 Neural Reasoning Architectures

Others attempted to learn logic operators as neural modules for reasoning [11, 25, 41]. LINN [41] is a neural dynamic approach that applies deep neural network for logic reasoning. In particular, logic variables are represented as latent vectors and use neural modules that regularized by logic rules to represent logic operations, supporting $\wedge$, $\vee$ and $\neg$ operators. Similarly, NCR [11] integrates learning and reasoning to learn logic operators as neural modules for reasoning, supports $\wedge$, $\vee$ and $\neg$ as well. NewLook [25] mitigated the cascading errors exist in traditional embedding methods by modeling each logical operation as a learnable neural network. As a result, NewLook outperforms both subgraph matching based methods and embedding based methods due to the relaxation of the linear transformation assumption. Also, NewLook is more computationally efficient than the embedding based methods.

2.2 Multi-relational Graph Embeddings

Our work is highly relevant to existing efforts on multi-relational knowledge graph embeddings (KGE), which solve knowledge graph reasoning by learning entity and relation embeddings for a given KG in latent spaces. According to the recent theoretical understanding of knowledge graph embeddings, we divide prior work on KGE into four categories. Section 2.2.1 presents recent KGE approaches in Euclidean space. Section 2.2.2 reviews recent work on KGE in Hyperbolic space. Section 2.2.3 summarizes the current complex embeddings approaches. Section 2.2.4 describes neural network approaches for KGE.

2.2.1 Euclidean Embeddings

Extensive studies are devoted to representing KGs by Euclidean embeddings [8,21,24,32,50,53], which focus on accurately predicting missing links for a given KG in Euclidean space. Translational models are typical Euclidean KGE modes, which design score functions to consider Euclidean distance between the translated head entity embedding and the tail entity embedding. For example, TransE [8], TransH [50], TransD [21] and TransR [24] focus on the translation approach, which represents a relation as a translation operating on the low dimensional space. HAKE [55] defines entities into the polar coordinate system with translation techniques to model semantic hierarchies. RotL and Rot2L [49] simplify Möbius operations in RotH [10] and use two-layer stacked transformation to present a Euclidean-based RotH [10]. Both RotL and Rot2L deliver competitive performance and outperforms RotH with faster training speed respectively. This is because the calculation based on Hyperbolic geometry is much more complicated than Euclidean operations. Bilinear models are another typical Euclidean KGE modes, which represent relations as linear transformations acting on entity vectors. For example, RESCAL [32] and DistMult [53] use tensor factorization approach. However, these approaches cannot capture important relational properties such as anti-symmetry due to their simple translation techniques.

Another line of research on KGE is order embedding, which aims to encode the ordering properties among entities for a given KG into target representation space. Order embeddings capture relational transitivity effectively [3,13,46,47]. Order-embeddings [46] explicitly model the partial order structure of the visual-semantic hierarchy through encoding order into learned distributed representations. For example, [47] is a probabilistic extension of order embedding via box lattice. Density-order-embeddings [3] learns hierarchical representation as probability Gaussian distributions. To learn hierarchical probabilistic representations, Density-order-embeddings selects negative samples via simple loss functions, distance metrics and graph-based schemes, leading to superior performance on the WordNet *hypernym* relationship prediction task. However, order embeddings usually come with strong distributional assumptions, which may inevitably lead to critical limitations to provide proper logical reasoning.

2.2.2 Hyperbolic Embeddings

Recently, there are increasing interest in learning embeddings in Hyperbolic space [4, 10, 33, 34, 38]. The entity and relation embeddings of a given KG learned in Hyperbolic space are referred to as hyperbolic embeddings. Hyperbolic spaces have recently been proven as a theoretic foundation for the development of representations for hierarchical data due to the relatively few demanded dimensions [33, 34, 38]. As many real-world KGs consist of a set of entities, which can form different hierarchies under different relations, such as WordNet knowledge graph, hyperbolic embeddings have been considered an ideal embedding model in capturing all hierarchies simultaneously. For example, MuRP [4] embedded multi-relational graph data into Poincaré ball in hyperbolic space. MuRP proposed a hyperbolic translation technique for relations to obtain entity and relation embeddings via Möbius matrix-vector multiplication and addition operators. The geometric intuition behind is to determine the radius of a hypersphere decision boundary for a center entity so as to encourage tail entities related with the center entity via a relation r to fall into a learnable radius of the hypersphere. Next, a classic Bernoulli negative log-likelihood loss is formulated to learn hyperbolic embeddings based on a set of augmented positive and negative triples in KG. Lastly, MuRP demonstrated its superior performance against prior non-hyperbolic KGE models (e.g., TransE [8], ComplEx [45], RotatE [43]) via the knowledge graph link prediction task using standard benchmarks, including FB15k-237 and WN18RR. MuRP also proposed to use the Krackhardt score [23] as hierarchy estimates for relations. Another classic hyperbolic knowledge graph embedding families are RefH, RotH, and AttH [10], which capture hierarchical information by adopting both curvature estimate and Krackhardt score to estimate the relational hierarchy. For example, AttH combines hyperbolic reflections and rotations with attention to learn complex relational patterns. Particularly, AttH adopts trainable hyperbolic parameters to learn embeddings, whereby the accurate geometry for each relation can be learned and generalized. RefH and RotH is the variants of AttH only use reflection and rotation respectively.

2.2.3 Complex Embeddings

Many researchers have studied the problem of learning representations of entities and relations by introducing rotation operations in complex space in the knowledge graph embedding community [43, 45]. These studies focus on learning complex embeddings in order to capture important relational properties. For example, ComplEx [45] can handle symmetric and anti-symmetric relations through embed relations in complex space. ComplEx proposed to use vectors with complex value and Hermitian dot product for matrix and tensor factorization link prediction data. The computational cost of ComplEx remains linear growth even working with larger-scale datasets. RotatE [43] represents a relation as a rotation in the complex vector space, which captures important logical properties including symmetry, anti-symmetry, inversion, and composition. To train the model effectively and efficiently, RotatE proposed a self-adversarial negative sampling approach. RotatE demonstrates its superior performance against the prior state-of-the-art such as ComplEx [45] on several benchmarks. Nonetheless, operations in complex space typically require high representation costs due to the extra dimensions introduced in complex space.

2.2.4 Neural Network Embedding Models

Another line of KGE models focuses on advanced neural networks to derive entity and relation embeddings [5,17,30,31,57]. For example, ConvE [17] and ConvKB [31] employ a multi-layer convolutional network to learn entity and relation embeddings. ConvE specifically adopted a two-dimensional convolutional layer and a large fully connected layer to obtain relation-specific entity embeddings. Recently, some research works introduce Graph Neural Network (GNN) models to learn continuous edge embedding vectors for each edge type to handle an increasing number of relations [5,30,57]. For example, KBGAT [30] and A2N [5] embed entities and relations via an attention-based graph neural network. RGHAT [57] proposed a graph neural network with a hierarchical attention mechanism to utilize the neighborhood information of an entity. Specifically, RGHAT proposed a two-level hierarchical attention framework, which involves relation and entity relation to capture the neighborhood information of an entity. The hierarchical attention mechanism of RGHAT improves interpretability and close to human intuition. However, monolithic models, such as ConvKB or KBGAT, do not decompose into relation-specific embeddings. Although monolithic models are generally more flexible, they also require considerably excessive training costs. Moreover, it remains unclear whether monolithic models can achieve comparable performance compared to relation-specific models.

2.3 Knowledge Graph Question Answering (KGQA)

The problem of Question Answering over KG (KGQA) is a relevant research area to logical query reasoning. The goal of KGQA is to answer natural language queries over the knowledge graphs by reasoning over multiple triples in the KG to arrive at the target answer [39]. KGQA approaches in the presence of question complexities can be categorized into two: *simple questions*, and *complex questions*. A simple question refers to a question that involves a single head entity and a single predicate in the knowledge graphs [20]. A complex question on the other hand refers to the question that involves multiple entities and multiple relationships in the knowledge graphs [26]. Section 2.3.1 reviews the current approaches for simple question answering. Section 2.3.2 introduces the current works for complex question answering.

2.3.1 Simple Question

KGQA attracts lots of attention recently. Several works attempted to address answering simple questions over KGs [1, 6, 20, 54]. Typical KGQA approaches perform KG embedding to learn the low-dimensional representations first, and then perform the question answering task over KGs. For example, KEQA proposes to jointly learn the question's head entity, predicate, and tail entity representations in the KG embedding spaces. KEQA employed an attention-based bidirectional LSTM model to perform the predicate and head entity representation learning. Given a triplet, the joint distance metric is used to measure its distances to all candidate facts, and return the closest fact of three vectors as the target answer in the KG. Other than neural network approaches for KG embedding based question answering like STAGG [54] and KEQA [20], Aqqu [6] and QUINT [1] use automatically learned templates to provide interpretable answers to natural-language questions over KGs. In addition, QUINT provides visualization for the reasoning processes in arriving at the final answers. If an answer is determined as a false answer, the visualization offers valuable insights to reformulate the question based on its reasoning process.

2.3.2 Complex Question

Other than factoid based question answering, more works devoted to answering complex questions over KGs recently, which require multiple steps of reasoning for question answering [7, 26, 39]. For example, EmbedKGQA [39] uses knowledge graph embeddings to answer multi-hop natural language questions. EmbedKGQA firstly learns a representation for a given KG in a low-dimensional space. Next, EmbedKGQA learns a question embedding for a given question. Finally, EmbedKGQA combines KG embeddings and question embedding to predict the final answer. TextRay [7] and QUEST [26] attempt to challenge complex questions. For example, given a question QUEST [26] proposed to construct the quasi KG for the question on-the-fly and the perform graph algorithm for ranking candidate answers. Specifically, QUEST considers the entire Web as an external corpus to retrieve question-relevant documents and extract relevant triples from these documents. A quasi KG is also constructed based on these extracted triples so as to compute the graph-based ranking score in search for target answers.

Recently, some works [16, 29, 40] attempted to address the more challenging problem of conversational QA. For instance, Convex [16] uses seen entities and predicates to infer missing or ambiguous parts of the questions and answer incomplete questions over KGs. With the development of QA reasoning systems, more recent work ComQA [48] proposed compositional QA to capture segments of a document by employing graph neural networks. ComQA employs a hierarchical graph neural network to represent one document from word-level to sentence-level, and pre-train the model by using both question selection and node selection tasks. Other recent work [22] regards the reformulation in conversational QA via reinforcement learning.

Chapter 3

Preliminaries

Section 3.1 reviews key concepts for first-order logic query and relational properties. Section 3.2 introduces a broad range of logical query structures. Section 3.3 introduces the typical estimators to quantify the hierarchical degree of relations. Section 3.4 presents the classic probabilistic representation space, Beta distribution. Lastly, we formalize the problem of logical query reasoning over multi-relational KGs in Section 3.5.

3.1 Logical Query Reasoning

A knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{R})$ is a multi-relational graph, where $v \in \mathcal{V}$ represents an entity, and each relation type $r \in \mathcal{R}$ is a binary function $r : \mathcal{V} \times \mathcal{V} \to \{True, False\}$ that indicates the existence of a type-r directed edge between a pair of entities. A multi-relational KG $\mathcal{G}$ can be represented by a set of knowledge triples $\mathcal{K} \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$, which often exhibits multiple relational properties, such as symmetry, anti-symmetry and transitivity (Figure 3.1).

Hierarchical Relation. Relations can be divided into two categories, *non-hierarchical* and *hierarchical* relations, according to their relational properties. *Non-hierarchical* relations do not simultaneously exhibit anti-symmetric and transitive relational properties. For example, the relation *adjoin* in the triple ("England", *adjoin*, "Scotland") in Figure 3.1 is a *non-hierarchical* relation as the triple remains factually true after the exchange of positions between the subject and object entities. *Hierarchical* relations simultaneously exhibit anti-symmetric and transitive relational properties. The relation *contains* in the triple ("United Kingdom", *contains*, "England") in Figure 3.1 is an example of hierarchical relation as it is simultaneously anti-symmetric and transitive.

Hierarchical Knowledge Graph. KGs can be either *hierarchical* or *non-hierarchical* based on the relational properties in the graphs. *Hierarchical KGs* contain at least one hierarchical relation, while *non-hierarchical KGs* contain no hierarchical relation.

First-order Logic Queries. First-order logic (FOL) queries are queries formulated by logical

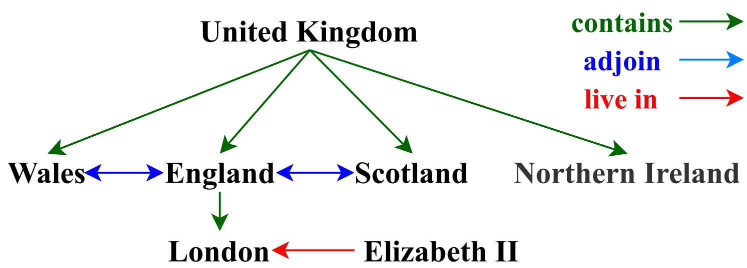


Figure 3.1: Examples of symmetric ('*adjoin*"), anti-symmetric ("*live in*") and transitive ("*contains*") relations on Freebase.

operators, including conjunction ($\wedge$), disjunction ($\vee$), existential quantification ($\exists$) and negation ($\neg$). An FOL query q can be expressed in disjunctive normal form (DNF), which is a disjunction of conjunctions as follows.

$$\begin{aligned} q[V_?] &= V_?.\, \exists\, V_1, \ldots, V_k : c_1 \vee c_2 \vee \ldots \vee c_n, \\ c_i &:= e_{i1} \wedge e_{i2} \wedge \ldots \wedge e_{im} \end{aligned} \tag{3.1}$$

where $V_?$ is the target variable, $\{V_i|1 \leq i \leq k\}$ are existentially quantified bound variables, and $\{c_i|1 \leq i \leq n\}$ are conjunction queries. Target variable $V_?$ indicates the final answer after the reasoning process is completed. Each existentially quantified bound variable V_i indicates the intermediate results during the reasoning process. A conjunction query c_i comprises of one or more atomic relation queries $\wedge_{j=1}^{m} e_{ij}$.

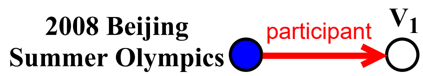
Atomic Relation Query. An atomic relation query e_{ij} is a relation projection between a pair of entity sets. Each entity set can be a non-variable entity, an intermediate variable, or a target variable. Formally, the atomic relation projection is defined as one of the following forms:

$$e_{ij} = r(v_a, V) \; or \; \neg r(v_a, V) \; or \; r(V^{'}, V) \; or \; \neg r(V^{'}, V), V \neq V^{'} \tag{3.2}$$

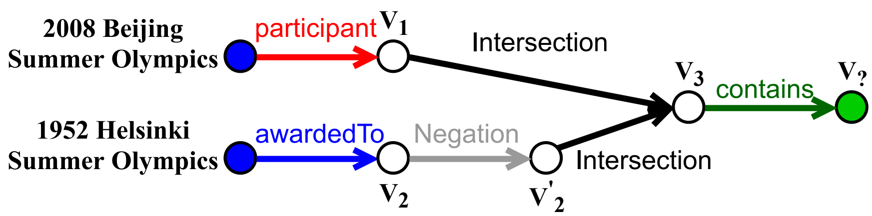
where $v_a \in \mathcal{V}_a$ is a non-variable anchor entity, $V \in \{V_?\} \cup \{V_i|1 \leq i \leq k\}$ is in the complete set of variables, $V^{'} \in \{V_i|1 \leq i \leq k\}$ is in the set of intermediate variables, and $r \in \mathcal{R}$ is a relation type. An example of atomic relation query q: "*list all participating countries of 2008 Beijing Summer Olympics*" is illustrated in Figure 3.2(a).

Compositional Relation Query. A compositional relation query comprises of multiple atomic relation queries in an FOL query q, which is typically structurized as a computation graph. Figure 3.2(b) illustrates the reasoning steps for the compositional relation query q: "*list all places of countries that participated in Beijing 2008 Summer Olympics and have not won any medals in Helsinki 1952 Summer Olympics*". Blue circles indicate the anchor entities, *"2008 Olympics"* and *"1952 Olympics Medal"*. Green circle indicates the target variable $V_?$ for the final answer to query q. Grey circles indicate the intermediate variables (V_1, V_2, V_3). q

$$q = V_?.\ \exists V_1, V_2, V_3 :\ participant(2008\ Olympics, V_1)$$
$$\wedge\ \neg awardedTo(1952\ Olympics\ Medal, V_2)\ \wedge\ contains(V_3, V_?)$$



(a) Atomic Relation Query



(b) Compositional Relation Query

Figure 3.2: Computation Graphs.

comprises of three atomic relation queries, *participant*, *awardedTo* and *contains*. To derive the answer, our reasoner firstly derives intermediate results for V_1 and V_2 from anchor entities. Next, our reasoner derives the complement of V_2, denoted as V_2', via a negation operator. Then, the reasoner derives the intersection of V_1 and V_2', resulting in V_3 ("Fiji") by an intersection operator. Finally, the reasoner derives the final answer for $V_?$ ("Suva") by a relational projection from V_3 via *contains* relation.

Hierarchical Logical Query. A logical query may involve multiple relations. We refer to the FOL queries that consist of at least one hierarchical relation ($\mathcal{R}_{\mathcal{T}}$) as *hierarchical logical queries*; whereas the FOL queries comprise of purely non-hierarchical relations ($\mathcal{R}_{\overline{\mathcal{T}}}$) are referred to as *non-hierarchical logical queries*.

3.2 Complex Query Structures

We follow [37] to examine FOL queries across 14 types of query structures. Figure 3.3 illustrates the 14 typical query structures considered in our experiments in computation graphs. To study the reasoning ability for diverse query types, we divide all query structures into three categories: *relation-heavy*, *logic-heavy*, and *negation-related* queries.

- **Relation-heavy**: queries that are purely tied to relation projections (1p/2p/3p)
- **Logic-heavy**: queries that are heavily tied to logical operations (2i/3i/ip/pi/2u/up)
- **Negation-related**: queries that involve negation operator (2in/3in/inp/pin/pni)

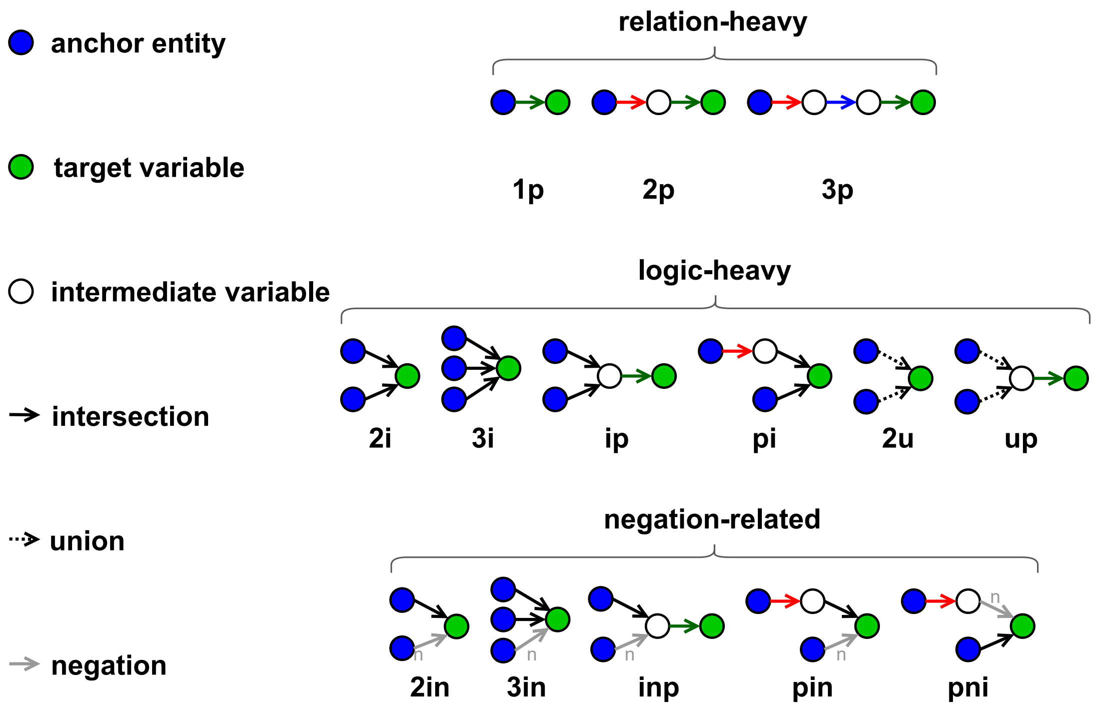


Figure 3.3: Illustration of the query structures for 14 query types with relation-specific projections (p), intersection (i), union (u) and negation (n) operations. Different relations are depicted in different colors (red/blue/green arrows).

3.2.1 Relation-heavy Query Structures

As illustrated in Figure 3.3, "1p" refers to the queries that require single relational projection from one anchor entity (blue node) as the sole reasoning step. "2p" refers to the queries that require two relational projections as the sequential reasoning steps: one relational projection for one anchor entity to generate intermediate results, followed by another relational projection from the previous intermediate results. "3p" refers to the queries that require three relational projections as the sequential reasoning steps: one relational projection for one anchor entity to generate intermediate results, second relational projection from the previous intermediate results, and third relational projection from the second intermediate results to generate the final target answer (green node).

3.2.2 Logic-heavy Query Structures

To illustrate, "2i" refers to the queries that require two relational projections for two anchor entities, followed by an intersection from the previous intermediate results to arrive at the target answer. "3i" refers to the queries that require three relational projections for three anchor entities, followed by an intersection from the previous intermediate results. "ip" refers to the queries that require the following reasoning steps: two relational projections for two anchor entities, followed by an intersection and another relational projection from the previous intermediate results to arrive at the target answer. "pi" refers to the queries that require the two parallel reasoning steps: (i) one relational projection for an anchor entity, followed by another relational projection; and (ii) one relational projection for another anchor entity. Finally, an intersection for previous intermediate results are preformed to derive the target answer. "2u" refers to the queries that require two relational projections for two anchor entities, followed by one union operation from the previous intermediate results to arrive at the target answer. "up" refers to the queries that require the following reasoning step: two relational projections for two anchor entities, followed by a union operation and another relational projection from the previous intermediate results.

3.2.3 Negation-heavy Query Structures

As illustrated in Figure 3.3, "2in" refers to the queries that require two parallel reasoning steps: (i) a relational projection for an anchor entities, and (ii) a complement operation of a relational projection for another anchor entity. Finally, an intersection from the above intermediate results are performed to derive the target answer. "3in" refers to the queries that require three parallel reasoning steps: two relational projections for two anchor entities, and a complement operation of a relational projection for last anchor entity. Finally, an intersection from the above intermediate results are performed to derive the target answer. 'inp" refers to the queries that require the following reasoning steps: a relational projections for an anchor entities, a complement operation of a relational projection for another anchor entity, followed by an intersection and another relational projection from the above intermediate results to arrive at the target answer. 'pin" refers to the queries that require two parallel reasoning steps: (i) a relational projection for an anchor entity, followed by another relational projection; and (ii) a complement operation of a relational projection for another anchor entity. Finally, an intersection for above intermediate results are performed to derive the target answer. 'pni" refers to the queries that require two parallel reasoning steps: (i) a relational projection for an anchor entity, followed by another relational projection and then take the complement; and (ii) a relational projection for another anchor entity. Finally, an intersection for above intermediate results are performed to derive the target answer.

3.3 Hierarchy Estimates

A hierarchy involves relations with anti-symmetry and transitivity properties [18, 23, 33], such as *contains*, *hypernym*, *has_part*. KGs are typically structured with mixed relations without explicit indications of hierarchical property. Therefore, we need to estimate the anti-symmetry and transitivity to distinguish hierarchical ($\mathcal{R}_{\mathcal{T}}$) from non-hierarchical relations ($\mathcal{R}_{\overline{\mathcal{T}}}$). Motivated by this, we explore two metrics, *Krackhardt hierarchy score* ($Khs_{\mathcal{G}_r}$) [23] and *curvature* ($\xi_{\mathcal{G}_r}$) [18], to estimate the degrees of anti-symmetry and transitivity for each relation $r \in \mathcal{R}$, respectively. A relation r is considered highly hierarchical if its induced relation graph $\mathcal{G}_r$ (i.e., the graph structured only with relation r) has higher anti-symmetry scores ($Khs_{\mathcal{G}_r}$) and higher transitivity scores ($\xi_{\mathcal{G}_r}$), and vice versa [18]. Our reasoner is guided by anti-symmetry scores and transitivity scores to learn respective relational regularities in KGs. Table 3.1 demonstrates an example of hierarchy estimates for WordNet benchmark. In total, there are seven out of 11 relations on WN18RR viewed as hierarchical relations according to their high anti-symmetry ($Khs_{\mathcal{G}_r}$) and negative transitive scores ($\xi_{\mathcal{G}_r}$).

3.3.1 Anti-symmetry Scores

The Krackhardt hierarchy score $Khs_{\mathcal{G}_r}$ [23] that captures the local anti-symmetry property of a relation graph $\mathcal{G}_r$ is defined as follows.

$$Khs_{\mathcal{G}_r} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} A_{i,j}(1 - A_{j,i})}{\sum_{i=1}^{n}\sum_{j=1}^{n} A_{i,j}} \tag{3.3}$$

where A is the adjacency matrix. $A_{i,j} = 1$ if there is an edge from node v_i to node v_j and 0 otherwise. $Khs_{\mathcal{G}_r}$ is in the range [0,1], where $Khs_{\mathcal{G}_r}$=0 if r is a fully symmetric relation and $Khs_{\mathcal{G}_r}$=1 if r is a fully anti-symmetric relation.

3.3.2 Transitive Scores

The curvature estimate $\xi_{\mathcal{G}_r}$ [18] that captures the global transitive behaviours for a given relation graph $\mathcal{G}_r$ is formally defined as follows.

$$\begin{aligned} \xi_{\mathcal{G}_r} = \frac{1}{|\Delta_{\mathcal{G}_r}|} \sum_{(v_i,v_j,v_k)\in\Delta_{\mathcal{G}_r}} \frac{1}{2\mathcal{D}_{\mathcal{G}_r}(v_i, v_m)} \big(\mathcal{D}_{\mathcal{G}_r}(v_i, v_m)^2 \\ +(\mathcal{D}_{\mathcal{G}_r}(v_j, v_k)^2)/4 - (\mathcal{D}_{\mathcal{G}_r}(v_i, v_j)^2 + \mathcal{D}_{\mathcal{G}_r}(v_i, v_k)^2)/2\big) \end{aligned} \tag{3.4}$$

where $\Delta_{\mathcal{G}_r}$ refers to a sample set of triangles from $\mathcal{G}_r$. $\mathcal{D}_{\mathcal{G}_r}$ is the shortest path distance for given node pair in $\mathcal{G}_r$. Given a triangle $(v_i, v_j, v_k) \in \Delta_{\mathcal{G}_r}$, we find the midpoint v_m of the shortest path connecting v_j to v_k to estimate how it structurally fits into the global topology. $\xi_{\mathcal{G}_r}$ is zero for triangles in lines, positive for triangles in circles, and negative for triangles in trees. Intuitively negative $\xi_{\mathcal{G}_r}$ suggests that r exhibits a strong transitivity.

Table 3.1: Hierarchical Knowledge Graph: WN18RR

Relation	$Khs_{\mathcal{G}_r}$	$\xi_{\mathcal{G}_r}$	Hierarchical
$memberMeronym$	1.00	-2.90	✓
$hypernym$	1.00	-2.46	✓
$hasPart$	1.00	-0.82	✓
instance hypernym	1.00	-0.78	✓
$memberOfDomainRegion$	1.00	-0.78	✓
$memberOfDomainUsage$	1.00	-0.74	✓
$synsetDomainTopicOf$	0.99	-0.69	✓
$alsoSee$	0.36	-2.09	✗
$derivationallyRelatedForm$	0.07	-3.84	✗
$similarTo$	0.07	-1.00	✗
$verbGroup$	0.07	-0.50	✗

3.4 Probabilistic Representation Space

In a probabilistic representation space, both the entities and queries S are viewed as probabilistic embeddings. For instance, BetaE [37] formulates a set of entities $S \subseteq \mathcal{V}$ as a Beta embedding, which is essentially a Beta distribution $\mathbf{B}(\cdot)$ shaped by two parameters α and β. The probability density function (PDF) controlled by (α,β) is defined as $p(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\mathbf{B}(\alpha,\beta)}$. An h-dimensional Beta embedding of an entity set S ($B^S \in \mathbb{R}^{2\times h}$) consists of h independent Beta distributions on the interval [0,1], denoted as $B^S = [(\alpha_1^S, \beta_1^S), \cdots, (_h^S, \beta_h^S)]$ with $x \in [0, 1]$. Note that a single entity is equivalently a set of a single element, and thus each entity itself in h-dimensional Beta distribution space is also an h-dimensional Beta embedding. A query q after executing logical operators ($\wedge$, $\vee$, $\exists$ and $\neg$) and relational projections ($r \in \mathcal{R}$) in the h-dimensional Beta representation space remains a h-dimensional Beta embedding, thanks for the closure property of BetaE. Nevertheless, BetaE is bounded by a strong distributional assumption, which limits representations into a presumed distributional shape. BetaE clearly limits its expressive power to (i) sensitively deal with mixed relational regularities (e.g., relational hierarchy), and (ii) represent entities and queries beyond Beta shapes. We therefore tackle the challenges of limited expressiveness by relaxing the distributional assumption and designing a more relation-sensitive representation space.

3.5 Problem Statement

Given a multi-relational KG $\mathcal{G} = (\mathcal{V}, \mathcal{R})$ and an FOL query q with relational and logical compositions, our goal is to enable *hierarchical logical query reasoning* of q over $\mathcal{G}$. Our approach is to design a novel neural logical reasoning framework that supports logical operators ($\wedge$, $\vee$, $\exists$ and $\neg$) and relation-specific projections ($\mathcal{R}$) for expressive specification of the FOL query (q); and return a set of entities ($V_? \subseteq \mathcal{V}$) that satisfy q over the facts captured by $\mathcal{G}$.

Table 3.2: The symbols and their descriptions used in this thesis.

Notation	Description
$\mathcal{K}$	a set of triples in knowledge graph $\mathcal{G}$=($\mathcal{V}$, $\mathcal{R}$)
$\mathcal{T}_{\mathcal{Q}}$	a set of training logical queries $\{q_1, q_2, \cdots, q_n\}$
$\mathcal{T}_{\mathcal{A}}$	a set of answers $\{q_1[V_?], q_2[V_?], \cdots, q_n[V_?]\}$ for $\mathcal{T}_{\mathcal{Q}}$
$\mathcal{R}_{\mathcal{T}}$ ($\mathcal{R}_{\overline{T}}$)	hierarchical (non-hierarchical) relations
$\mathcal{K}_{\mathcal{T}}$ ($\mathcal{K}_{\overline{T}}$)	triples bounded by hierarchical (non-hierarchical) relations
h	Beta embedding dimension
$S \subseteq \mathcal{V}$	an entity set
$q[V_?]$	a logical query with target answer variable $V_?$
$B^S \in \mathbb{R}^{2\times h}$	set S representation in h-dimensional Beta space
$L^S \in \mathbb{R}^{2\times h}$	set S representation in h-dimensional LinE space
$P^S \in \mathbb{R}^{2\times h}$	mean and variance of the Beta distribution for set S
$M^S \in \mathbb{R}^{k\times h}$	mode of the Beta distribution for set S
F^S	$[B^S]$, $[P^S]$, $[M^S]$, $[B^S, P^S, M^S]$

Chapter 4

Methodology

This chapter presents our **Lin**e **E**mbeddings model (**LinE**) that performs compositional relational projections and logical operations for logical queries. Section 4.1 highlights an overview of the proposed reasoning pipeline. In Section 4.2, we propose a logic space transformation (LST) layer to cover entities and queries from the source logic space to the proposed LinE representation space. In Section 4.3, we propose a logical query inference (LQI) component to the ground in knowledge triples and logical regularities in the LinE space to support compositional logical query inference. In Section 4.4, we formulate both *reasoning loss* and *query inference loss* to jointly optimize LinE embeddings and parameterized neural functions. In Section 4.5, we generate the QA dataset from WN18RR, which richly contains hierarchical relations in order to study the reasoning sensitivity for hierarchical logical queries.

4.1 Reasoning Pipeline Overview

The proposed reasoning model, LinE, supports a complete set of first-order logic operations and the relation-specific projections in the LinE space. Figure 4.1 illustrates the reasoning pipeline. Given a KG ($\mathcal{G}$), a set of query answering (QA) pairs as a training set $\mathcal{T}_{\mathcal{Q}} = \{q_1, q_2, \cdots, q_n\}$ and $\mathcal{T}_{\mathcal{A}} = \{q_1[V_?], q_2[V_?], \cdots, q_n[V_?]\}$, we train a logical query reasoner to answer FOL queries by performing relational projections and logical operations in order to closely approach the true answers in the LinE space. For each query q_i as input, its computation graph is generated to indicate the reasoning steps. We follow the reasoning steps to execute logical operators in the query and obtain KG entity embeddings in the BetaE space as the initial entity embeddings as shown in details in Figure 4.2. To enhance the expressive power, we propose a logic space transformation (LST) to project the initial entity embeddings from the BetaE space to the LinE space. The logical query inference (LQI) is proposed to refine the transformed embeddings for KG entities and query q_i by imposing multi-relational and logical regularities in the LinE space. Logical operations ($\wedge$, $\vee$, $\exists$ and $\neg$) and relation-specific projections ($r \in \mathcal{R}$)

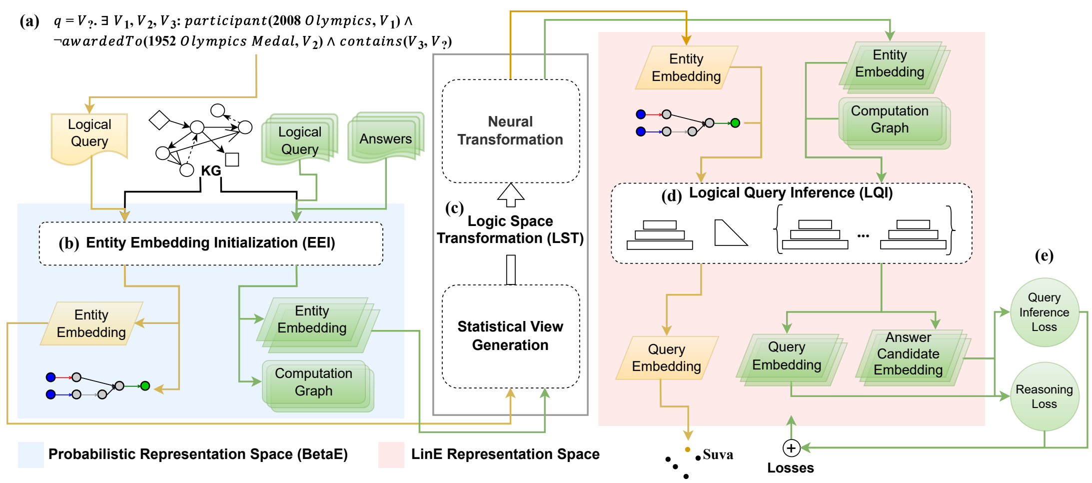


Figure 4.1: The Model Architecture of LinE. The green color indicates the training process, while the yellow color indicates the real-time testing process for unseen queries. (a) For training, LinE takes as input the KG and pairs of logical queries and answers. (b) Given the computation graph for each query, the entity embedding initialization (EEI) component generates the initial embeddings for KG entities in a probabilistic representation space (blue area). (c) We present the logic space transformation (LST) to better capture multi-relational and logical regularities. The statistical view of the initial entity embeddings is thus generated to learn the entity embeddings in the LinE representation space (LinE space). (d) The logical query inference (LQI) component learns a set of relation-specific neural functions and refines the embeddings for KG entities and queries based on multi-relational and logical regularities in the LinE space (pink area). (e) To optimize, two training objectives, query inference loss and reasoning loss, are proposed to jointly preserve multi-relational and logical regularities in the LinE space.

are mathematically formulated. To preserve the multi-relational and logical regularities, we formulate (i) *reasoning loss* to estimate the distances between the query embedding and answer candidates; and (ii) *query inference loss* to estimate relational violations and deviations from logical regularities, respectively. Lastly, the final answers are obtained by performing nearest-neighbor search (NNS) for the final query embedding and returning the closest entities ($V_? \subseteq \mathcal{V}$) to query q_i in the LinE space.

4.2 Logic Space Transformation

We introduce logic space transformation (LST), which transforms representations from a probabilistic representation space (source logic space) to the LinE space (target logic space) with enhanced expressive power. To this end, we propose a neural transformation from the source to target logic space to (i) preserve a diverse aspect of properties in Beta distribution; (ii) support multi-relational projections and logical operations in the LinE space. Figure 4.4 illustrates the details of the LST layer.

$$q = V_?.\ \exists\ V_1, V_2, V_3: participant(\mathbf{2008}\ Olympics, V_1) \land \neg awardedTo(\mathbf{1952}\ Olympics\ Medal, V_2) \land contains(V_3, V_?)$$

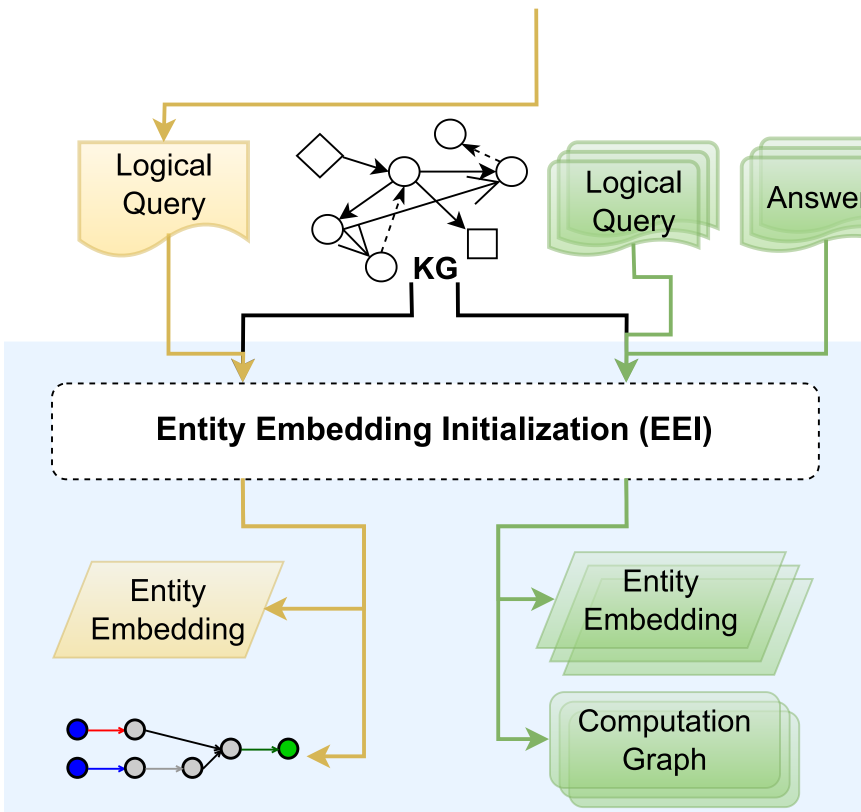


Figure 4.2: For training, LinE takes as input the KG and pairs of logical queries and answers. Given the computation graph for each query, the entity embedding initialization (EEI) component generates theinitial embeddings for KG entities in a probabilistic representation space (blue area).

4.2.1 LinE Representation Space

In the LinE space, both the entities and queries S are represented as *LinE embeddings*, a low-dimensional representation, whereby each dimension in its general form is expressed as a sequence of k values. LinE embeddings are no longer restricted by any distributional shape like $\mathbf{B}(\cdot)$. Formally, an h-dimensional LinE embedding for the set S is defined as $L^S = [(p^S_{j,1}, p^S_{j,2}, \cdots, p^S_{j,k})]^h_{j=1} \in \mathbb{R}^{k \times h}$ across k positions in the range $[0,1]$ per dimension $1 \leq j \leq h$. For instance, suppose we want to learn 200-dimensional LinE embeddings with a sequence of three values per dimension. A logic space transformation function predicts values at three positions (e.g., $\{0.25, 0.50, 0.75\}$) per dimension in the LinE space, and forms the 200-dimensional LinE embedding $[(p_{j,0.25}, p_{j,0.50}, p_{j,0.75})]^{200}_{j=1}$. Figure 4.3 illustrates the comparison between a Beta embedding and the LinE embedding. Note that LinE embedding preserves the closure property because the following properties hold. First, each entity itself in the LinE space is an h-dimensional LinE embedding because an entity is equivalently a set with a single element. Second, a query q, after executing logical operators ($\wedge$, $\vee$, $\exists$ and $\neg$) and relational projections ($r \in \mathcal{R}$) as appropriate in the LinE space, returns a set of answer entities, which is also a LinE embedding in h-dimensional LinE space. In this work, we learn neural transform functions in two steps: (i) augmenting the entity features based on the n-dimensional Beta embedding in the source logic space; and (ii) applying an entity-wise learnable function to derive LinE embeddings.

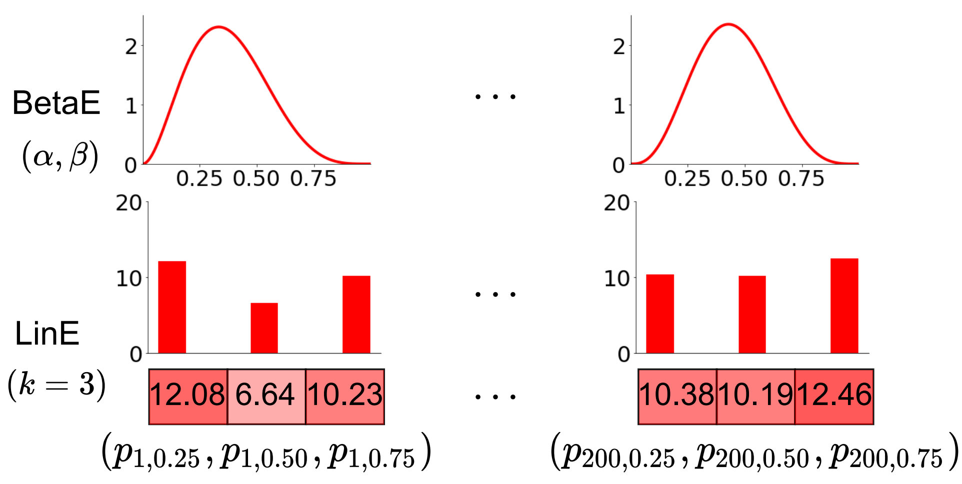


Figure 4.3: Logic Space Transformation: the top and bottom row depict the 200-dimensional Beta embedding and 200-dimensional LinE embedding at positions $(0.25, 0.50, 0.75)$.

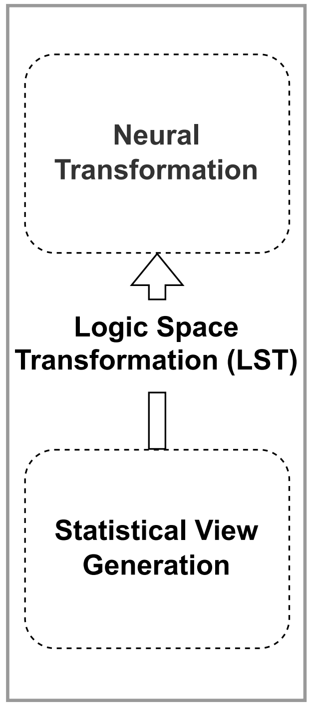


Figure 4.4: The logic space transformation (LST) to better capture multi-relational and logical regularities. The statistical view of the initial entity embeddings is thus generated to learn the entity embeddings in the LinE representation space (LinE space).

4.2.2 Statistical View Generation

To obtain effective features, we explore multiple features, which statistically describe the shapes of Beta distributions. The neural transformation function optimizes the network parameters so that two entities with the same shape features are likely to share the same initial LinE embeddings. Specifically, we consider the following shape features.

- α **and** β: α and β are explicit parameters of a Beta distribution. Given an h-dimensional Beta embedding B^{S_i} for entity set S_i, we obtain the h pairs of (α,β) parameters as input features for the neural transformation function as follows.

$$B^{S_i} = [(\alpha_j^{S_i}, \beta_j^{S_i})]_{j=1}^{h} \in \mathbb{R}^{2\times h} \tag{4.1}$$

- **Mean and Variance**: Mean ($\mu(\mathbf{X})$) measures the central tendency of a Beta distribution (e.g., left-skewed or right-skewed). Variance ($var(\mathbf{X})$) measures the statistical dispersion of a distribution.

Mean. The expected value μ of a Beta distributed random variable $\mathbf{X}$ is formally defined with (α,β) as follows.

$$\mu(\mathbf{X}) = E[\mathbf{X}] = \frac{\alpha}{\alpha+\beta} \tag{4.2}$$

Variance. The expected variance var of a Beta distributed random variable X is formally defined with (α,β) as follows.

$$var(\mathbf{X}) = E[(\mathbf{X}-\mu(\mathbf{X}))^2] = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} \tag{4.3}$$

Given an h-dimensional Beta embeddings B^{S_i} for set S_i, we obtain h pairs of mean and variance for h Beta distributions as follows.

$$P^{S_i} = [(\mu(\mathbf{X}_j)^{S_i}, var(\mathbf{X}_j)^{S_i})]_{j=1}^{h} \in \mathbb{R}^{2\times h}. \tag{4.4}$$

- **Mode and Concentration**: Mode is another indicator to measure central tendency by locating the highest points of the distribution (e.g., bimodal, unimodal, and no mode). Different modes differ in their mathematical formulations. Formally, the mode ($mode(\mathbf{X})$) of a Beta distributed random variable $\mathbf{X}$ with (α,β) is defined in accordance with their central tendency as follows.

$$mode(\mathbf{X}) = \begin{cases} \frac{\alpha-1}{\alpha+\beta-2} & \text{if } \alpha,\beta > 1 \\ \mathbb{R} \in (0,1) & \text{if } \alpha,\beta = 1 \\ 0,1 & \text{if } \alpha,\beta < 1 \\ 0 & \text{if } \alpha \leq 1, \beta > 1 \\ 1 & \text{if } \alpha > 1, \beta \leq 1 \end{cases} \tag{4.5}$$

To express modes with valid numeric ranges, we utilize three-dimensional binary codes (l,m,r) to correlate different mode positions along a dimension. Specifically, l equals to 1 when $mode(X) = 0$, and 0 otherwise.r equals to 1 when $mode(X) = 1$, and 0 otherwise. m equals to 1 when $mode(X) \neq 0$ or 1. When $\alpha,\beta < 1$, the binary codes become $(1,0,1)$; whereas the binary codes becomes $(0,1,0)$ when $\alpha,\beta > 1$. Formally, mode binary codes for an h-dimensional Beta embedding for entity set S_i is defined as follows.

$$M^{S_i} = [(l_j^{S_i}, m_j^{S_i}, r_j^{S_i})]_{j=1}^{h} \in \mathbb{R}^{3\times h} \tag{4.6}$$

4.2.3 Neural Transformation

To learn an entity-wise neural transformation function, we adopt a Multi-Layer Perceptron (MLP) with the shape features as input. Formally, given n entity sets **S**=$\{S_1,\cdots,S_n\}$, the shape features extracted from n Beta embeddings **F**=$\{F_1, \cdots, F_n\}$ is fed to an entity-wise MLP, which generates n LinE embeddings **L**=$\{L_1, \cdots, L_n\}$. A neural network that converts a set of Beta embeddings to a set of LinE embeddings is trained as follows.

$$\mathbf{L} = \mathbf{MLP}(\mathbf{F}) \tag{4.7}$$
$$F^{S_i} = [B^{S_i}] — [P^{S_i}], \forall S_i \in \mathbf{S}$$
$$L^{S_i} = [(p^{S_i}_{j,1}, \cdots, p^{S_i}_{j,k})]^h_{j=1} \in \mathbb{R}^{k \times h}, \forall S_i \in \mathbf{S}$$

where $F_i \in F$ is the final shape feature concatenated from any of B^{S_i} and P^{S_i} for entity set S_i.

4.3 Logical Query Inference

Logical query inference (LQI) grounds in knowledge triples and logical regularities in the LinE space to support compositional logical query inference. Given the computation graph for a query q, LQI derives the final LinE embedding for q by executing logical operators ($\wedge$, $\vee$, $\exists$ and $\neg$) and relation-specific projections ($r \in \mathcal{R}$) in the computation graph. Figure 4.5 illustrates the LQI component in details. We describe each logical operator and relation-specific projection in the LinE space, including relational projection X_r, intersection L_{Inter}, negation L_{Neg}, and union L_{Union}.

4.3.1 Relation-specific Projection

Relations in multi-relational KGs have diverse properties: symmetric (e.g., *adjoins*), anti-symmetric (e.g., *is_a_member_of*), and transitive relations (e.g., *contains*). Preserving each type of relational behavior therefore requires a range of neural functions to capture relational properties in latent representation space.

To capture diverse relational properties in KGs, each atomic relational projections of any forms in Eq. (3.2) in a compositional logical query is realized by a relation-specific projection X_r for each $r \in \mathcal{R}$. For illustration, given an atomic relation query $r(v_a, V)$, a relation-specific projection learns a projection function X_r that takes v_a as input and projects v_a closer to the representations of a set of tail entities, corresponding to the existentially quantified bound variables V, in the LinE space. A projection function X_r is formulated as a neural network for the relation type $r \in \mathcal{R}$ by implementing a Multi-Layer Perceptron $\mathbf{MLP}_r$ with ReLU as activation function as follows.

$$\hat{L}^{S_j}(S_i, r) = \mathbf{MLP}_r(L^{S_i}) \tag{4.8}$$

where L^{S_i} and $\hat{L}^{S_j}(S_i, r)$ denote the LinE embedding of an entity set $S_i = \{v_i\}$ and the estimated LinE embedding for the entity set $S_j = \{v_j\}$ after projection via relation r from v_i, with respect to the knowledge triples $r(S_i,S_j) \in \mathcal{K}$ in the LinE space.

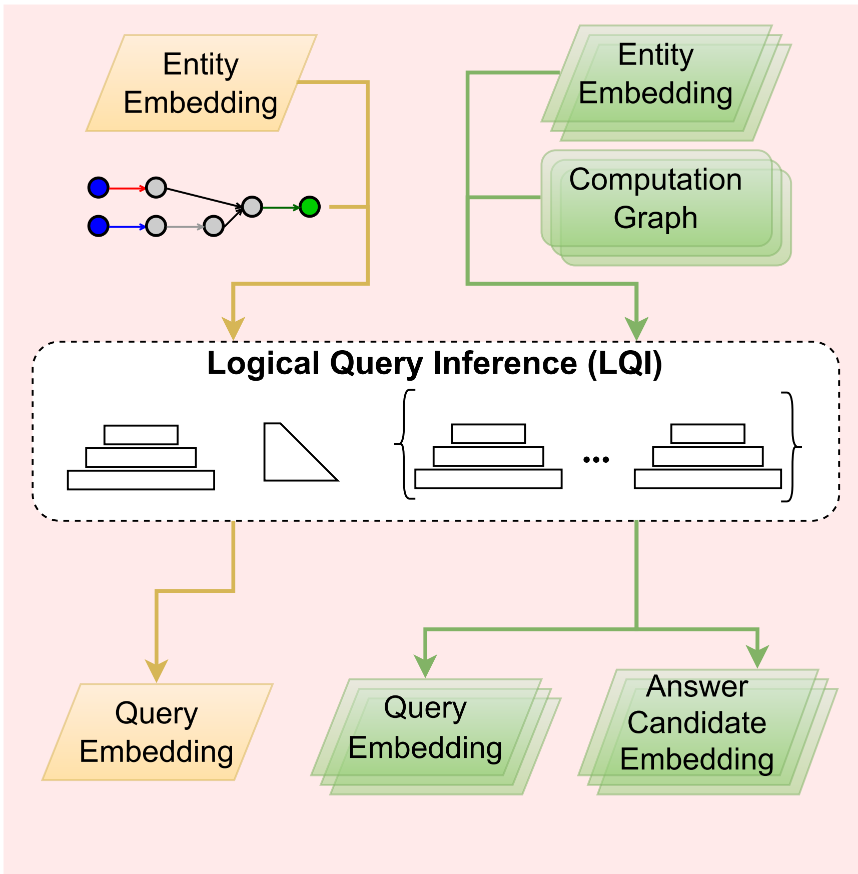


Figure 4.5: The logical query inference (LQI) component learns a set of relation-specific neural functions and refines the embeddings for KG entities and queries based on multi-relational and logical regularities in the LinE space(pink area).

4.3.2 Relational Regulations

To sensitively capture mixed structural regularities in a KG, we use hierarchy estimates in Section 3.1 to disjointly distinguish hierarchical ($\mathcal{R}_{\mathcal{T}}$) and non-hierarchical relations ($\mathcal{R}_{\overline{\mathcal{T}}}$). Accordingly, we propose the following relational regulations $\mathcal{D}_{\mathcal{T}}$ and $\mathcal{D}_{\overline{\mathcal{T}}}$ to preserve hierarchical relations ($\mathcal{R}_{\mathcal{T}}$) and non-hierarchical relations ($\mathcal{R}_{\overline{\mathcal{T}}}$), respectively, where $\mathcal{R}_{\mathcal{T}} \bigcap \mathcal{R}_{\overline{\mathcal{T}}} = \varnothing$ and $\mathcal{R}_{\mathcal{T}} \bigcup \mathcal{R}_{\overline{\mathcal{T}}} = \mathcal{R}$.

To preserve the hierarchical properties in the LinE space, the hierarchical violation against knowledge triples $\mathcal{K}$ is approximated by minimizing the following order violation.

$$\mathcal{D}_{\mathcal{T}} = \sum_{r(S_i,S_j)\in\mathcal{K}_{\mathcal{T}}} \max\{0, L^{S_i} - \hat{L}^{S_j}(S_i, r)\}, \forall r \in \mathcal{R}_{\mathcal{T}} \tag{4.9}$$

where $\mathcal{K}_{\mathcal{T}} = r(S_i, S_j) \subseteq \mathcal{K}$ is the set of triple bounded by hierarchical relations $r \in \mathcal{R}_{\mathcal{T}}$. In essence, Eq. (4.9) encourages LinE embeddings for entity set $S_i \subseteq \mathcal{V}$ associated with entity set $S_j \subseteq \mathcal{V}$ via hierarchical relation r to have smaller $L_d^{S_i}$ than $L_d^{S_j}$ for every dimension $d \in [1, h]$ in the LinE space, where $L_d^S \in \mathbb{R}$. A triple $r(S_i,S_j)$ has zero order violation if $L_d^{S_i} \leq L_d^{S_j}$ in the LinE space.

To preserve the non-hierarchical properties in LinE space, the LinE embeddings associated with non-hierarchical properties are alternatively regulated by the Mean Square Error (MSE).

$$\mathcal{D}_{\overline{\mathcal{T}}} = \sum_{r(S_i,S_j)\in\mathcal{K}_{\overline{\mathcal{T}}}} (L^{S_j} - \hat{L}^{S_j}(S_i, r))^2, \forall r \in \mathcal{R}_{\overline{\mathcal{T}}} \tag{4.10}$$

where $\mathcal{K}_{\overline{\mathcal{T}}} = r(S_i, S_j) \subseteq \mathcal{K}$ is the set of triples bounded by non-hierarchical relations $r \in \mathcal{R}_{\overline{\mathcal{T}}}$. Eq. (4.10) essentially preserves the L2-distance between LinE embeddings and the projected LinE embeddings of the entity set S_j, i.e., forcing L^{S_j} to be as close to $\hat{L}^{S_j}(S_i, r)$ obtained by the neural function $\mathbf{MLP}_r(L_{S_i})$ as possible.

4.3.3 Intersection Operator

The intersection of multiple quantities is essentially the minimum of all. Following this intuition, we propose to use $\mathbf{min}$ function to simulate the intersection operation as shown in Figure 4.6(a). Given n h-dimensional LinE embeddings $\{L^{S_1}, L^{S_2}, ..., L^{S_n}\}$, we calculate the intersection L_{Inter} by simply applying the minimum function, to amplify the agreements amongst n LinE embeddings across $d \in [1, h]$. The intersection operator is formally defined as follows.

$$L_{Inter} = [\min\{p_1^{S_1}, p_1^{S_2}, ..., p_1^{S_n}\}, ..., \min\{p_k^{S_1}, p_k^{S_2}, ..., p_k^{S_n}\}]^h \tag{4.11}$$

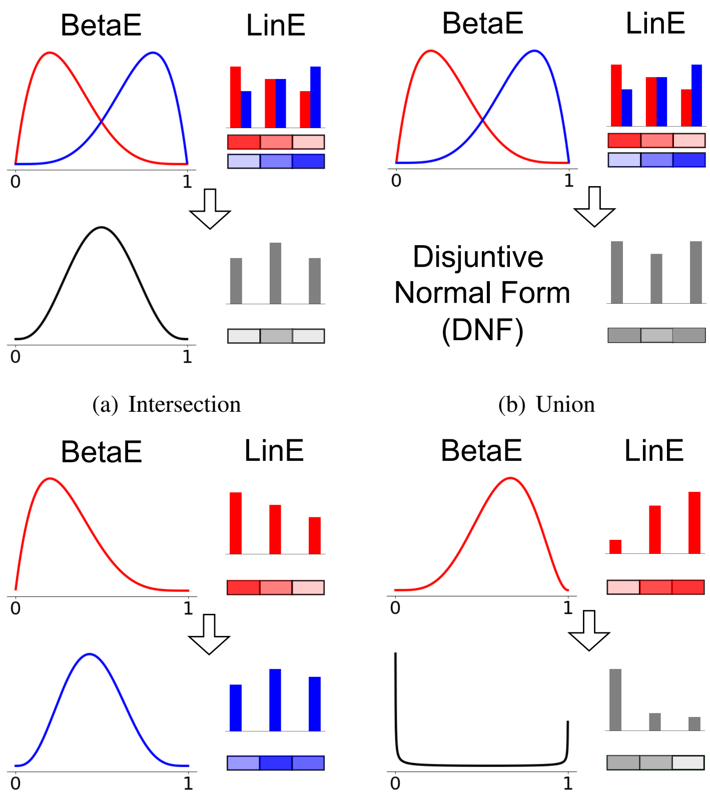
where $p_j^{S_i}$ represents the j-th position in the LinE embedding L^{S_i} for i-th entity set and k is the number of sampled positions.

4.3.4 Union Operator

Prior work dealt with union operations by drastically restructuring the computation graphs into DNF [37]. On the contrary, we directly formulate the union operator with a maximum function for given LinE embeddings as shown in Figure 4.6(b). Formally, the union is defined as follows.

$$L_{Union} = [\max\{p_1^{S_1}, p_1^{S_2}, ..., p_1^{S_n}\}, ..., \max\{p_k^{S_1}, p_k^{S_2}, ..., p_k^{S_n}\}]^h \tag{4.12}$$

where $p_j^{S_i}$ represents the j-th value in the LinE embedding L^{S_i} for the entity set S_i, and k is the number of sampled positions.



(a) Intersection (b) Union

(c) Relational projection (d) Negation

Figure 4.6: Logical Operations for BetaE and LinE Embeddings: (a) intersection, (b) union, (c) relational projection and (d) negation operations. Note that BetaE cannot directly handle union operations.

4.3.5 Negation Operator

Given the Beta embedding (α,β) for a set S, we follow a prior work [37] to compute $(\frac{1}{\alpha}, \frac{1}{\beta})$ as the negation of (α, β) as illustrated in Figure 4.6(d). Formally, we define the negation operation for LinE embeddings for the set S as follows.

$$L_{Neg} = [\frac{1}{p_1^S}, \frac{1}{p_2^S}, ..., \frac{1}{p_k^S}]^h \tag{4.13}$$

where p_j^S represents the j-th position in the LinE embedding L^S and k is the number of sampled positions.

4.3.6 Logical Regulations

Intuitively, the closer the resulting LinE embeddings after intersection operators to the initial transformed LinE embeddings for ground-truth answer entities, the better quality of refined LinE embeddings. To preserve the logical laws in the LinE space, the violations against the intersection operators ($\wedge$) is formulated as an MSE estimator as follows.

$$\mathcal{D}_{FOL} = \sum_{\hat{L}_{Inter} := \cap_{i=1}^{n} L^{S_i}} (\hat{L}_{Inter} - L_{Inter})^2 \tag{4.14}$$

where $\hat{L}_{Inter}$ is the LinE embedding after performing intersection for n LinE embeddings L^{S_i} with $i \in [1, n]$, and L_{Inter} is the LinE embedding of the true answers.

4.4 Logical Query Reasoning

Joint Learning Objective. We consider both *reasoning loss* and *query inference loss* to jointly optimize LinE embeddings and parameterized neural functions as follows.

$$\mathcal{L} = \mathcal{L}_{\mathcal{A}} + \lambda \mathcal{L}_{\mathcal{Q}} \tag{4.15}$$

where $\mathcal{L}_{\mathcal{A}}$ is the reasoning loss, $\mathcal{L}_{\mathcal{Q}}$ is the query inference loss, and λ is a hyper-parameter controlling their respective importance. We describe each of them in the following.

Reasoning Loss. The reasoning loss estimates the distance between a logical query q and its true answer in the LinE space. Let $S_y = \{v_y | v_y \in \mathcal{V}\}$ be the true answers to the query q. For each true query and answer pair (q, v_y), we randomly select K false answers $v_{y'}$, $S_{y'} = \{v_{y'} | v_{y'} \in \mathcal{V}\}$. To optimize, we minimize the skip-gram loss $\mathcal{L}_{\mathcal{A}}$ on training pairs $\mathcal{T}_{\mathcal{Q}}$ and $\mathcal{T}_{\mathcal{A}}$ as follows.

$$\mathcal{L}_{\mathcal{A}} = -\log(\gamma - \mathcal{D}_{QA}(L^{v_y}, L^q)) - \sum_{v_{y'} \in S_{y'}} \frac{1}{K} \log(\mathcal{D}_{QA}(L^{v_{y'}}, L^q) - \gamma) \tag{4.16}$$

where L^q is the query embedding, L^{v_y} is the true answer embedding in the LinE space, $\mathcal{D}_{QA}$ is the MSE estimator for a QA pair, and γ is a margin. Eq. (4.16) encourages the query L^q to be positioned closer to the true answers within γ L2-distance while far away from the false answers at least γ L2-distance in the LinE space.

Query Inference Loss. The logical inference loss estimates the *relational violations* and the *deviations* from *logical regularities*, in particular, intersections executed in a computation graph for query q. The query inference loss is formally defined as follows.

$$\mathcal{L}_{\mathcal{Q}} = \sum_{r \in \mathcal{R}_{\overline{\mathcal{T}}}} \mathcal{D}_{\overline{\mathcal{T}}} + \sum_{r \in \mathcal{R}_{\mathcal{T}}} \mathcal{D}_{\mathcal{T}} + \sum_{\cap} \mathcal{D}_{FOL} \tag{4.17}$$

where $\mathcal{D}_{\overline{\mathcal{T}}}$ and $\mathcal{D}_{\mathcal{T}}$ are the relational violations measured against hierarchical and non-hierarchical relations, respectively. $\mathcal{D}_{FOL}$ measures the logical violations against intersection operations. Note that due to limitations in training QA pairs, we only regulate LinE embeddings with respect to queries using intersection operators throughout the corresponding computation graphs.

An Illustrative Example. Given a query q, LinE optimizes LinE embeddings and the set of parameterized relational/logical functions by Eq.4.15. The specific learnable items for the example query in Figure 3.2(b) are as follows: (i) the set of participant countries of 2008 Beijing Olympics (V_1) via "*participant*" relational projection (Eq.4.8); (ii) the set of countries won medals in 1952 Helsinki Olympics V via "*awardedTo*" relational projection (Eq.4.8); (iii) the complement of set V (V_2), obtained by performing negation (Eq.4.13); (iv) the set of countries (V_3) by performing intersection between V_1 and V_2 (Eq.4.11); and lastly (v) the set of places in set V_3 ($V_?$) via "*contains*" relational projection (Eq.4.8). Entities in $V_?$ are returned as the final answers to query q, which is the capital city in Fiji ("Suva"). The overall reasoning formulation to derive the learnable parameters for the example query in Figure 3.2(b) is illustrated in Figure 4.7.

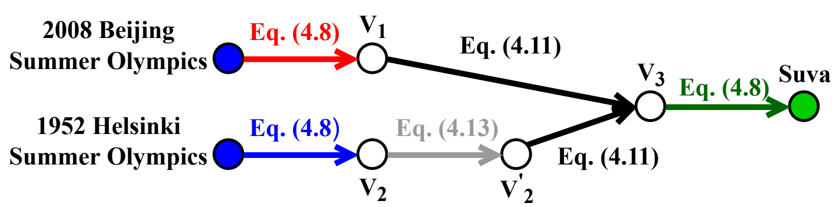


Figure 4.7: An Illustrative Example.

4.5 WN18RR-QA Benchmark

To study the reasoning sensitivity for hierarchical logical queries, we consider to generate QA pairs for WN18RR, which richly contains hierarchical relations. In Table 3.1, seven out of 11 relations on WN18RR are viewed as hierarchical relations due to their high anti-symmetry ($Khs_{\mathcal{G}_r}$) and negative transitive scores ($\xi_{\mathcal{G}_r}$).

4.5.1 Grounding Query Structures

To generate queries for training, we follow prior work [36, 37] to generate ten types of query structures, including 1p, 2p, 3p, 2i, 3i, 2in, 3in, inp, pin and pni. For evaluation, we consider all 14 query structures that are both seen and unseen during the training process. Given a KG and a query structure s seen as a directed acyclic graph (DAG), we adopt pre-order traversal to assign entities and relation types for each node and edge in the DAG to construct the query q

Algorithm 1: Logical Query and Answer Pair Generation

Input: $\mathcal{G} = (\mathcal{V}, \mathcal{R})$, $\mathcal{K}$, 14 query types S;

Output: queries $\mathcal{T}_{\mathcal{Q}}$, answers $\mathcal{T}_{\mathcal{A}}$;

Function `GoundQueries(`G_s, $\mathcal{G}$, $\mathcal{K}$`)`**:**
- $v \leftarrow$ uniformly sample (w/o replacement) an entity $v \in \mathcal{V}$
- $\mathcal{T}_{\mathcal{Q},s} \leftarrow$ assign entity v for the target root node $\boldsymbol{r_s} \in V_s$
- **foreach** *node* $\boldsymbol{v_s} \in V_s \setminus \{\boldsymbol{r_s}\}$ *in pre-order traversal ordering* **do**
  - $v \leftarrow$ the entity assigned for the parent of node $\boldsymbol{v_s}$
  - $\mathcal{K}_v \leftarrow$ set of triples with the object as v via any relation
  - $r(v', v) \leftarrow$ uniformly sample a triple from $\mathcal{K}_v$
  - $\mathcal{T}_{\mathcal{Q},s} \leftarrow$ assign entity v' for the node $\boldsymbol{v_s}$
  - $\mathcal{T}_{\mathcal{Q},s} \leftarrow$ assign relation r for the edge from $\boldsymbol{v_s}$ to its parent
- **return** $\mathcal{T}_{\mathcal{Q},s}$

End Function

foreach *query structure* $s \in S$ **do**
- `/* step 1: generate queries */`
- $G_s = (V_s, E_s) \leftarrow$ *induce a DAG according to the query structure* s
- $\mathcal{T}_{\mathcal{Q}} \leftarrow \mathcal{T}_{\mathcal{Q}} \cup$ *GoundQueries(*G_s, $\mathcal{G}$, $\mathcal{K}$*)*
- `/* step 2: generate answers */`
- $\mathcal{T}_{\mathcal{A},s} \leftarrow$ *collect target entities as the final answers for* $\mathcal{T}_{\mathcal{Q},s}$
- $\mathcal{T}_{\mathcal{A}} \leftarrow \mathcal{T}_{\mathcal{A}} \cup \mathcal{T}_{\mathcal{A},s}$

return $\mathcal{T}_{\mathcal{Q}}$, $\mathcal{T}_{\mathcal{A}}$

with type s. That is, starting from the target root node to the anchor leaf nodes for the given DAG, we uniformly sample an entity $v \in \mathcal{V}$ in the KG as the target node. For each child node linked to the target node in the DAG, we uniformly sample a triplet $r(v', v)$ with object entity v via relation r in the KG. We then assign the relation r to the edge and entity v' to the child node. Iteratively, we continue the next assignment of edges and nodes via pre-order traversal until each edge and node in the DAG are grounded with specific relation and entity in the KG. As a result, the leaf nodes in the DAG are viewed as the anchor nodes and the target root node is collected as the ground-truth answer to the query q. We follow this procedure to generate the set of QA pairs, $\mathcal{T}_{\mathcal{Q}} = \{q_1, q_2, \cdots, q_n\}$ and $\mathcal{T}_{\mathcal{A}} = \{q_1[V_?], q_2[V_?], \cdots, q_n[V_?]\}$, for training, validation and testing on WN18RR. The overall procedure of logical query and answer pair generation is summarized in Algorithm 1. Examples of generated FOL queries across 14 types of query structures on WN18RR benchmark are illustrated in Table 4.1.

Table 4.1: Examples of constructed logical queries on WN18RR benchmark.

Relation-heavy Queries

Type	**FOL Queries** ($q = V_?$)
1p	$hasPart(V_?, England)$
2p	$\exists V:\ hasPart(England, V) \wedge hasPart(V, V_?)$
3p	$\exists V:\ hasPart(England, V) \wedge instanceHypernym(V, V) \wedge hypernym(V_?, V)$

Logic-heavy Queries

Type	**FOL Queries** ($q = V_?$)
2i	$hasPart(England, V_?) \wedge derivationallyRelatedForm(V_?, Britannic)$
3i	$hasPart(America, V_?) \wedge hasPart(West, V_?) \wedge\ instanceHypernym(V_?, Continent)$
ip	$\exists V:\ derivationallyRelatedForm(V, Britannic) \wedge hasPart(V, England)$ $\wedge\ memberOfDomainRegion(V, V_?)$
pi	$\exists V:\ hasPart(V, England) \wedge memberOfDomainRegion(V, V_?)$ $\wedge\ memberOfDomainRegion(United\ Kingdom, V_?)$
2u	$instanceHypernym(V_?, river) \vee hasPart(Germany, V_?)$
up	$\exists V:\ memberMeronym(V, Roman) \vee hasPart(Asia, V)$ $\wedge\ synsetDomainTopicOf(V, V_?)$

Negation-related Queries

Type	**FOL Queries** ($q = V_?$)
2in	$hasPart(England, V_?) \wedge \neg derivationallyRelatedForm(Londoner, V_?)$
3in	$instanceHypernym(V_?, Urban\ Center) \wedge instanceHypernym(V_?, Port)$ $\wedge\ \neg hasPart(England, V_?)$
inp	$\exists V:\ hasPart(V, NorthernIreland) \wedge \neg hasPart(V, England)$ $\wedge\ memberOfDomainRegion(V, V_?)$
pin	$\exists V:\ hasPart(V, England) \wedge memberOfDomainRegion(V, V_?)$ $\wedge\ \neg memberOfDomainRegion(United\ Kingdom, V_?)$
pni	$\exists V:\ memberMeronym(Great\ Britain, V) \wedge \neg hypernym(V, V_?)$ $\wedge\ memberMeronym(England, V_?)$

4.5.2 Evaluation Protocol

For fair performance comparison, we follow the split setup for the generated queries. Namely, the distribution of the number of queries for each query structure remains approximately identical to FB15k-237 and NELL995 in BetaE [37]. Specifically, we generate all "1p" queries (10,350), which is exactly the number of triplets $|\mathcal{K}|$ in the original WN18RR (Table 5.2). The entire set of "1p" queries are used for training. For each query type of (2p/3p/2i/3i), we generate the same amount of queries as "1p" query (i.e., 103,509) as training set. For each query type of (2in/3in/inp/pin/pni), we generate at one tenth the "1p" query (i.e., 10,350) as training set. For validation and testing queries, we generate approximately one fifth the "1p" query in validation and testing query sets (~5K) for each query type (i.e., 1K). For example, given 5,202 and 5,356 of "1p" query in validation and testing sets from WN18RR, respectively, we generate 1,000 queries for other 13 query types for validation and testing. Table 5.3 reports the detail distributions of query types for FB15k-237, NELL995, and WN18RR.

Chapter 5

Experiment

This chapter describes the experimental evaluation of the reasoning performance of LinE. Section 5.1 describes the KGs and benchmarks dataset. Section 5.2 describes the overall experiment setups. Section 5.3 to Section 5.6 illustrate respective research questions. Particularly, we study the following four research questions:

- RQ1:What is the reasoning capability across complex logical queries?
- RQ2:What is the reasoning capability for logical queries with and without hierarchical relations?
- RQ3:What is the impact of logic space transformation?
- RQ4:What is the impact of relation projections and logical operators?

5.1 Datasets

We consider three KG benchmark datasets for logical query reasoning.

- FB15k-237 [8]: a collection of relations between entities constructed from subset of FB15k in Freebase. All simple inversible relations are removed so as to focus on more complex relations.
- NELL995 [9]: a collection of relations between entities constructed from the 995-th iteration of the Never-Ending Language Learning (NELL) system.
- WN18RR [17]: a hierarchical collection of relations between words created from subset of WordNet [28].

Data statistics on KGs and logical query are summarized in Table 5.1 and Table 5.2 (see Table 5.3 for more details), respectively.

Table 5.1: The statistics of KG datasets.

KG	$\|\mathcal{V}\|$	$\|\mathcal{R}\|$	$\|\mathcal{K}\|$
FB15k-237	14,505	273	149,689
NELL995	63,361	200	107,982
WN18RR	40,943	11	93,003

Table 5.2: The statistics of logical queries datasets generated from KG benchmarks.

KG	1p	2p	3p	2i	3i	ip	pi	2u	up	2in	3in	inp	pin	pni
FB15k-237	192,602	159,689	159,689	159,689	159,689	10,000	10,000	10,000	10,000	24,968	24,968	24,968	24,968	24,968
NELL995	141,943	115,982	115,982	115,982	115,982	8,000	8,000	8,000	8,000	18,798	18,798	18,798	18,798	18,798
WN18RR	114,067	105,509	105,509	105,509	105,509	2,000	2,000	2,000	2,000	12,350	12,350	12,350	12,350	12,350

Table 5.3: Number of training, validation and testing queries generated for different query structures across three benchmarks.

KG	1p	2p	3p	2i	3i	ip	pi	2u	up	2in	3in	inp	pin	pni
Training														
FB15k-237	149,689	149,689	149,689	149,689	149,689	-	-	-	-	14,968	14,968	14,968	14,968	14,968
NELL995	107,982	107,982	107,982	107,982	107,982	-	-	-	-	10,798	10,798	10,798	10,798	10,798
WN18RR	103,509	103,509	103,509	103,509	103,509	-	-	-	-	10,350	10,350	10,350	10,350	10,350
Validation														
FB15k-237	20,101	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000
NELL995	16,927	4,000	4,000	4,000	4,000	4,000	4,000	4,000	4,000	4,000	4,000	4,000	4,000	4,000
WN18RR	5,202	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
Testing														
FB15k-237	22,812	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000
NELL995	17,034	4,000	4,000	4,000	4,000	4,000	4,000	4,000	4,000	4,000	4,000	4,000	4,000	4,000
WN18RR	5,356	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000

5.2 Experimental Settings

Baselines. We consider the following dominant baselines for answering complex logical queries on KGs. The first category is generic logical query reasoners (GQE [19], Q2B [36] and BetaE [37]), which formulate embeddings in Euclidean space. The second category is hierarchical logical query reasoners (HypE [15]), which formulate embeddings in Hyperbolic space. Each single experiment is run on a single NVIDIA Quadro RTX 8000 GPU, and we run each method for 100k iterations.

- Graph Query Embedding (**GQE**) [19]: GQE can answer FOL queries without negation operator ($\neg$).
- Query2Box (**Q2B**) [36]: Q2B can answer FOL queries without negation operator ($\neg$).
- Beta Embedding (**BetaE**) [37]: BetaE is the first model that tackles FOL queries with

the complete set of logical operators.

- Hyperboloid Embeddings (**HypE**) [15]: HypE can answer FOL queries without negation operator ($\neg$).

Metrics. To evaluate the performance of examined methods, we measure the answer quality by the ranking of the true answers. We report two standard evaluation metrics: MRR and HITS@N, which is the proportion of correct answers in the top-N candidates.

5.3 Query Reasoning Complexity (RQ1)

Table 5.4: Performance comparison in MRR (%) on benchmarks. avg$_p$, avg$_l$, and avg$_n$ denote the average MRR on relation-heavy, logic-heavy, and negation-related queries, respectively. Best (second best) of each column are in bold (underlined). The last row shows relative improvement (%) of LinE$_{\alpha,\beta}$ compared to the best baseline.

Model	**FB15k-237**			**NELL995**			**WN18RR**		
	avg$_p$	avg$_l$	avg$_n$	avg$_p$	avg$_l$	avg$_n$	avg$_p$	avg$_l$	avg$_n$
Generalization Reasoning Task									
GQE	9.85	10.43	-	12.26	12.95	-	8.41	11.62	-
Q2B	13.04	13.89	-	14.59	16.17	-	9.80	16.25	-
BetaE	<u>17.50</u>	<u>16.41</u>	<u>4.78</u>	<u>19.79</u>	<u>16.60</u>	<u>5.11</u>	<u>19.28</u>	**32.83**	<u>17.87</u>
HypE	17.34	8.88	-	18.25	14.50	-	9.62	19.83	-
LinE$_{\alpha,\beta}$	**18.50**	**17.65**	**5.71**	**22.27**	**17.52**	**5.46**	**21.39**	<u>28.21</u>	**18.72**
Rel. Gain (%)	5.71	7.56	19.46	12.53	5.54	6.85	10.94	-14.07	4.76
Logical Entailment Task									
GQE	12.77	15.40	-	21.38	21.95	-	16.99	19.06	-
Q2B	20.30	22.60	-	26.23	30.53	-	18.81	29.49	-
BetaE	<u>23.09</u>	<u>23.72</u>	<u>6.44</u>	<u>31.60</u>	<u>27.78</u>	<u>7.54</u>	<u>37.63</u>	<u>52.40</u>	<u>28.98</u>
LinE$_{\alpha,\beta}$	**26.29**	**26.35**	**7.07**	**39.44**	**33.47**	**9.39**	**43.23**	**54.36**	**32.71**
Rel. Gain (%)	13.86	11.09	9.78	24.81	20.48	24.54	14.88	3.74	12.87

5.3.1 Setup

To study the reasoning complexity of logical queries, we have two different tasks to compare our model with baselines.

Generalization Reasoning Task. We follow the formulation of the *generalization reasoning task* on 14 queries with at least one link prediction [36] to evaluate the performance in generalizing to plausible answers.

Table 5.5: Performance comparison in HITS@3 (%) on benchmarks. avg_p, avg_l, and avg_n denote the average HITS@3 on relation-heavy, logic-heavy, and negation-related queries, respectively. Best (second best) of each column are in bold (underlined). The last row shows relative improvement (%) of $\text{LinE}_{\alpha,\beta}$ compared to the best baseline.

Model	**FB15k-237**			**NELL995**			**WN18RR**		
	avg_p	avg_l	avg_n	avg_p	avg_l	avg_n	avg_p	avg_l	avg_n
Generalization Reasoning Task									
GQE	12.58	12.24	-	18.06	15.79	-	12.70	16.48	-
Q2B	15.15	16.07	-	<u>21.79</u>	**19.04**	-	15.97	27.08	-
BetaE	<u>18.50</u>	<u>17.47</u>	<u>3.97</u>	21.61	17.84	<u>4.51</u>	<u>20.11</u>	**35.64**	<u>19.08</u>
HypE	18.46	8.82	-	20.53	15.65	-	14.46	24.43	-
$\text{LinE}_{\alpha,\beta}$	**19.69**	**19.04**	**4.40**	**24.32**	<u>18.87</u>	**4.72**	**22.17**	<u>30.12</u>	**20.37**
Rel. Gain (%)	6.43	8.99	10.83	11.61	-0.89	4.66	10.24	-15.49	6.76
Logical Entailment Task									
GQE	16.85	19.42	-	33.15	28.06	-	27.31	32.92	-
Q2B	<u>27.59</u>	<u>28.41</u>	-	<u>38.88</u>	**37.61**	-	31.60	46.05	-
BetaE	24.97	25.88	<u>5.43</u>	34.63	30.25	<u>6.15</u>	<u>40.15</u>	<u>56.84</u>	<u>32.11</u>
$\text{LinE}_{\alpha,\beta}$	**28.94**	**28.64**	**6.08**	**43.04**	<u>36.45</u>	**7.89**	**45.45**	**58.20**	**35.27**
Rel. Gain (%)	4.89	0.81	11.97	10.70	-3.08	28.29	13.20	2.39	9.84

Logical Entailment Task. Likewise, we follow the formulation of *logical entailment task* on 14 queries without missing links to evaluate the performance in deductive reasoning. We evaluate the task on three benchmarks, including FB15k-237, NELL995 and WN18RR.

To evaluate the reasoning sensitivity to query diversity, we divide 14 query structures into three categories: (i) *relation-heavy*: queries that are purely tied to relation projections (1p/2p/3p), (ii) *logic-heavy*: queries that are heavily tied to logical operations (2i/3i/ip/pi/2u/up), and (iii) *negation-related*: queries that involve negation operator as shown in Figure 3.3 in Section 3.2.

5.3.2 Results

Generalization Reasoning Task. Tables 5.4 and 5.5 report the comparative results for each query group on three benchmarks. Our findings are three-fold.

First, we observe that BetaE and LinE consistently outperform other baselines (GQE, Q2B and HypE) across three benchmarks. In particular on FB15k-237, LinE achieves significant performance gain against GQE by 87.82% and 69.22% for avg_p and avg_l in MRR, respectively. LinE achieves relative gain against Q2B by 41.87% and 27.07% for avg_p and avg_l in MRR. LinE achieves 6.69% and 98.76% relative gain over HypE for avg_p and avg_l in MRR. As for

Table 5.6: Performance comparison in MRR (%) on FB15k-237. Best (second best) of each column are in bold (underlined).

Model	FB15k-237													
	1p	2p	3p	2i	3i	ip	pi	2u	up	2in	3in	inp	pin	pni
Generalization Reasoning Task														
GQE	21.56	4.54	3.45	14.36	22.57	7.29	9.94	4.56	3.89	-	-	-	-	-
Q2B	29.75	5.51	3.86	19.18	31.96	8.67	14.08	5.35	4.13	-	-	-	-	-
BetaE	34.99	8.79	8.72	20.92	33.36	9.54	16.58	10.04	8.01	4.68	5.96	**6.67**	3.35	3.24
HypE	26.97	**14.44**	**10.62**	8.83	10.13	6.30	6.42	10.39	**11.24**	-	-	-	-	-
LinE$_{\alpha,\beta}$	**36.65**	9.85	9.00	**23.91**	**35.15**	**10.07**	**17.32**	**10.90**	8.53	**5.24**	**7.03**	6.65	**6.16**	**3.47**
Logical Entailment Task														
GQE	27.22	6.08	5.02	20.98	34.69	10.72	15.67	5.80	4.55	-	-	-	-	-
Q2B	47.06	**20.75**	5.39	34.35	47.23	12.40	23.69	12.13	5.79	-	-	-	-	-
BetaE	44.21	12.96	12.10	31.18	45.68	14.46	24.85	14.35	11.78	7.23	7.95	**7.76**	4.30	4.98
LinE$_{\alpha,\beta}$	**51.10**	14.42	**13.34**	**34.82**	**50.02**	**16.67**	**26.86**	**17.34**	**12.36**	**8.52**	**8.85**	7.50	**4.54**	**5.95**

Table 5.7: Performance comparison in HITS@3 (%) on FB15k-237. Best (second best) of each column are in bold (underlined).

Model	FB15k-237													
	1p	2p	3p	2i	3i	ip	pi	2u	up	2in	3in	inp	pin	pni
Generalization Reasoning Task														
GQE	30.08	4.41	3.24	17.85	29.31	7.44	11.79	3.77	3.27	-	-	-	-	-
Q2B	36.42	5.51	3.52	23.74	39.24	8.63	16.69	4.62	3.52	-	-	-	-	-
BetaE	38.53	8.42	8.56	23.02	37.13	9.50	17.37	9.96	7.82	3.93	4.84	**6.24**	2.42	2.41
HypE	29.94	**14.72**	**10.73**	8.74	10.01	6.11	6.01	**10.81**	**11.22**	-	-	-	-	-
LinE$_{\alpha,\beta}$	**40.29**	9.76	9.02	**26.80**	**39.52**	**10.31**	**18.58**	**10.81**	8.25	**4.27**	**6.26**	6.16	**2.73**	**2.57**
Logical Entailment Task														
GQE	39.73	5.99	4.83	28.81	46.89	11.80	20.22	5.01	3.79	-	-	-	-	-
Q2B	**68.63**	9.05	5.08	**46.21**	**59.65**	13.10	**32.28**	13.93	5.27	-	-	-	-	-
BetaE	49.84	12.88	12.19	35.20	51.65	14.60	27.46	14.28	12.07	6.12	6.87	**7.27**	3.26	3.64
LinE$_{\alpha,\beta}$	58.13	**14.75**	**13.94**	39.16	55.88	**17.37**	29.35	**17.75**	**12.33**	**7.61**	**7.97**	6.61	**3.40**	**4.80**

NELL995, LinE achieves significant performance gain against GQE by 81.65% and 35.29% for avg$_p$ and avg$_l$ in MRR, respectively. LinE achieves relative gain against Q2B by 52.64% and 8.35% for avg$_p$ and avg$_l$ in MRR. LinE achieves 22.03% and 20.83% relative gain over HypE for avg$_p$ and avg$_l$ in MRR.

Second, LinE achieves considerable performance gain in most cases on three benchmarks compared to BetaE. Take FB15k-237 as an example, LinE shows superior reasoning ability

Table 5.8: Performance comparison in MRR (%) on NELL995. Best (second best) of each column are in bold (underlined).

Model	NELL995													
	1p	2p	3p	2i	3i	ip	pi	2u	up	2in	3in	inp	pin	pni
Generalization Reasoning Task														
GQE	22.21	8.55	6.01	18.52	25.33	9.96	12.40	5.36	6.16	-	-	-	-	-
Q2B	30.64	7.90	5.22	25.25	34.88	**10.07**	15.23	6.24	5.33	-	-	-	-	-
BetaE	42.62	8.49	8.28	25.49	35.25	8.66	**15.31**	8.50	6.38	4.57	6.43	8.78	3.10	2.67
HypE	26.84	**15.59**	**12.33**	16.84	23.07	7.97	11.95	**16.10**	**11.06**	-	-	-	-	-
LinE$_{\alpha,\beta}$	**47.16**	10.01	9.64	**27.54**	**36.91**	9.21	14.91	9.32	7.22	**4.96**	**6.71**	**9.26**	**3.38**	**3.01**
Logical Entailment Task														
GQE	34.37	15.91	13.86	30.03	37.27	20.85	22.25	10.18	11.15	-	-	-	-	-
Q2B	52.23	15.67	10.81	**48.00**	52.34	20.04	**34.01**	18.97	9.83	-	-	-	-	-
BetaE	60.08	17.90	16.82	41.48	48.79	18.52	27.28	14.88	15.76	7.79	8.82	10.66	4.40	6.05
LinE$_{\alpha,\beta}$	**68.93**	**25.39**	**24.00**	47.46	**55.87**	**23.71**	31.86	**21.93**	**19.98**	**10.75**	**9.88**	**11.92**	**5.66**	**8.75**

Table 5.9: Performance comparison in HITS@3 (%) on NELL995. Best (second best) of each column are in bold (underlined).

Model	NELL995													
	1p	2p	3p	2i	3i	ip	pi	2u	up	2in	3in	inp	pin	pni
Generalization Reasoning Task														
GQE	36.73	9.48	7.96	25.64	32.50	10.39	15.09	4.97	6.16	-	-	-	-	-
Q2B	50.98	8.50	5.91	**32.43**	**41.57**	**10.79**	**17.73**	6.62	5.07	-	-	-	-	-
BetaE	48.11	8.49	8.24	28.15	38.76	8.68	16.53	8.68	6.24	4.01	5.06	8.70	2.11	2.11
HypE	31.19	**17.08**	**13.32**	17.81	25.15	7.89	12.51	**18.85**	**11.70**	-	-	-	-	-
LinE$_{\alpha,\beta}$	**52.81**	10.23	9.92	30.06	40.86	9.53	16.17	9.30	7.31	**4.04**	**5.65**	**9.03**	**2.46**	**2.43**
Logical Entailment Task														
GQE	61.61	19.90	17.93	43.09	49.84	23.32	30.22	10.62	11.26	-	-	-	-	-
Q2B	**86.20**	18.18	12.25	**61.73**	**63.50**	21.69	**44.62**	**24.41**	9.68	-	-	-	-	-
BetaE	68.68	17.98	17.25	46.55	55.31	19.84	29.03	14.52	16.24	5.85	7.31	10.25	2.91	4.40
LinE$_{\alpha,\beta}$	77.65	**26.48**	**25.00**	53.36	62.46	**24.40**	35.10	23.09	**20.27**	**8.79**	**8.06**	**11.43**	**3.71**	**7.44**

on relation-heavy (5.71% gain in avg$_p$ MRR), logic-heavy (7.56% gain in avg$_l$ MRR) and negation-related queries (19.46% gain in avg$_n$ MRR). The experiment results on NELL995 also prove that LinE has superior reasoning ability on relation-heavy (12.53% gain in avg$_p$ MRR), logic-heavy (5.54% gain in avg$_l$ MRR) and negation-related queries (6.85% gain in avg$_n$ MRR). This suggests that LinE generally shows superior reasoning ability, particularly relation-heavy and negation-related queries despite the occasional miss on logic-heavy queries.

Third, LinE shows superior reasoning ability particularly for certain query types as shown in Tables 5.6, 5.7, 5.8 and 5.9. For example, LinE outperforms all baseline models on certain query types in both MRR and HITS@3 on FB15k-237, including "1p", "2i", "3i", "ip", "pi", "2u", "2in", "3in", "pin" and "pni". As for NELL995, LinE also achieves significant performance over these query types ("1p", "2in", "3in", "inp", "pin" and "pni") in terms of both MRR and HITS@3 in Tables 5.8 and 5.9. Note that HypE cannot deal with negation operator and thus no comparison on avg_n.

Logical Entailment Task. Tables 5.4 and 5.5 show the comparative results for logical entailment task over three benchmarks. Our findings are three-fold.

First, we obtain the similar observation from generalization reasoning task, BetaE and LinE consistently outperform other baselines (GQE and Q2B) across three benchmarks. For example, LinE achieves significant performance gain against GQE by 84.47% and 52.48% for avg_p and avg_l in MRR on NELL995, respectively. LinE achieves 50.36% and 9.63% relative gain over Q2B for avg_p and avg_l in MRR. As for FB15k-237, LinE achieves excellent performance gain against GQE by 105.87% and 71.10% for avg_p and avg_l in MRR, respectively. LinE achieves 29.51% and 16.59% relative gain over Q2B for avg_p and avg_l in MRR.

Second, LinE achieves considerable performance gain in most cases on three benchmarks compared to BetaE. In particular on NELL995, LinE shows superior reasoning ability on relation-heavy (24.81% gain in avg_p MRR), logic-heavy (20.48% gain in avg_l MRR) and negation-related queries (24.54% gain in avg_n MRR). The experiment results on FB15k-237 also prove that LinE has superior reasoning ability on relation-heavy (13.86% gain in avg_p MRR), logic-heavy (11.09% gain in avg_l MRR) and negation-related queries (9.78% gain in avg_n MRR). This consistently suggests that LinE generally shows superior reasoning ability, particularly relation-heavy and negation-related queries on logical entailment task as well.

Lastly, LinE shows superior reasoning ability for certain query types on logical entailment task. In Tables 5.8 and 5.9, LinE outperforms all baseline models on certain query types in both MRR and HITS@3 on NELL995, including "2p", "3p", "ip", "up", "2in", "3in", "inp", "pin" and "pni". As for FB15k-237, LinE also achieves significant performance for eight query types, including "3p", "ip", "2u", "up", "2in", "3in", "pin" and "pni" in terms of both MRR and HITS@3 as shown in Tables 5.6 and 5.7.

5.4 Hierarchical Logical Query (RQ2)

To study the reasoning ability for hierarchical logical queries, we adopt curvature estimates and Krackhardt scores to estimate the relational hierarchy for each relation ($r \in \mathcal{R}$) on three benchmarks.

5.4.1 Setup

The estimation suggests that WN18RR richly contains hierarchical relations, seven out of 11 relations as shown in Table 3.1, compared to FB15k-237 and NELL995. Thus, we additionally generate 14 types of FOL queries for WN18RR as shown in Table 5.2 to investigate the reasoning sensitivity to hierarchical logical queries.

Table 5.10: Performance comparison on in MRR (%) WN18RR. Best (second best) of each column are in bold (underlined). The last row shows relative improvement (%) of $\text{LinE}_{\alpha,\beta}$ compared to the best baseline.

Model	WN18RR													
	1p	2p	3p	2i	3i	ip	pi	2u	up	2in	3in	inp	pin	pni
Generalization Reasoning Task														
GQE	18.02	4.54	2.68	19.32	23.97	10.60	9.91	2.37	3.55	-	-	-	-	-
Q2B	22.46	4.63	2.31	25.59	41.23	11.04	13.27	2.89	3.47	-	-	-	-	-
BetaE	44.13	9.85	3.86	**57.19**	**76.26**	**17.97**	**32.59**	7.57	5.39	**12.77**	59.98	5.07	4.04	7.48
HypE	20.93	5.40	2.54	30.10	58.06	9.30	13.44	3.52	4.54	-	-	-	-	-
$\text{LinE}_{\alpha,\beta}$	**45.12**	**12.35**	**6.70**	47.11	67.13	14.73	24.87	**8.49**	**6.93**	12.50	**60.81**	**7.34**	**5.20**	**7.74**
Rel. Gain	2.24	25.38	73.58	-17.63	-11.97	-18.03	-23.69	12.15	28.57	-2.11	1.38	44.77	28.71	3.48
Logical Entailment Task														
GQE	35.64	9.05	6.28	32.47	32.09	17.26	19.77	7.43	5.45	-	-	-	-	-
Q2B	43.05	8.93	4.44	46.82	60.62	16.95	30.89	15.93	5.73	-	-	-	-	-
BetaE	82.18	22.14	8.57	**88.84**	95.71	31.21	**56.85**	25.82	12.53	31.12	77.08	9.48	7.51	19.73
$\text{LinE}_{\alpha,\beta}$	**87.79**	**28.08**	**13.82**	88.17	**96.32**	**33.26**	54.04	**30.74**	**15.56**	**37.26**	**79.51**	**11.19**	**11.12**	**24.49**
Rel. Gain	6.83	26.83	61.26	-0.75	0.64	6.57	-4.94	19.05	24.19	19.73	3.15	18.04	48.07	24.13

5.4.2 Results

In Tables 5.10 and 5.11, we observe that both LinE and BetaE outperform other baselines (GQE, Q2B and HypE) across query groups in both MRR and HITS@3 on generalization reasoning task. Compare to BetaE on relation-heavy queries, LinE achieves nearly 11.0% improvement in MRR (avg_p). Particularly, LinE significantly improves the answer accuracy for multi-hop relational queries like “2p” and “3p” with nearly 25.4% and 73.6% in MRR, respectively. For negation-related queries, LinE also slightly outperforms BetaE with 4.76% improvement in MRR (avg_n). For logic-heavy queries, BetaE occasionally outperforms LinE, which constitutes an overall 14.07% relative loss in MRR (avg_l). We leave this improvement for future work.

Moreover, we observe that both LinE and BetaE outperform other baselines on logical entailment task as well. Compare to BetaE on relation-heavy queries, LinE achieves 14.88% improvement in MRR (avg_p). Particularly, LinE significantly improves the answer accuracy for

Table 5.11: Performance comparison in HITS@3 (%) on WN18RR. Best (second best) of each column are in bold (underlined). The last row shows relative improvement (%) of LinE$_{\alpha,\beta}$ compared to the best baseline.

Model	**WN18RR**													
	1p	2p	3p	2i	3i	ip	pi	2u	up	2in	3in	inp	pin	pni
Generalization Reasoning Task														
GQE	30.86	4.68	2.57	31.67	37.90	11.02	13.30	2.25	2.77	-	-	-	-	-
Q2B	42.11	3.73	2.07	48.27	76.55	11.39	20.53	3.10	2.62	-	-	-	-	-
BetaE	46.01	10.42	3.92	**63.99**	**81.20**	**19.89**	**35.11**	7.90	5.73	13.12	65.75	5.01	4.21	7.33
HypE	36.07	5.00	2.32	38.59	73.75	10.73	16.36	3.17	3.98	-	-	-	-	-
LinE$_{\alpha,\beta}$	**47.08**	**12.87**	**6.56**	49.46	72.12	15.85	26.66	**9.40**	**7.24**	**13.22**	**66.56**	**8.25**	**5.79**	**8.04**
Rel. Gain	2.33	23.51	67.35	-22.71	-11.18	-20.31	-24.07	18.99	26.35	0.76	1.23	64.67	37.53	9.69
Logical Entailment Task														
GQE	66.30	8.85	6.24	72.38	63.15	18.45	34.77	4.50	4.26	-	-	-	-	-
Q2B	85.11	6.28	3.40	92.68	98.00	15.69	52.96	13.61	3.37	-	-	-	-	-
BetaE	89.61	22.70	8.13	**94.35**	**99.00**	33.65	**62.76**	28.81	12.59	34.82	**88.76**	9.68	6.59	20.71
LinE$_{\alpha,\beta}$	**93.16**	**29.37**	**13.83**	92.20	98.70	**36.77**	58.38	**34.59**	**16.08**	**42.35**	86.54	**11.23**	**10.96**	**25.29**
Rel. Gain	3.96	29.38	70.11	-2.28	-0.30	9.27	-6.98	20.06	27.72	21.63	-2.50	16.01	66.31	22.11

multi-hop relational queries like “2p” and “3p” with 29.38% and 70.11% in HITS@3, respectively. For negation-related queries, LinE also outperforms BetaE with 12.87% improvement in MRR (avg$_n$). For logic-heavy queries, LinE beats BetaE on logical entailment task with slightly improvement of 3.74% and 2.39% in MRR and HITS@3 (avg$_l$), respectively.

5.5 Logic Space Transformation (RQ3)

To evaluate the effectiveness of statistical signals, we study the formulations of neural transformation functions that project entity representations from the Beta distribution (BetaE) to the LinE space. We compare three types of statistical features, (α,β), (μ, var) and ($mode$), given the same MLPs setting. In Table 5.12, we observe that LinE$_{\alpha,\beta}$ outperforms LinE$_{\mu,var}$ across 14 query types on WN18RR. This suggests that overall LinE$_{\alpha,\beta}$ gives a more reliable performance with (α,β) as the primary statistical signals for logic space transformation with and without hierarchical relations. Note that LinE$_{mode}$ delivers poor performance in this experiment, and thus we do not list the experiment results for LinE$_{mode}$ in Table 5.12.

5.6 Relation and Logical Operators (RQ4)

To evaluate the effectiveness of relation and logical operators, we present a detailed study on mathematical formulations of the relational projection and union operator in the LinE space.

Table 5.12: Ablation study on WN18RR for logic space transformation in both MRR and HITS@3 (%).

Model	1p	2p	3p	2i	3i	ip	pi	2u	up	2in	3in	inp	pin	pni
							MRR							
LinE$_{\mu,var}$	44.02	9.70	4.63	44.84	65.12	13.20	21.95	7.26	5.49	11.98	58.78	5.87	4.21	7.55
LinE$_{\alpha,\beta}$	**45.12**	**12.35**	**6.70**	**47.11**	**67.13**	**14.73**	**24.87**	**8.49**	**6.93**	**12.50**	**60.81**	**7.34**	**5.20**	**7.74**
							HITS@3							
LinE$_{\mu,var}$	46.03	11.00	4.91	47.28	70.52	14.22	24.49	7.73	5.52	13.03	65.63	5.84	4.39	**8.27**
LinE$_{\alpha,\beta}$	**47.08**	**12.87**	**6.56**	**49.46**	**72.12**	**15.85**	**26.66**	**9.40**	**7.24**	**13.22**	**66.56**	**8.25**	**5.79**	8.04

5.6.1 Relation-specific Projection

We search for the optimal setting with 1,600 hidden dimensions and the two layers as our final MLPs setting. Table 5.4 and 5.5 report the average MRR (avg$_p$) and HITS@3 (avg$_p$) for relation-heavy queries (1p/2p/3p) across benchmarks. Our relational projections in LinE$_{\alpha,\beta}$ significantly outperform BetaE across benchmarks. For FB15k-237 benchmark, LinE achieves 5.71% and 6.43% relative gain in MRR and HITS@3, respectively. LinE delivers the best performance on NELL995 with 12.53% and 11.61% relative gain in MRR and HITS@3. LinE achieves 73.58% relative gain in MRR, particularly for the most complex queries "3p" on WN18RR as shown in Table 5.10.

5.6.2 Intersection

To perform the intersection operation, a simple approach is to use minimum function based on the properties of LinE (Eq.4.11) and Beta distribution [37]. Minimum function with attention weights is another valid formulation proposed by [37]. Another sophisticated formulation is proposed by [27], We evaluate all these formulations across benchmarks to discover the most optimal intersection formulation. As a result, the model that simply use minimum function without any attention weights (Eq.4.11) gives the best reasoning performance.

5.6.3 Union

DNF [36, 37] has proven superiority in the literature. Our union formulation (U) adopts a maximum function for a set of LinE embeddings. We study both (i) DNF, and (ii) U (Eq.4.12) to evaluate their effectiveness.

Table 5.13 shows that our LinE$_{\alpha,\beta}$ with DNF outperforms U on union queries (2u/up). As a result, the DNF is still the most effective formulation for union operators. Note that although U is slightly less effective than DNF for union queries, DNF takes extra computation to alter the logical query structures. Overall, our straightforward formulation is competitive to DNF.

Table 5.13: Ablation study on WN18RR for formulations of union operator in both MRR and HITS@3 (%).

Model	1p	2p	3p	2i	3i	ip	pi	2u **DNF**	2u **U**	up **DNF**	up **U**	2in	3in	inp	pin	pni
MRR																
LinE$_{\alpha,\beta}$	45.12	12.35	6.70	47.11	67.13	14.73	24.87	**8.49**	7.00	**6.93**	5.99	12.50	60.81	7.34	5.20	7.74
HITS@3																
LinE$_{\alpha,\beta}$	47.08	12.87	6.56	49.46	72.12	15.85	26.66	**9.40**	7.73	**7.24**	5.52	13.22	65.45	8.25	5.79	8.04

5.6.4 Negation

To preserve the closure property, prior work BetaE [37] employs reciprocal function as the result of negation for a Beta embedding. As LinE embedding is transformed from Beta embedding, we adopt the reciprocal function to deal with negation operators in the LinE space as well. The experiment results show that LinE can effectively deal with queries with negations (avg$_n$) compared to BetaE in both MRR and HITS@3 across two reasoning tasks. Specifically, LinE gives significant improvements for two types of queries: “inp” and “pin”. For example, LinE has nearly 27% relative gain of BetaE in MRR (“inp”) in generalization reasoning task, and nearly 36.5% relative gain of BetaE in MRR (“pin”).

Chapter 6

Conclusion

This chapter summarizes our work, identifies current limitations, and discusses possible future work. Section 6.1 highlights major achievements of our work. Section 6.2 identifies the main limitations of our work. Section 6.3 discusses the future work.

6.1 Achievements

We present a logical query reasoning framework (LinE) to preserve multi-relational complexities and logical regularities in the LinE space. The following list highlights the major achievements of our research.

- **Logic space transformation.** A logic space transformation component is proposed to relax strong distributional assumptions to better support relational and logical operations.

- **Relation regularization.** We design neural relation-specific projections to sensitively capture mixed relational properties in the KG guided by curvature estimate and Krackhardt score.

- **WN18RR queries.** We generate FOL queries of 14 types from WN18RR to investigate the reasoning sensitivity of LinE for hierarchical logical queries.

- **Reasoning performance.** Extensive experiments on three benchmarks demonstrate better reasoning sensitivity of LinE to answer logical queries with and without hierarchical relations against dominant baselines. LinE generally shows superior reasoning ability for relation-heavy and negation logical queries on both generalization reasoning and logical entailment tasks.

6.2 Limitations

Despite the achievements, there are still limitations to be addressed. In this section, we identify the following limitations of our work.

6.2.1 Enhancement for Logic-heavy Query

Despite the significant improvement of the proposed LinE framework in the answer accuracy for multi-hop relational queries (like 2p and 3p) and negation-related queries, LinE can be occasionally outperformed compared to BetaE on logic-heavy queries. In particular, on WN18RR, LinE constitutes an overall 14.07 and 15.49 relative loss on MRR and HITS@3, respectively. Therefore, improving the performance of LinE on logic-heavy queries is needed.

6.2.2 Logical Regularity Modeling

LinE shows unstable performance with intersection operators among all logical operators in logic-heavy queries. This may be caused by either limited logical regularities considered or poor formulations. This calls for considering a more comprehensive set of logical regularities and designing a more reasonable formulation for the intersection operator.

6.2.3 Computational Bottleneck

All experiments in this work are conducted on NVIDIA Quadro RTX 8000 GPU. We observe that BetaE requires higher computational resources for model training and evaluation time compared to other baselines due to its reasoning and representation complexities. LinE, unfortunately, demands more model training and evaluation time than BetaE. This leads to high time complexity and high GPU memory usage. The problem of improving computational resource utilization of LinE reasoning framework is needed.

6.3 Future Directions

We suggest the following as directions for future work.

6.3.1 Complex Logical Query Structures

Recent work [51] introduces a logical entailment dataset consisting of 301 complex logical query structures. This is more than 20 times larger than the current popular benchmark of 14 logical query structures introduced by BetaE [37]. Considering the explosive logical complexities in the benchmark [51], one of our future directions is to adapt our LinE reasoning framework to tackle the 301 complex logical query structures.

6.3.2 Hierarchical Regularity Modeling

Currently, our LinE reasoning framework proposed in Chapter 4 works well on multi-relational complexities, including both hierarchical and non-hierarchical relations. Considering plenty of prior work on knowledge graph embeddings to deal with hierarchical natures, we aim to enhance our relational regularization to better encapsulate multi-relational complexities. In particular, a better formulation of query loss for hierarchical logical queries so as to accurately ground the hierarchical regularization in the LinE space.

6.3.3 Evaluation Benchmarks

In Chapter 4.5, we have introduced a new logical entailment dataset based on WordNet across 14 typical query structures, which constitutes up to three benchmarks in total for evaluation purposes, i.e., FB15k-237, NELL995, and WN18RR. Another promising future direction is introducing a new logical entailment dataset of richer query structures by considering 301 logical query structures for other popular KG benchmarks, e.g., Yago. This contributes a new benchmark to the research community on logical query reasoning and raises the evaluation standard to focus on generalization performance.

6.3.4 Beyond the First-order Logic

The current queries LinE can reason with are purely first-order logic expressions. One of our future directions is to extend our LinE framework beyond first-order logic to deal with logical queries in natural language expressions. To tackle such queries (or questions), our LinE framework needs to understand and reason with natural language expressions apart from the presence of multi-relational and logical complexities.

Bibliography

[1] Abdalghani Abujabal, Rishiraj Saha Roy, Mohamed Yahya, and Gerhard Weikum. QUINT: Interpretable question answering over knowledge bases. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 61–66, 2017.

[2] Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. Complex query answering with neural link predictors. In *International Conference on Learning Representations (ICLR)*, 2021.

[3] Ben Athiwaratkun and Andrew Gordon Wilson. Hierarchical density order embeddings. In *International Conference on Learning Representations (ICLR)*, 2018.

[4] Ivana Balazevic, Carl Allen, and Timothy Hospedales. Multi-relational poincaré graph embeddings. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[5] Trapit Bansal, Da-Cheng Juan, Sujith Ravi, and Andrew McCallum. A2N: Attending to neighbors for knowledge graph inference. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

[6] Hannah Bast and Elmar Haussmann. More accurate question answering on freebase. In *ACM International Conference on Information and Knowledge Management (CIKM)*, page 1431–1440, 2015.

[7] Nikita Bhutani, Xinyi Zheng, and H V Jagadish. Learning to answer complex questions over knowledge bases with query composition. In *ACM International Conference on Information and Knowledge Management (CIKM)*, page 739–748, 2019.

[8] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems (NeurIPS)*, page 2787–2795, 2013.

[9] Andrew Carlson, Justin Betteridge, Richard C Wang, Estevam R Hruschka Jr, and Tom M Mitchell. Coupled semi-supervised learning for information extraction. In *ACM International Conference on Web Search and Data Mining (WSDM)*, page 101–110, 2010.

[10] Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. Low-dimensional hyperbolic knowledge graph embeddings. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 6901–6914, 2020.

[11] Hanxiong Chen, Shaoyun Shi, Yunqi Li, and Yongfeng Zhang. Neural collaborative reasoning. In *Proceedings of the International World Wide Web Conference (WWW)*, page 1516–1527, 2021.

[12] Xuelu Chen, Ziniu Hu, and Yizhou Sun. Fuzzy logic based logical query answering on knowledge graph. *arXiv preprint arXiv:2108.02390*, 2021.

[13] Meng-Fen Chiang, Ee-Peng Lim, Wang-Chien Lee, Xavier Jayaraj Siddarth Ashok, and Philips Kokoh Prasetyo. One-class order embedding for dependency relation prediction. In *International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, page 205–214, 2019.

[14] Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K. Reddy. Probabilistic entity representation model for reasoning over knowledge graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[15] Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K Reddy. Self-supervised hyperboloid representations from logical queries over knowledge graphs. In *Proceedings of the International World Wide Web Conference (WWW)*, page 1373–1384, 2021.

[16] Philipp Christmann, Rishiraj Saha Roy, Abdalghani Abujabal, Jyotsna Singh, and Gerhard Weikum. Look before you hop: Conversational question answering over knowledge graphs using judicious context expansion. In *ACM International Conference on Information and Knowledge Management (CIKM)*, page 729–738, 2019.

[17] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[18] Albert Gu, Frederic Sala, Beliz Gunel, and Christopher Ré. Learning mixed-curvature representations in product spaces. In *International Conference on Learning Representations (ICLR)*, 2018.

[19] Will Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. Embedding logical queries on knowledge graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, page 2030–2041, 2018.

[20] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. Knowledge graph embedding based question answering. In *ACM International Conference on Web Search and Data Mining (WSDM)*, page 105–113, 2019.

[21] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Annual Meeting of the Association for Computational Linguistics (ACL) and International Joint Conference on Natural Language Processing (IJCNLP)*, 2015.

[22] Magdalena Kaiser, Rishiraj Saha Roy, and Gerhard Weikum. Reinforcement learning from reformulations in conversational question answering over knowledge graphs. In *International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, page 459–469, 2021.

[23] David Krackhardt. Graph theoretical dimensions of informal organizations. In *Computational organization theory*, pages 107–130. 2014.

[24] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI Conference on Artificial Intelligence (AAAI)*, page 2181–2187, 2015.

[25] Lihui Liu, Boxin Du, Heng Ji, Cheng Xiang Zhai, and Hanghang Tong. Neural-answering logical queries on knowledge graphs. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, page 1087–1097, 2021.

[26] Xiaolu Lu, Soumajit Pramanik, Rishiraj Saha Roy, Abdalghani Abujabal, Yafang Wang, and Gerhard Weikum. Answering complex questions by joining multi-document evidence with quasi knowledge graphs. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, page 105–114, 2019.

[27] Francois Luus, Prithviraj Sen, Pavan Kapanipathi, Ryan Riegel, Ndivhuwo Makondo, Thabang Lebese, and Alexander Gray. Logic embeddings for complex query answering. 2021.

[28] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[29] Thomas Müller, Francesco Piccinno, Massimo Nicosia, Peter Shaw, and Yasemin Altun. Answering conversational questions on structured data without logical forms. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5902–5910, 2019.

[30] Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. Learning attention-based embeddings for relation prediction in knowledge graphs. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[31] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. A novel embedding model for knowledge base completion based on convolutional neural network. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*.

[32] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *International Conference on Machine Learning (ICML)*, page 809–816, 2011.

[33] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, page 6341–6350, 2017.

[34] Maximillian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *International Conference on Machine Learning (ICML)*, pages 3779–3788, 2018.

[35] Hongyu Ren, Hanjun Dai, Bo Dai, Xinyun Chen, Denny Zhou, Jure Leskovec, and Dale Schuurmans. Smore: Knowledge graph completion and multi-hop reasoning in massive knowledge graphs. *arXiv preprint arXiv:2110.14890*, 2021.

[36] Hongyu Ren, Weihua Hu, and Jure Leskovec. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In *International Conference on Learning Representations (ICLR)*, 2020.

[37] Hongyu Ren and Jure Leskovec. Beta embeddings for multi-hop logical reasoning in knowledge graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[38] Frederic Sala, Chris De Sa, Albert Gu, and Christopher Ré. Representation tradeoffs for hyperbolic embeddings. In *International conference on machine learning (ICML)*, pages 4460–4469, 2018.

[39] Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th annual meeting of the association for computational linguistics (ACL)*, pages 4498–4507, 2020.

[40] Tao Shen, Xiubo Geng, Tao Qin, Daya Guo, Duyu Tang, Nan Duan, Guodong Long, and Daxin Jiang. Multi-task learning for conversational question answering over a large-scale knowledge base. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2442–2451, 2019.

[41] Shaoyun Shi, Hanxiong Chen, Weizhi Ma, Jiaxin Mao, Min Zhang, and Yongfeng Zhang. Neural logic reasoning. In *ACM International Conference on Information and Knowledge Management (CIKM)*, page 1365–1374, 2020.

[42] Haitian Sun, Andrew O Arnold, Tania Bedrax-Weiss, Fernando Pereira, and William W Cohen. Faithful embeddings for knowledge base queries. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[43] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Machine Learning (ICML)*, 2019.

[44] Komal Teru, Etienne Denis, and Will Hamilton. Inductive relation prediction by subgraph reasoning. In *International Conference on Machine Learning (ICML)*, pages 9448–9457, 2020.

[45] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning (ICML)*, page 2071–2080, 2016.

[46] Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. Order-embeddings of images and language. In *International Conference on Learning Representations (ICLR)*, 2015.

[47] Luke Vilnis, Xiang Li, Shikhar Murty, and Andrew McCallum. Probabilistic embedding of knowledge graphs with box lattice measures. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

[48] Bingning Wang, Ting Yao, Weipeng Chen, Jingfang Xu, and Xiaochuan Wang. Comqa:compositional question answering via hierarchical graph neural networks. In *Proceedings of the International World Wide Web Conference (WWW)*, page 2601–2612, 2021.

[49] Kai Wang, Yu Liu, Dan Lin, and Quan Z. Sheng. Hyperbolic geometry is not necessary: Lightweight euclidean-based models for low-dimensional knowledge graph embeddings. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

[50] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI Conference on Artificial Intelligence (AAAI)*, page 1112–1119, 2014.

[51] Zihao Wang, Hang Yin, and Yangqiu Song. Benchmarking the combinatorial generalizability of complex query answering on knowledge graphs. *arXiv preprint arXiv:2109.08925 year=2021*.

[52] Gerhard Weikum. A core of semantic knowledge unifying wordnet and wikipedia. In *Proceedings of the International World Wide Web Conference (WWW)*, 2007.

[53] Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *nternational Conference on Learning Representations (ICLR)*, 2015.

[54] Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Annual Meeting of the Association for Computational Linguistics (ACL) and International Joint Conference on Natural Language Processing (IJCNLP)*, pages 1321–1331, 2015.

[55] Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. Learning hierarchy-aware knowledge graph embeddings for link prediction. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 3065–3072, 2019.

[56] Zhanqiu Zhang, Jie Wang, Jiajun Chen, Shuiwang Ji, and Feng Wu. Cone: Cone embeddings for multi-hop reasoning over knowledge graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[57] Zhao Zhang, Fuzhen Zhuang, Hengshu Zhu, Zhiping Shi, Hui Xiong, and Qing He. Relational graph neural network with hierarchical attention for knowledge graph completion. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 9612–9619, 2020.