Neural-based Mixture Probabilistic Query Embedding for Answering FOL queries on Knowledge Graphs

Xiao Long[1], **Liansheng Zhuang**[1]*, **Li Aodi**[1], **Shafei Wang**[2], **Houqiang Li**[1]
[1]University of Science and Technology of China, Hefei 230026, China
[2]Peng Cheng Laboratory, Shenzhen 518000, China
longxiao1997@mail.ustc.edu.cn; lszhuang@ustc.edu.cn

Abstract

Query embedding (QE)—which aims to embed entities and first-order logical (FOL) queries in a vector space, has shown great power in answering FOL queries on knowledge graphs (KGs). Existing QE methods divide a complex query into a sequence of mini-queries according to its computation graph and perform logical operations on the answer sets of mini-queries to get answers. However, most of them assume that answer sets satisfy an individual distribution (e.g., Uniform, Beta, or Gaussian), which is often violated in real applications and limit their performance. In this paper, we propose a Neural-based Mixture Probabilistic Query Embedding Model (NMP-QEM) that encodes the answer set of each mini-query as a mixed Gaussian distribution with multiple means and covariance parameters, which can approximate any random distribution arbitrarily well in real KGs. Additionally, to overcome the difficulty in defining the closed solution of negation operation, we introduce neural-based logical operators of projection, intersection and negation for a mixed Gaussian distribution to answer all the FOL queries. Extensive experiments demonstrate that NMP-QEM significantly outperforms existing state-of-the-art methods on benchmark datasets. In NELL995, NMP-QEM achieves a 31% relative improvement over the state-of-the-art.

1 Introduction

Answering FOL queries on KGs is a fundamental problem in KGs. It aims to find answer entities of given complex queries using operators including existential quantification($\exists$), conjunction($\wedge$), disjunction($\vee$), and negation($\neg$), which has attracted great attention from both academic and industry recently (Sun et al., 2019; Li et al., 2017; Dalvi and Suciu, 2007; Dong et al., 2020).

Early methods transform a FOL query into a computation graph like Figure 2(a), where each node represents a set of entities and each edge represents a logical operation. Then, these methods traverse the KG according to the computation graph to identify the answer set (Lin et al., 2018; Guo et al., 2018; Saxena et al., 2020; Sun et al., 2019). However, this type of approach suffers two major challenges. First, it has difficulties identifying the correct answers when some links are missing in KGs. Second, it needs to traverse all the intermediate entities on reasoning paths, which may lead to exponential computation costs.

Recently, query embedding (QE) methods have attracted much attention and keep them close enough (Hamilton et al., 2018; Ren et al., 2020; Guu et al., 2015; Zhang et al., 2021; Amayuelas et al., 2021). These methods divide a query into a sequence of mini-queries according to its computation graph, and perform logical operations on the answer set of each mini-query to get the answer. Existing QE methods differ in their way of modeling answer sets and logical operations, which can be divided into geometry-based and distribution-based categories. Geometry-based models represent each answer set as a "region" (e.g., box or cone) in Euclidean spaces, while distribution-based models usually leverage an individual distribution (e.g., Beta or Gaussian) to represent each answer set. Compared with early methods, QE methods do not need to track all intermediate entities, and can use the nearest neighbor search in the embedding space to quickly discover answers.

Though having achieved promising performance, most existing QE methods assume that answer sets satisfy an individual distribution. For example, geometry-based models (Ren et al., 2020; Zhang et al., 2021) assume that answer sets satisfy a uniform distribution in a region, and distribution-based models assume that answer sets satisfy the Gaussian distribution (Choudhary et al., 2021) or Beta distribution (Ren and Leskovec, 2020). In the real KGs, distributions of answer sets are often very

*Corresponding author.

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 3001 - 3013

complicated. As shown in the left of Figure 1, the embedding vectors of the answer set to the query (*US, Location_Contains, ?*) in NELL995 (Xiong et al., 2017) are scattered across two class centers: city-related center containing (e.g., *New York, Joplin, Pittsburgh*) and college-related center containing (e.g., *Cambridge, Yale University, Brooklyn College*). Similarly, shown in the right of Figure 1, the embedding vectors of the answer set to the query (*Sport_Baseball, Plays_Sport_reverse, ?*) in Freebase (Toutanova and Chen, 2015; Bollacker et al., 2008) are scattered across three class centers: pitcher-related center, coach-related center, and catcher-related center, which indicate different positions among baseball players. Obviously, it is difficult to model the answer set of each query with only one individual distribution, which limits the performance of existing methods.

Inspired by the above insights, this paper proposes a Neural-based Mixture Probabilistic Query Embedding Model (NMP-QEM), which uses mixed Gaussian distribution with multiple means and covariance parameters to encode the answer set of each query. Since mixed Gaussian distributions can approximate any random distribution arbitrarily well (Wu and Perloff, 2007), the proposed NMP-QEM can model more complicated answer sets' distribution in real KGs. Furthermore, due to the unboundedness of Gaussian random variables, the closed form of the negation operators of mixed Gaussian distributions is hard to define, which prevents answering the queries with negation. To tackle this problem, we propose neural-based logical operators of projection, intersection and negation for mixed Gaussian distributions, so that the proposed NMP-QEM can answer all the FOL queries. To summarize, the contributions of our works are as follows:

- We propose a novel NMP-QEM to answer FOL queries on KGs. To the best of our knowledge, NMP-QEM is the first attempt that models answer sets of queries with mixture distributions instead of individual distributions, which makes it more suitable for real KGs.
- To handle all the FOL queries, we propose neural-based logical operators of projection, intersection, negation for mixed Gaussian distributions, which has difficulty in defining the closed solution of negation operation.
- Extensive experiments on three popular benchmark datasets demonstrate that NMP-QEM achieves superior performances in answering FOL queries on KGs and significantly outperforms state-of-the-art baselines. In NELL995, NMP-QEM even achieves up to 31% relative improvement over the state-of-the-art.

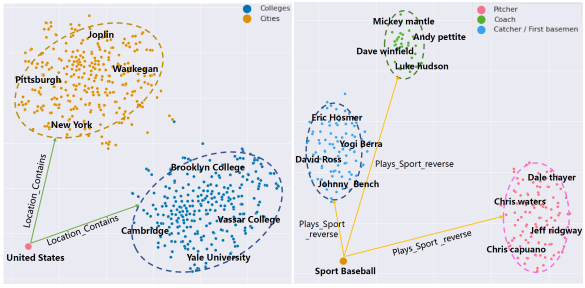

Figure 1: Visualization of the embedding vectors obtained from TransE (Bordes et al., 2013) with TSNE dimension reduction. Left: the answer set of the (United States, Location_Contains, ?). Right: the answer set of the (Sport Baseball, Plays_Sport_reverse, ?). The results show that there always exist multiple clusters in answer sets.

2 Related Work

In this section, we briefly review the related work. To answer complex logical queries on KGs, path-based methods (Lin et al., 2018; Guo et al., 2018) start from anchor entities and require traversing the intermediate entities on the path, which leads to exponential computation cost. Query embedding models are another line of work. Roughly speaking, existing query embedding methods can be divided into two categories – geometry-based models and distribution-based models.

Geometry-based models usually represent answer sets as "regions" e.g., points, boxes, or cones in Euclidean spaces, and then design set operations upon them. For example, Query2Box (Ren et al., 2020; Dasgupta et al., 2020) represents entities as points and queries as boxes. If a point is inside a box, then the corresponding entity is the answer to the query. Geometric shapes provide a natural and easy way to represent sets and logical relations among them. To better handle the queries with negation, recently (Zhang et al., 2021) use the closure of cones to design the negation operation. While (Amayuelas et al., 2021) directly uses MLP to define the negation operation and embed the query and the entities in the same vector space with single-point vectors.

Distribution-based models usually embed entities

and queries as an individual distribution e.g., Beta, Gaussian. (Ren and Leskovec, 2020) uses Beta distributions defined on the $[0, 1]$ interval to model entities, queries, and design neural logical operators that support full first-order logic. (Choudhary et al., 2021) encodes entities as a multivariate Gaussian density with mean and covariance parameters, and defines the closed logical operations upon the Gaussian distribution. Although it defines the logical operators for mixture input during chaining the queries, this model can not deal with the negation operation. Besides, it needs to calculate the inverse of the large matrix and solve a linear system, which requires huge memory and is very time-consuming.

Although these QE models make significant progress in answering FOL queries on KGs, they overlook that the distribution of answer sets in real KGs is often complicated. Thus, the proposed NMP-QEM models answer sets of queries with mixed Gaussian distributions instead of individual distributions, in order to approximate any distributions well in real applications.

3 Preliminaries

In this section, we first give the formal definition of answering FOL queries on KGs.

A KG can be denoted as $\mathcal{G} = (\mathcal{V}; \mathcal{R})$, where $v \in \mathcal{V}$ represents an entity, and $r \in \mathcal{R}$ is a binary function $r : \mathcal{V} \times \mathcal{V} \to \{True; False\}$, indicating whether the relation r holds between a pair of entities or not. The goal of answering FOL queries on KGs is to answer First-Order Logical (FOL) Queries on KGs. We can define them as follows:

Definition 1 (First-order logic queries) A first-order logic query q is formed by an anchor entity set $\mathcal{V}_a \subseteq \mathcal{V}$, an unknown target variable $V_?$ and a series of existentially quantified variables $V_1, \ldots, V_k$. In its Disjunctive Normal Form (DNF) (Davey and Priestley, 2002), it is written as a disjunction of conjunctions:

$$q[V_?] = V_? \cdot \exists V_1, \ldots V_k : c_1 \vee c_2 \vee \cdots \vee c_n \quad (1)$$

Specifically, c_i represents a conjunctive query of one or several literals: $c_i = e_{i1} \wedge \cdots \wedge e_{im}$. And the literals represent a relation or its negation: $e_{ij} = r(v_i, v_j)$ or $\neg r(v_i, v_j)$ where v_i, v_j are entities and $r \in \mathcal{R}$.

Computation Graph. Given a query, we represent the reasoning procedure as a computation graph (see Figure 2 a for an example), of which nodes represent entity sets and edges represent logical operations over entity sets. We map edges to logical operators according to the following rules.

Projection Operator $\mathcal{P}$. Given a set of entities $\mathcal{S} \subset \mathcal{V}$ and a relational function $r \in \mathcal{R}$, the projection operator $\mathcal{P}$ outputs all adjacent entities $\cup_{v \in \mathcal{S}} N(v, r)$ where $N(v, r)$ is the set of entities such that $r(v, v') = \text{True}$ for all $v' \in N(v, r)$.

Intersection Operator $\mathcal{I}$. Given n sets of entities $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$, the intersection operator $\mathcal{I}$ performs set intersection to obtain $\cap_{i=1}^{n} \mathcal{S}_n$.

Negation Operator $\mathcal{N}$. Given an entity set $\mathcal{S} \subset \mathcal{V}$, $\mathcal{N}$ gives $\bar{\mathcal{S}} = \mathcal{V} \backslash \mathcal{S}$.

It is worth noting that the union operation is unnecessary, as (Ren et al., 2020) shows that a union operator becomes intractable in distance-based metrics, and can be implemented using intersection and negation.

4 Methods

Overall, the architecture of the NMP-QEM and the corresponding computation graph are shown in Figure 2. To get answers, the complex query is divided into a sequence of mini-queries according to its computation graph. Then we use the neural-based operators to embed the entity and do logical operations between the answer set of each mini-query. In this query, the basketball players of America may play different positions (e.g., center, pg) and players who win the NBA championship may also come from different countries (e.g., Spain, France). In the following subsections, we will introduce the details of the proposed method.

4.1 Mixture Probabilistic Embedding for Entities and Queries

Firstly, we introduce how the NMP-QEM embeds entities and queries by which the output obey a mixture of Gaussian distributions. We assume that for the answer set V_i, there are K semantic centers. And the answer set is modeled by $V_i = \sum_{i=1}^{K} \omega_i \mathcal{N}(\mu_i, \Sigma_i)$, where a diagonal matrix is used instead of full covariance matrix of Gauss density function in consideration of computational cost. The learnable parameters $\omega_i \in \mathbb{R}$, $\mu_i \in \mathbb{R}^d$ and $\Sigma_i \in \mathbb{R}^d$ indicate the i^{th} component's weights, semantic position and spatial query area of answer sets, respectively. Finally, for the convenience of calculation, we concatenate three parameters and mask them as: $W = [[\mu_1; \omega_1], \ldots, [\mu_K; \omega_k], [\Sigma_1; 0], \ldots, [\Sigma_K; 0]] \in$

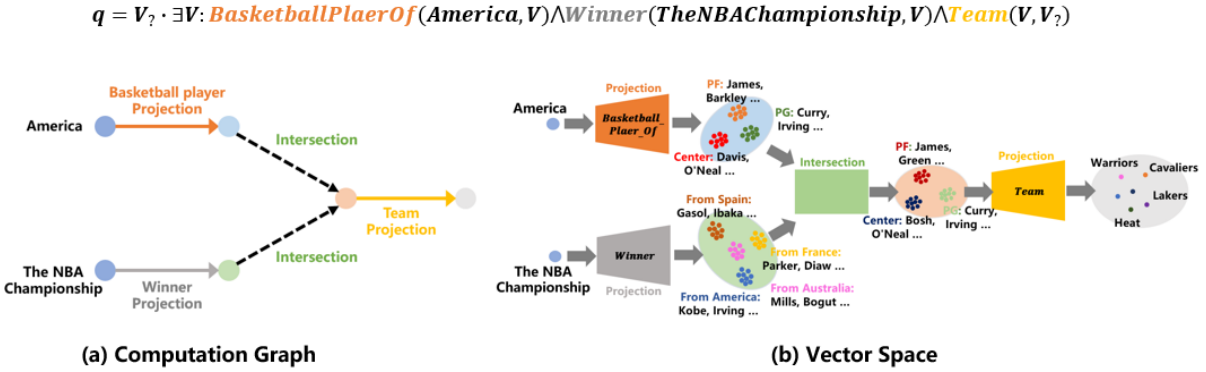

Figure 2: NMP-QEM Framework for answering FOL queries on KGs. Representation of a query: "List the teams of American basketball players who won NBA championship." (a) Query represented by its logical statements and dependency graph. (b) 2D Representation of the answer entities in the mixture Gaussian vector space used by the NMP-QEM. And answer sets often have multiple semantic centers.

$\mathbb{R}^{2K\times(d+1)}$, where $[\cdot;\cdot]$ indicates concatenation on the sequence length dimension. Next, each source node representing an anchor entity $v \in \mathcal{V}$ is a vector in $\mathbb{R}^{1\times d}$, and we regard it as the K identical components and then concatenate it into the same shape as $W \in \mathbb{R}^{2K\times(d+1)}$. Thus, we unify the dimensions of anchor entities and answer sets' embeddings to facilitate subsequent logical operations.

4.2 Neural-based Operators

To answer a query using the computation graph, we need projection operators and logical operators for the mixture probabilistic embedding. Next, we describe the design of these operators used in computation graphs, which include relation projection $\mathcal{P}$, intersection $\mathcal{I}$ and negation $\mathcal{N}$. As discussed before, the union can be implemented using intersection and negation. Accordingly, we do not define this operator.

Projection Operator $\mathcal{P}$: The goal of $\mathcal{P}$ is to map an anchor entity or an answer set to another answer set. Therefore, we design a probabilistic projection operator $\mathcal{P}$ that maps one mixture probabilistic embedding W to another mixture probabilistic embedding W' given the relation type r. We then learn a transformation neural network for each relation type r, which is implemented as a multi-layer perceptron (MLP):

$$W' = MLP_r(W) \tag{2}$$

Noticing that the output W' is also the mixture of Gaussian components, we must apply different activation functions to satisfy the following restrictions. The parameters of different component's weights ω_i should satisfy $\omega_i > 0$ and $\sum_{i=1}^{M} \omega_i = 1$. And the diagonal Σ_i should be positive semidefinite. So, we use the SoftMax function and Relu function to activate these parameters:

$$\omega_i = \frac{exp(\omega_i')}{\sum_{i=1}^{K} exp(\omega_j')}, \quad \Sigma_i = Relu(\Sigma_i') \tag{3}$$

Intersection Operator $\mathcal{I}$: Given n input embeddings $\{W_1, \ldots, W_n\}$, the goal of neural-based intersection operator $\mathcal{I}$ is to calculate the mixture probabilistic embedding W_{Inter} that represents the intersection of the distributions. Here we consider the intersection operation between two input embeddings $\{W_i, W_j\}$, since N input embeddings can be transformed into a pairwise intersection. The W_i contains K Gaussian components and the key of the $\mathcal{I}$ is how to make sufficient intersections between K Gaussian components. Thus, we model $\mathcal{I}$ by taking the attention mechanism (Vaswani et al., 2017). Additionally, considering that the intersection satisfies commutative law, we design a symmetric attention mechanism as follows:

$$\begin{aligned}\mathcal{I}\{W_i, W_j\} =& SoftMax(\frac{Q^{(i)}K^{(j)T}}{\sqrt{m}})V^{(j)} + \\ &+ SoftMax(\frac{Q^{(j)}K^{(i)T}}{\sqrt{m}})V^{(i)}\end{aligned} \tag{4}$$

where $Q^{(\alpha)} = (q_1^{(\alpha)T}, \cdots, q_K^{(\alpha)T}) \in \mathbb{R}^{K\times 2(d+1)}$, $K^{(\alpha)} = (k_1^{(\alpha)T}, \ldots, k_K^{(\alpha)T}) \in \mathbb{R}^{K\times 2(d+1)}$, $V^{(\alpha)} = (v_1^{(\alpha)T}, \ldots, v_K^{(\alpha)T}) \in \mathbb{R}^{K\times 2(d+1)}$. The input embeddings W_i are reshaped as $\hat{W}_i = [[\mu_1; \omega_1; \Sigma_1; 0], \ldots, [\mu_K; \omega_k; \Sigma_K; 0]] \in$

$\mathbb{R}^{K\times 2(d+1)}$, and the $Q^{\alpha}, K^{\alpha}, V^{\alpha}$ are calculated by:

$$\begin{aligned} Q^{(\alpha)} &= \hat{W}_i \bar{W}_Q, \\ K^{(\alpha)} &= \hat{W}_i \bar{W}_K, \\ V^{(\alpha)} &= \hat{W}_i \bar{W}_V, \end{aligned} \tag{5}$$

where $\alpha = 1,\cdots,n$. The $\bar{W}_Q \in \mathbb{R}^{2(d+1)}, \bar{W}_K \in \mathbb{R}^{2(d+1)}, \bar{W}_V \in \mathbb{R}^{2(d+1)}$ are parameter matrices. Observing the Equation 4, it is easy to find that: $\mathcal{I}\{W_i, W_j\} = \mathcal{I}\{W_j, W_i\}$. This result also guarantees the commutative law of intersection $\mathcal{I}$.

Negation Operator $\mathcal{N}$: Finally, we require a neural-based negation operator $\mathcal{N}$ that takes mixture probabilistic embedding W as input and produces an embedding of the $\mathcal{N}(W)$. This operator should reverse in the sense that regions of high probability in W have low probability in $\mathcal{N}(W)$ and vice versa. And we use a multi-layer $MLP_{\mathcal{N}}$ to model this transformation in distribution.

$$\mathcal{N}(W) = MLP_{\mathcal{N}}(W) \tag{6}$$

After the transformation in distribution, we also need to activate the parameters in $\mathcal{N}(W)$ like Equation 3, to satisfy the constraints of w and Σ.

4.3 Training Objective

The goal is to jointly train the neural-based logical operators and the entity embeddings. Our training objective is to minimize the distance between the entity embeddings and query embeddings, while maximizing the distance from the query to incorrect random entities, which can be done via negative samples. Equation 7 expresses this training objective in mathematical terms.

$$\begin{aligned} \mathcal{L} = &- log\sigma(\gamma - Dist(v;q)) - \\ &- \sum_{j=1}^{n} \frac{1}{k} log\sigma(Dist(v'_j;q) - \gamma)) \end{aligned} \tag{7}$$

where q is the query, $v \in [\![q]\!]$ is an answer of query (the positive sample); $v'_j \notin [\![q]\!]$ represents a random negative sample; $\gamma > 0$ is a fixed margin, and $\sigma(\cdot)$ is the sigmoid function. Both the margin and the number of negative samples k remain as hyperparameters of the model.

Distance: When defining the training objective, we need to specify the distance between the embeddings of entity v: E_v and the query q: W_q. We use the L_1 distance between source entity's embedding E_v and weighted μ of query q:

$$Dist(v,q) = |E_v - \sum_{i=1}^{K} W_q[\,\omega_i\,] \cdot W_q[\,\mu_i\,]| \tag{8}$$

where $W_q[\omega_i]$ and $W_q[\mu_i]$ denote the ω part and μ part of mixture probabilistic embedding W_q, respectively.

5 Experiments

In this section, we first introduce the experiment settings including datasets, baselines and evaluation protocols. Secondly, we compare NMP-QEM with competitive models on answering FOL queries over KGs and demonstrate its superiority. Thirdly, we reduce training queries to four types $(1p/2i/2in/inp)$ to observe the further generalization of the model. Then, we analyze the impact of two important parameters on the proposed model. Finally, we do the case study to exploit how the NMP-QEM models the answer sets in NELL995.

5.1 Experiment Setup

We adopt the commonly used experimental settings in (Ren and Leskovec, 2020). The only difference is that we used the WN18RR instead of the FB15K. For the reason that (Dettmers et al., 2018) raises WN18 and FB15K suffer test leakage through inverse relations problem. So they put forward the FB15K-237, WN18RR − a subset of FB15k and WN18, where inverse relations are removed. Accordingly, there is no need to use FB15K when FB15K-237 is already used.

Datasets and Queries: We use three datasets: FB15k-237, NELL995 and WN18RR. To obtain the queries from the datasets and their ground truth answers, we used the same query structures in (Ren and Leskovec, 2020). The training and validation queries consist of five conjunctive structures $(1p/2p/3p/2i/3i)$ and five structures with negation $(2in/3in/inp/pni/pin)$. Please refer to Appendix A.1 for more details about the datasets and query structures.

Evaluation Protocols: For each non-trivial answer v of a test query q, we rank it against non-answer entities $\mathcal{V}\backslash[\![q]\!]_{test}$. We denote the rank as r and then calculate the Mean Reciprocal Rank (MRR) and Hit@N. The definitions of these protocols are provided in Appendix A.3. Higher MRR and Hit@N indicate a better performance.

Baselines: We compare NMP-QEM against six state-of-the-art models, including GQE (Hamilton

Dataset	Model	1p	2p	3p	2i	3i	pi	ip	2u	up	AVG
FB15K-237	GQE	35.0	7.2	5.3	23.3	34.6	16.5	10.7	8.2	5.7	20.1
	Q2B	40.6	9.4	6.8	29.5	42.3	21.2	12.6	11.3	7.6	16.3
	BETAE	39.0	10.9	10.0	28.8	42.5	22.4	12.6	12.4	9.7	20.9
	ConE	41.8	12.8	11.0	32.6	47.3	25.5	14.0	14.5	10.8	23.4
	PRME	42.3	11.4	10.7	29.3	40.6	23.5	13.3	12.5	10.1	21.5
	MLP	42.7	12.4	10.6	31.7	43.9	24.2	14.9	13.7	9.7	22.6
	NMP-QEM	**46.2**	**12.9**	**11.3**	**35.0**	**47.8**	**25.6**	**15.1**	**15.0**	**10.9**	**24.4**
NELL995	GQE	33.1	12.1	9.9	27.3	35.1	16.7	10.9	8.5	9.0	18.7
	Q2B	42.7	14.5	11.7	34.7	45.8	23.2	17.4	12.0	10.7	23.6
	BETAE	53.0	13.0	11.4	37.6	47.5	24.1	14.3	12.5	8.5	24.6
	ConE	53.1	16.1	13.9	40.0	50.8	26.3	17.5	26.3	11.3	27.2
	PRME	43.6	14.2	11.6	38.8	46.8	24.3	15.2	13.3	9.2	24.1
	MLP	55.4	16.5	14.9	36.4	48.0	22.7	18.2	14.7	11.3	26.5
	NMP-QEM	**68.8**	**23.9**	**17.8**	**47.0**	**55.0**	**31.1**	**26.0**	**29.6**	**20.7**	**35.6**
WN18RR	GQE	21.2	7.8	3.6	29.7	35.7	18.3	13.5	3.4	5.1	15.3
	Q2B	24.8	8.4	3.8	43.3	74.5	25.0	14.1	4.5	5.4	22.0
	BETAE	44.7	15.7	5.3	57.8	77.5	36.3	16.6	7.9	6.7	29.8
	ConE	44.4	18.5	10.0	61.7	85.6	37.2	15.3	9.6	9.3	32.4
	PRME	44.9	16.6	8.9	60.3	75.9	36.9	17.7	7.7	7.2	30.6
	MLP	48.3	19.4	9.3	62.9	82.3	37.5	19.1	10.0	9.4	33.3
	NMP-QEM	**53.1**	**24.3**	**14.1**	**68.5**	**86.6**	**38.2**	**19.2**	**12.7**	**13.2**	**36.6**

Table 1: MRR results on answering EPFO ($\exists, \wedge, \vee$) queries. The bold values are the best results, and the underlined values are the second best results.

Dataset	Model	2in	3in	inp	pin	pni	AVG
FB15K-237	BETAE	5.1	7.9	7.4	3.6	3.4	5.4
	ConE	5.8	8.8	7.6	4.3	4.1	6.1
	MLP	6.6	10.7	8.1	4.7	4.4	6.9
	NMP-QEM	**6.8**	**11.7**	**8.2**	**5.5**	**4.7**	**7.4**
NELL995	BETAE	5.1	7.8	10.0	3.1	3.5	5.9
	ConE	5.6	8.1	10.9	3.5	3.9	6.4
	MLP	5.1	8.0	10.0	3.1	3.5	5.9
	NMP-QEM	**10.0**	**9.2**	**12.9**	**4.8**	**7.4**	**8.9**
WN18RR	BETAE	20.6	66.1	16.1	9.6	13.4	25.1
	ConE	22.6	68.4	18.9	11.9	14.2	27.2
	MLP	23.0	**68.5**	19.2	10.8	15.2	27.3
	NMP-QEM	**24.2**	68.2	**19.8**	**12.0**	**16.3**	**28.1**

Table 2: MRR results for answering queries with negation

et al., 2018), Query2Box (Q2B) (Ren et al., 2020), BETAE (Ren and Leskovec, 2020), ConE (Zhang et al., 2021), MLP (Amayuelas et al., 2021) and PREM (Choudhary et al., 2021). GQE, Q2B and PREM are trained only on five conjunctive structures as they cannot model the queries with negation.

Parameter Settings: For a fair comparison, we assign the best dimension to the embeddings of the six methods. We list the hyperparameters, architectures and more details in Appendix A.2

5.2 Main Results

In this subsection, we compare the MRR and HIT@1 of the NMP-QEM with all the baselines on three datasets. We run our model five times with different random seeds and report the average performance. Due to the limited space, the HIT@1 performance and error bars of the performance are shown in Table 9 and Table 10 in Appendix B.

The performance of all models is reported in Table 1 and Table 2. From the results, we have the following observations. Overall, NMP-QEM significantly outperforms compared models. For the EPFO ($\exists, \wedge, \vee$) queries, on average, NMP-QEM obtains 1.0% (4.2%relative), 8.4% (30.8% relative), and 3.3% (9.9% relative) improved MRR over the best models on FB15k-237, NELL995 and WN18RR respectively, which demonstrates the superiority of NMP-QEM on the whole. NMP-QEM also gains an impressive improvement on queries ($ip/pi/2u/up$), which are not in the training data. These results show the superior generality ability of NMP-QEM. Additionally, for the queries with negation ($2in/3in/inp/pin/pni$), NMP-QEM obtains 0.5% (7.2%relative), 8.4% (39.0% relative), and 0.8% (2.9% relative) improved MRR over the best baselines on three datasets respectively. Finally, we refer the reader to Table 7 and Table 8 in Appendix B for Hit@1 results. NMP-QEM also achieves better performance than six baselines on Hit@1. Furthermore, we found that NMP-QEM achieves a better improvement on NELL995 and WN18RR than FB15K-237, which may be caused by the entity-relationship ratio. In NELL995 and WN18RR, the entity-relationship ratio ($entity/relationship$) are 317 and 3681, re-

Model	2p	3p	3i	pi	ip	2u	up	3in	pin	pni	AVG	Descent Rate
BetaE	7.5	7.0	40.2	19.5	9.2	11.4	7.0	7.5	2.4	3.0	11.5	↓ 15.1%
ConE	8.4	7.2	43.6	21.1	9.4	14.1	7.3	9.5	**3.4**	3.9	12.7	↓ 16.4%
MLP	6.4	5.3	40.5	20.5	10.4	10.9	5.3	9.3	2.8	1.8	11.3	↓ 24.1%
NMP-QEM	**9.6**	**7.5**	**44.0**	**22.6**	**12.7**	**14.9**	**7.4**	**9.6**	3.1	**4.9**	**13.6**	↓ **15.0%**

Table 3: MRR results on FB15K-237 when training query structures are four types ($1p/2i/2in/inp$).

spectively, while in FB15K-237, the ratio is only 61. This phenomenon shows that with a relatively high entity-relationship ratio, the answer sets may have more semantic centers, which can be modeled better by NMP-QEM. We will discuss in detail the impact of the semantic center's number K on the model in the following sections.

5.3 Generalization of the models

In this subsection, we reduce training queries to four types to observe the further generalization of the model. Since in the previous 5.2 experiments, the model improves less on FB15K-237. Thus, we conduct more experiments on FB15K-237. Previous work (Ren and Leskovec, 2020) uses 10 types of queries for training as shown in Figure 6 in Appendix A. Then, they evaluate the model's generalization ability using queries with logical structures that the model has never seen during training, which includes $(ip/pi/2u/up)$ for evaluation. To further explore the model's generalization, we use $(1p/2i/2in/inp)$ queries for training and test the remaining 10 types of queries. We evaluate the MRR of the models and the results are shown in Table 3. We can find that although these ten types of queries did not appear in the training set, our model still achieve better performance nearly on all types compared with other methods. Moreover, compared with the results in section 5.2, which has more query types in the training set, NMP-QEM drops a relatively small proportion of average MRR, while other methods drop more on average MRR. These results demonstrate that NMP-QEM can generalize beyond query structures better than other models.

5.4 Influence of hyperparameters

As mentioned above, a good number of semantic centers K may bring the model a higher performance boost. Accordingly, in this subsection, we conduct two experiments to analyze the impact of important parameter K on the proposed model.

Firstly, we reduce the number of semantic centers to 1 which also means the model has only one Gaussian component. And we compare its average MRR results on all types of queries with those under the best K of the model. From Figure 3, we can find that no matter which datasets we choose, the model with multiple semantic centers always performs better than those with a single center. And this result further illustrates that the answer set of the query is more inclined to have multiple semantic centers. Additionally, we can find that on datasets with a higher entity-relationship ratio (NELL995 and WN18RR), the improvement is more significant than the FB15K-237. This result also shows that with a higher entity-relationship ratio, the answer set may have more semantic centers, which makes it modeled better.

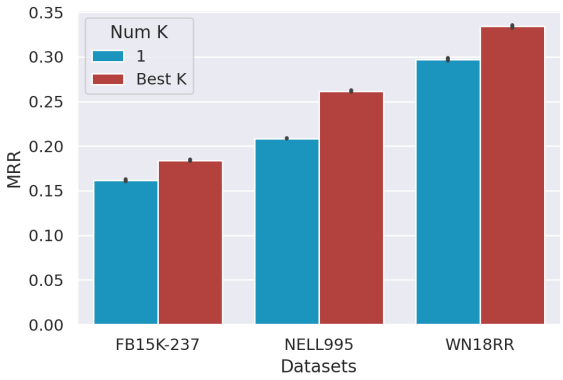

Figure 3: The average MRR results on all types of queries with those under the best K of NMP-QEM

Secondly, we change the number of semantic centers K in the range of {2, 4, 6, 8, 10, 20} to observe the average MRR results on all types of queries. From Figure 4, we can find that for different datasets, the appropriate K is also different. The value of K is related to the entity-relationship ratio. Specifically, for the FB15K-237, the entity-relationship ratio is relatively small, and the best K is only about 4. While for the WN18RR and NELL995, the entity-relationship ratio is relatively large, and the best K are 6 and 10 respectively. These results show that appropriate K is of great significance to the model's performance. If K is

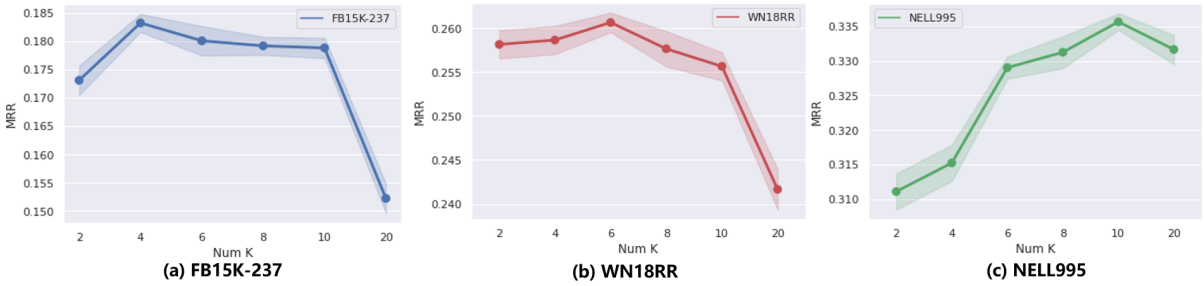

Figure 4: The average MRR results of NMP-QEM with different number of semantic centers K on three datasets

too large, there are more parameters to be trained, which may degrade the performance. If K is too small, the NMP-QEM can not model the high entity-relationship ratio dataset with many semantic centers.

5.5 Case Study

Finally, we explore how the NMP-QEM models answer sets in the test sample. We provide an example from NELL995 in Figure 5. The FOL query is "List the province of city Tazewell in the USA that proxy for", and the process of answering this question can be divided into four parts. We show each part's semantic center of the answer set and their weight ω_i at the bottom of the figure. Firstly, the answer set1 of the question $subpartof\_reverse(USA, V)$ are the different continents in the USA, and they have six semantic centers (different continents) since the K of the model is six on NELL995. Additionally, the six centers have similar weights ω_i, which means the probability of these components is similar. The second part is to answer $located\_in(Tazewell, V)$, and the answer set2 has five similar components related to $Illinois$. The sum of their weights reaches 0.885, which is close to the truth that $Tazewell$ is located in $Illinois$. Then, we intersect two answer sets above. The intersected set also has five similar components related to $Illinois$, and the sum of their weights reaches 0.784. Finally, we project the intersected set to obtain the final answers. We can observe that most answers are cities in $Illinois$, and the result given in red is the incorrect answer. From this example, we find that NMP-QEM can capture the semantic component in the answer set effectively, and the intersection operation can capture the same component of two sets, which helps NMP-QEM model the FOL queries better.

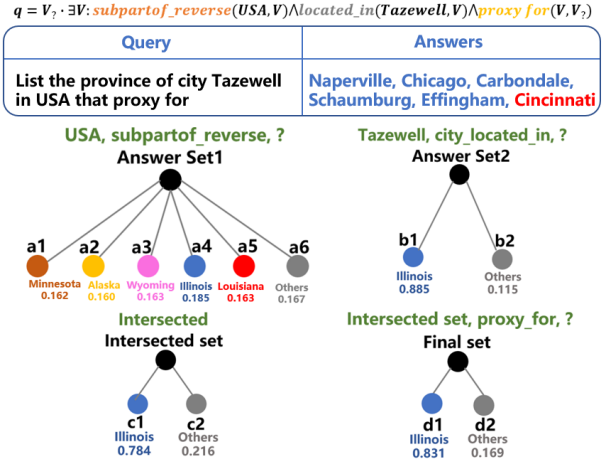

Figure 5: Examples of NMP-QEM model the answer sets in NELL995

6 Conclusions

In this paper, we propose a novel query embedding model NMP-QEM for answering FOL queries over knowledge graphs. The proposed NMP-QEM models the answer set of queries with mixed Gaussian distributions instead of individual distributions, which enables it to approximate any distributions well in real applications. Furthermore, to model the logic operators in FOL queries, we propose the neural-based logical operators of projection, intersection and negation for mixed Gaussian distributions, which are hard to be used to define the closed solution of negation operator. Extensive experimental results show a significant performance improvement compared to other state-of-the-art methods built for this purpose.

7 Limitations

In this section, we discuss the limitations of the proposed model. As mentioned above, (Amayuelas et al., 2021) used the vector to represent relation and defines the projection as the concatenation of the relation vector and query embedding. (Ren

et al., 2020; Zhang et al., 2021) used few parameters to represent the projection operators. Compared with these methods, our model may have more parameters and need more time during the training. This is because we define the neural-based projection operators P for each relation. So, if the amount of relation is too much, the parameters will increase accordingly. The Table 11 in Appendix B shows an example. But compared with PREM (Choudhary et al., 2021) which needs to calculate the inverse of the big matrix and solve a linear system, NMP-QEM uses less training time and memory. So, in the future, we will consider how to decrease the parameters in the projection operators P, which will reduce memory consumption during training and improve training speed.

Acknowledgement

This work was supported in part by Next Generation AI Project of China No.2018AAA0100602, in part to Dr. Liansheng Zhuang by NSFC under contract No.U20B2070 and No.61976199, and in part to Dr. Houqiang Li by NSFC under contract No.61836011.

References

Alfonso Amayuelas, Shuai Zhang, Xi Susie Rao, and Ce Zhang. 2021. Neural methods for logical reasoning over knowledge graphs. In *International Conference on Learning Representations*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

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

Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan Reddy. 2021. Probabilistic entity representation model for reasoning over knowledge graphs. *Advances in Neural Information Processing Systems*, 34.

Nilesh Dalvi and Dan Suciu. 2007. Efficient query evaluation on probabilistic databases. *The VLDB Journal*, 16(4):523–544.

Shib Dasgupta, Michael Boratko, Dongxu Zhang, Luke Vilnis, Xiang Li, and Andrew McCallum. 2020. Improving local identifiability in probabilistic box embeddings. *Advances in Neural Information Processing Systems*, 33:182–192.

Brian A Davey and Hilary A Priestley. 2002. *Introduction to lattices and order*. Cambridge university press.

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

Xin Luna Dong, Xiang He, Andrey Kan, Xian Li, Yan Liang, Jun Ma, Yifan Ethan Xu, Chenwei Zhang, Tong Zhao, Gabriel Blanco Saldana, et al. 2020. Autoknow: Self-driving knowledge collection for products of thousands of types. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2724–2734.

Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2018. Knowledge graph embedding with iterative guidance from soft rules. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. *arXiv preprint arXiv:1506.01094*.

Will Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. 2018. Embedding logical queries on knowledge graphs. *Advances in neural information processing systems*, 31.

Feng-Lin Li, Minghui Qiu, Haiqing Chen, Xiongwei Wang, Xing Gao, Jun Huang, Juwei Ren, Zhongzhou Zhao, Weipeng Zhao, Lei Wang, et al. 2017. Alime assist: An intelligent assistant for creating an innovative e-commerce experience. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 2495–2498.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. *arXiv preprint arXiv:1808.10568*.

Hongyu Ren, Weihua Hu, and Jure Leskovec. 2020. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. *arXiv preprint arXiv:2002.05969*.

Hongyu Ren and Jure Leskovec. 2020. Beta embeddings for multi-hop logical reasoning in knowledge graphs. *Advances in Neural Information Processing Systems*, 33:19716–19726.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. 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*, pages 4498–4507.

Haitian Sun, Tania Bedrax-Weiss, and William W Cohen. 2019. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. *arXiv preprint arXiv:1904.09537*.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pages 57–66.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Ximing Wu and Jeffrey M Perloff. 2007. Gmm estimation of a maximum entropy distribution with interval data. *Journal of Econometrics*, 138(2):532–546.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690*.

Zhanqiu Zhang, Jie Wang, Jiajun Chen, Shuiwang Ji, and Feng Wu. 2021. Cone: Cone embeddings for multi-hop reasoning over knowledge graphs. *Advances in Neural Information Processing Systems*, 34.

Appendix

A More Details about Experiments

In this section, we show more details about experiments that are not included in the main text due to the limited space.

A.1 Query Generation and Statistics

Statistics about the datasets can be found in Table 4. To obtain the queries from the datasets and its ground truth answers, we consider the 9 query basic structures (without negation) from Query2box (Ren et al., 2020) and additional 5 query structures (Ren and Leskovec, 2020) with negation to include FOL queries as shown in Figure 6. Given the 3 datasets, 3 graphs are created: $\mathcal{G}_{train} \subseteq \mathcal{G}_{val} \subseteq \mathcal{G}_{test}$ for training, testing and validation, respectively. Therefore the generated queries are also: $[\![q]\!]_{train} \subseteq [\![q]\!]_{val} \subseteq [\![q]\!]_{test}$. Thus, we evaluate and tune hyperparameters on $[\![q]\!]_{val} \backslash [\![q]\!]_{train}$ and report the results on $[\![q]\!]_{test} \backslash [\![q]\!]_{val}$. We always evaluate on queries and entities that were not part of the already seen dataset used before.

As summarized in Figure 6, our training and evaluation queries consist of the 5 conjunctive structures ($1p/2p/3p/2i/3i$) and also 5 structures with negation ($2in/3in/inp/pni/pin$). Furthermore, we also evaluate the model's generalization ability which means answering queries with logical structures that the model has never seen during training. We further include ($ip/pi/2u/up$) for evaluation.

For the experiments, we have used the train/valid/test set of queries-answers used in BetaE (Ren and Leskovec, 2020). This query-generation system differs from the original in the fact that it limits the number of possible answers to a specific threshold since some queries in Query2box (Ren et al., 2020) had above 5000 answers, which is unrealistic. And, for dataset WN18RR, we adopt the same generation process. The average number of queries is shown in Table 5.

A.2 Experimental Details

We implement our code using Pytorch. For the baselines, we have used the implementation of the baselines and the testing framework from Query2Box, GQE, BetaE, MLP, ConE and PREM. For a fair comparison with the models in the paper, we have selected the same hyperparameters listed in the paper. For our method, we fine-tune the hyperparameters including number of embedding dimensions from $\{200, 400, 800\}$, and the learning rate from $\{1e^{-4}, 7e^{-5}, 5e^{-5}, 1e^{-5}\}$, batch size from $\{128, 256, 512\}$, negative sample size from $\{32, 64, 128\}$, number of semantic centers K from $\{4, 8, 12, 20\}$ and the margin γ from $\{10, 20, 30, 40, 50, 60, 70\}$. For the datasets FB15K-237 and NELL995, we cite results in BETAE, ConE and MLP for comparison. But for dataset WN18RR, we conduct a new experiment using the best hyperparameters in their paper. Noticing that, PREM uses queries different from BetaE, which pointed out that there are unrealistic queries (Ren and Leskovec, 2020). So, for a fair comparison, we conduct new experiments with PREM on the three datasets. We list the hyperparameters of each model in Table 6. Every single experiment is run on a single NVIDIA Tesla V100 GPU, and we run each method for 400k iterations.

A.3 Evaluation Metrics

Given the rank of answer entities $v_i \in \mathcal{V}$ to a non-trivial test query q, we compute the evaluation metrics defined according to Equation 9 below. Then,

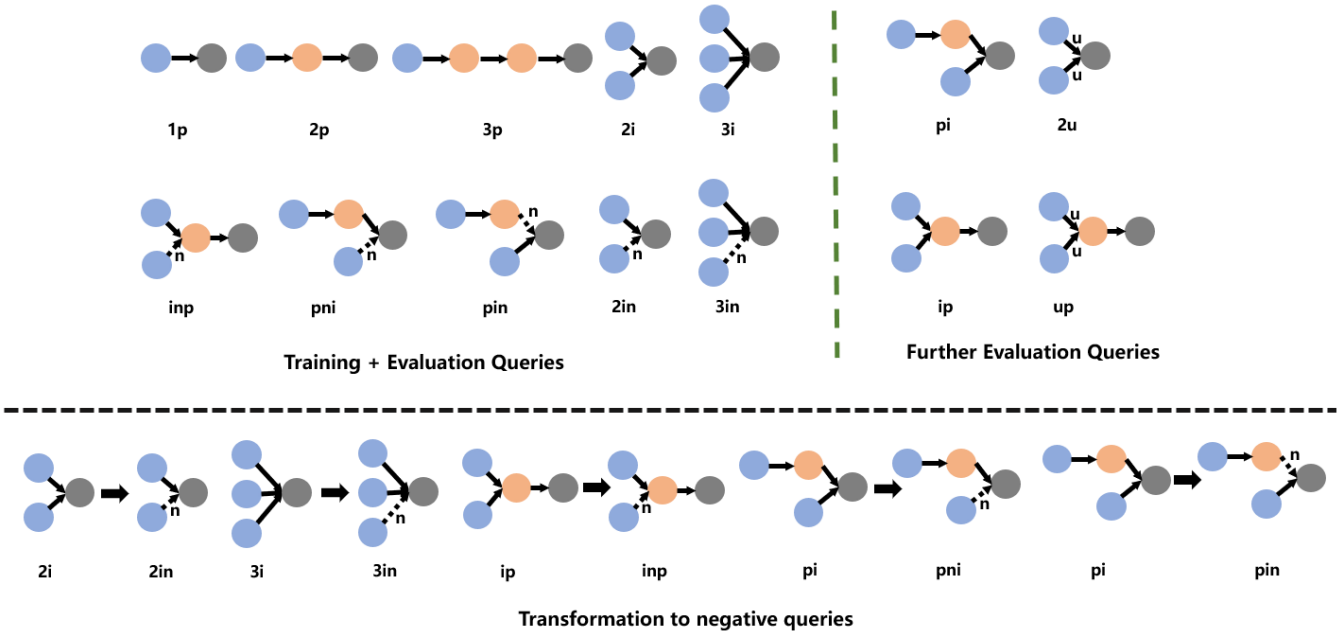

Figure 6: Top: Training and evaluation queries represented with their graphical structures, an abbreviation of the computation graph. We consistently use the following nomenclature: p projection, i intersection, n negation, and u union. Bottom: we show the transformation process from the original queries to their negative structure.

Dataset	Entities	Relations	Training Edges	Val Edges	Test Edges	Total Edges
FB15K-237	14,505	237	272,115	17,526	20,438	310,079
NELL995	63,361	200	114,213	14,324	14,267	142,804
WN18RR	40,493	11	86,835	3,034	3,134	93,003

Table 4: Knowledge graph datasets statistics as well as training, validation and test edge splits.

Queries	Training		Validation		Test	
Dataset	$1p/2p/3p/2i/3i$	$2in/3in/inp/pni/pin$	$1p$	others	$1p$	others
FB15K-237	149,689	14,968	20,101	5000	22,812	5000
NELL995	107,982	10,798	16,927	4000	17,034	4000
WN18RR	86,830	16,830	12,830	3000	12,830	3000

Table 5: Number of training, validation, and test queries generated for different query structures.

Models	embedding dim	learning rate	batch size	negative sample size	margin
GQE	800	0.0005	512	128	30
Q2B	400	0.0005	512	128	30
BetaE	400	0.0005	512	128	60
ConE	800	0.0001	512	128	30
MLP	800	0.0001	512	128	24
PREM	400	0.0001	512	128	24
NMP-QEM	400	0.00005	512	128	20

Table 6: Hyperparameters used for each method.

Dataset	Model	1p	2p	3p	2i	3i	pi	ip	2u	up	AVG
FB15K-237	GQE	22.4	2.8	2.1	11.7	20.9	8.4	5.7	3.3	2.1	8.8
	Q2B	28.3	4.1	3.0	17.5	29.5	12.3	7.3	5.2	3.3	12.3
	BETAE	28.9	5.5	4.9	18.3	31.7	14.0	6.7	6.3	4.6	13.4
	ConE	32.1	6.1	5.4	22.0	36.8	16.2	7.2	7.7	4.8	15.4
	PRME	31.5	5.9	4.9	20.9	33.3	15.8	6.9	6.5	4.5	14.4
	MLP	32.5	6.4	5.3	21.4	33.4	16.0	8.9	7.5	4.3	15.1
	NMP-QEM	**35.8**	**7.1**	**5.5**	**24.3**	**37.1**	**16.5**	**9.2**	**8.3**	**5.1**	**16.5**
NELL995	GQE	15.4	6.7	5.0	14.3	20.4	10.6	9.0	2.9	5.0	9.9
	Q2B	23.8	8.7	6.9	20.3	31.5	14.2	10.7	5.0	6.0	14.1
	BETAE	43.5	8.1	7.0	27.1	36.5	17.4	9.3	6.9	4.7	17.8
	ConE	42.9	10.7	8.8	27.7	39.4	18.6	11.7	8.6	6.6	19.4
	PRME	41.3	9.3	7.6	26.5	37.7	15.1	10.8	7.2	5.9	17.9
	MLP	45.6	11.2	10.0	25.3	36.7	15.4	12.4	8.6	6.5	19.0
	NMP-QEM	**60.2**	**15.2**	**12.1**	**33.7**	**44.0**	**20.0**	**16.9**	**18.0**	**12.1**	**25.8**
WN18RR	GQE	0.2	2.3	0.6	1.1	3.7	3.2	7.7	0.2	2.9	2.4
	Q2B	0.7	2.4	0.7	20.3	61.3	10.7	6.6	0.2	3.0	11.7
	BETAE	43.3	16.0	5.2	54.6	74.9	33.5	16.2	7.5	6.3	28.6
	ConE	43.9	19.1	9.5	53.0	78.9	34.3	14.8	9.9	8.9	30.2
	PRME	44.6	15.4	7.7	50.3	73.1	32.0	15.1	7.4	7.1	28.0
	MLP	46.5	18.1	9.3	59.4	**79.6**	34.4	16.0	9.9	10.8	31.5
	NMP-QEM	**47.5**	**20.0**	**9.6**	**59.7**	78.9	**34.5**	**17.3**	**9.9**	**11.1**	**32.1**

Table 7: HIT@1 results (%) of all the models on answering EPFO ($\exists, \wedge, \vee$) queries.

we average all queries with the same query format.

$$Metric(q) = \frac{\sum_{v_i \in \mathcal{V}} f_{metric}(rank(v_i)))}{|[\![q]\!]_{test} \backslash [\![q]\!]_{val}|} \quad (9)$$

where v_i is the set of answers $V \subset [\![q]\!]_{test} \backslash [\![q]\!]_{val}$, f_{metric} is the specific metric function and rank(v_i) is the rank of answer entities returned by the model. In our experiments, we use the flowing two f_{metric} functions.

Mean Reciprocal Rank (MRR): It is a statistic measure used in Information Retrieval to evaluate the systems that returns a list of possible responses ranked by their probability to be correct. Given an answer, this measure is defined as the inverse of the rank for the first correct answer, averaged over the sample of queries $\mathcal{Q}$. Equation 10 express this measure formally.

$$MRR = \frac{1}{|\mathcal{Q}|} \sum_{i=1}^{|\mathcal{Q}|} \frac{1}{rank_i} \quad (10)$$

where $rank_i$ refers to the rank position of the first correct answer for the $i-th$ query.

Hits rate at K (H@K): This measure considers how many correct answers are ranked above K. It directly provides an idea of how the algorithm is performing. Equation 11 defines this metric mathematically.

$$H@K = 1_{v_i \leq K} \quad (11)$$

B More Experimental Results

In this section, we give more experimental results that are not included in the main text due to the limited space.

We show in Table 7 and Table 8 the Hit@1 results of the three methods for answering FOL queries. Our method still shows a significant improvement over the six baselines in all three datasets.

To evaluate the performance of NMP-QEM, we run the model five times. We report the error bars of these results. Table 9 and Table 10 show the error bar of NMP-QEM's MRR results on all FOL queries. Overall, the standard variances are small, which demonstrates that the performance of NMP-QEM is stable.

Additionally, Table 11 shows the numbers of parameters for some models on FB15K-237.

Dataset	Model	2in	3in	inp	pin	pni	AVG
FB15K-237	BETAE	1.5	2.8	2.8	0.7	0.9	1.7
	ConE	1.7	3.7	3.1	1.2	1.0	2.1
	MLP	2.2	4.2	3.4	1.4	1.2	2.5
	NMP-QEM	**2.3**	**4.3**	**3.4**	**1.5**	**1.5**	**2.6**
NELL995	BETAE	1.6	2.6	4.4	0.9	1.1	2.1
	ConE	1.4	2.6	5.1	0.8	1.2	2.2
	MLP	1.4	2.6	4.2	0.9	1.1	2.0
	NMP-QEM	**2.9**	**2.9**	**6.1**	**1.3**	**2.4**	**3.1**
WN18RR	BETAE	17.1	59.1	8.6	3.9	9.5	19.6
	ConE	17.3	57.0	13.7	6.8	11.1	21.1
	MLP	18.2	60.1	17.2	8.3	11.7	23.1
	NMP-QEM	**17.6**	**60.5**	**19.8**	**7.5**	**12.0**	**23.4**

Table 8: HIT@1 results (%) of all the models on answering negative queries.

Dataset	1p	2p	3p	2i	3i	pi	ip	2u	up	AVG
FB15K-237	46.2	12.9	11.3	35.0	47.8	25.6	15.1	15.0	10.9	24.4
	±0.056	±0.086	±0.182	±0.132	±0.283	±0.232	±0.392	±0.332	±0.455	±0.067
NELL995	68.8	23.9	17.8	47.0	55.0	31.1	26.0	29.6	20.7	35.6
	±0.086	±0.236	±0.133	±0.245	±0.453	±0.216	±0.336	±0.343	±0.452	±0.089
WN18RR	53.1	24.3	14.1	68.5	86.6	38.2	19.2	12.7	13.2	36.6
	±0.029	±0.137	±0.024	±0.253	±0.179	±0.288	±0.290	±0.347	±0.235	±0.113

Table 9: The mean values and standard variances of NMP-QEM's MRR results on EPFO queries

Dataset	2in	3in	inp	pin	pni	AVG
FB15K-237	6.8	11.7	8.2	5.5	4.7	7.4
	±0.116	±0.131	±0.156	±0.096	±0.137	±0.129
NELL995	10.0	9.2	12.9	4.8	7.4	8.9
	±0.089	±0.052	±0.036	±0.071	±0.058	±0.076
WN18RR	24.2	68.2	19.8	12.0	16.3	28.1
	±0.096	±0.087	±0.374	±0.134	±0.156	±0.151

Table 10: The mean values and standard variances of NMP-QEM 's MRR results on queries with negation.

Models	Q2B	ConE	MLP	PREM	NMP-QEM
Parameter Numbers	682,280,0	238,904,01	247,920,00	262,764,800	198,126,776

Table 11: The numbers of parameters for some models on FB15K-237.