Project Digitalization - SoSe 24

Project Report

On

Evaluation of the NRN Model for Knowledge Graph Reasoning over Numerical Values - WP 3

Submitted by:

Clemens Friesl (1342071)

Mohamed Darkaoui (1311484)

Omme Salma (1420582)

Under the supervision of

Prof. Dr. Martin Simon, Emanuele Pepe

Frankfurt University of Applied Sciences

Faculty of Computer Science and Engineering

August 27, 2024

Abstract

The rising demand for processing large numerical data has exposed the limitations of current knowledge graph question-answering (KGQA) methods, especially in numerical reasoning. This report evaluates the Number Reasoning Network (NRN) model, which improves numerical complex query answering (Numerical CQA) by separately encoding entities and numerical values. We analyze the NRN model's performance using metrics like Mean Reciprocal Rank (MRR) and Hit@K across various datasets. Our results show that the NRN model surpasses baseline methods in numerical reasoning, despite challenges with resources and code comprehension. The report concludes with recommendations to enhance evaluation transparency and efficiency, highlighting NRN's potential in accurate numerical data analysis.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The ability to process and evaluate large volumes of numerical data is becoming increasingly important. Many real-word applications such as financial institutions, healthcare or e-commerce are required to organize and analyze their data efficiently. Knowledge graphs have become an emerging trend for structuring and reasoning data and are drawing attention from both the industrial-and scientific world [2]. In a knowledge graph, data is described as entities (nodes) and the relationships between them (edges). Furthermore the field of Knowledge graph question answering (KGQA) deals with reasoning and analyzing knowledge graphs. However, existing knowledge graph question answering methods are sometimes inaccurate and still have difficulties answering questions on numerical values and dealing with mathematical operations on them.

Over the course of this project we have therefore investigated the emerging topic of Numerical Complex Query Answering (Numerical CQA) based on the proposed method in [1], which aims to improve the processing of numerical values in Knowledge Graphs. Our aim was to understand the underlying topics of the paper to then give a reasonable evaluation on the proposed method. In addition to that we tried to reproduce the performance metrics contained in the paper by running the code that was provided by the authors.

The following report will introduce the problem statement of our project and will give a quick overview over our objectives. In the main part we will explain the most significant findings of our project and last but not least we will conclude the topic by making a reasonable assessment about the performance and applicability of the proposed method.

1.1 Aims and Objective

1.1.1 Problem Statement

In most knowledge graphs, edges are used to describe either the relationships between entities or their attribute values. These attribute values, which can be in categorical or numerical format are present in almost all knowledge graphs [3]. However, existing query encoding methods handle these numerical values in the same way as entities. This can lead to difficulties in answering numerical reasoning-related queries, as it fails to encode the semantic information of these values and cannot describe their distributional changes during computations and comparisons. Consequently, these encoding methods are unable to provide accurate reasoning for queries that require numerical value processing. This limitation makes it challenging to handle numerical data (e.g. dates, temperatures, carbon emissions) which are common and important in many knowledge graph based applications.

1.1.2 Objective

In this R&D project, we evaluated the NRN (Number Reasoning Network) model, which uses separate encoding methods for entities and numerical values to address challenges with numerical value reasoning in current query encoding methods. Our investigation specifically focused on its data preprocessing techniques for handling numerical data. Additionally, we assessed the strengths and weaknesses of the model concerning its applicability.

Chapter 2

Research Strategy and Approach

When we started the project our initial plan and objectives were focusing on the theoretical understanding of the overall topic. We wanted to take a deep dive into the subtopics that were mentioned in the paper to fully comprehend the technical aspects of the NRN model. We therefore made a plan which topics were worth exploring and started doing a broad literature research. After we began to get a better basic understanding of the topic, we realized that we would not reach our project goals by solely focusing on literature research.

We therefore changed our plan from exclusively doing research to combining research of the most important parts of the paper with understanding the code of the NRN model. After discussing the model, we decided it would be best to focus on the pre-processing part of the NRN model. That is why we then started to set up the code base the authors provided and concentrated on the parts where the pre-processing takes place. In addition to that we reduced our literature research to a minimum by choosing less but more fine-grained topics.

At this point of the project, the most important milestone was to identify and understand the most important parts of the code. Because of the time constraints of the project and the amount of code, we stuck to the following plan proposed in table 2.1 below. In the right column you will find questions or statements that we tried to answer and take into account to not get lost in the code.

After doing some code reading we always made sure to note down the findings in a way that the other team members could comprehend the content and that we would be able to include it in our final report.

Schedule	**Action**
Before reading code	What question do I want to answer? Which part of the model is most important for later evaluating the model?
While reading code	Is the code I am looking at answering my question?
After reading code	In which way does this answer contribute to the overall project goal?

Table 2.1: Code reading in the project

Chapter 3

Number Reasoning Network (NRN) Model

In this R&D project, we did a comprehensive analysis of the Number Reasoning network (NRN) model which is proposed by [1]. It is a Query encoding method that especially focuses on Numerical Complex Query Answering (Numerical CQA) aiming at answering the complex queries that need reasoning over numerical values. In this section, we will discuss the general overview of the architecture of the Number Reasoning Network (NRN) and will provide a detailed approach of the data preprocessing part of the model.

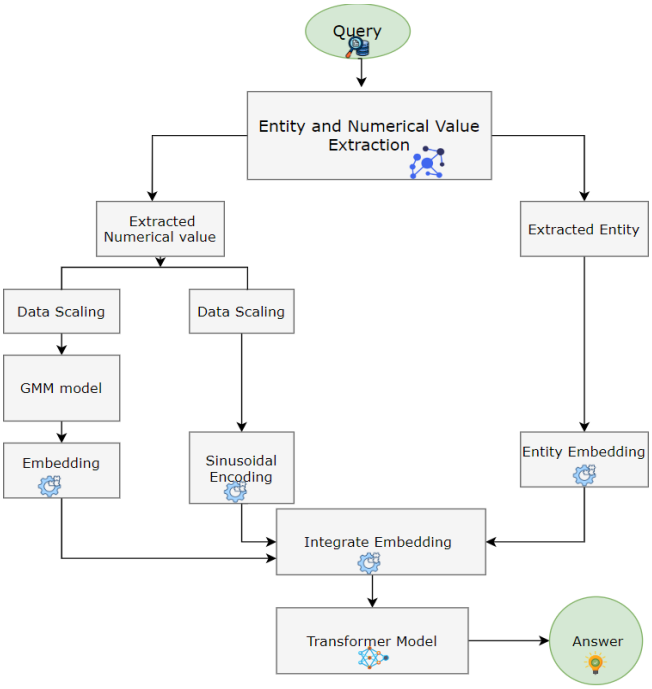

Figure 3.1: Overview architecture of NRN model

3.1 NRN Workflow Approach

In this section, we explain how the Numerical Reasoning Network (NRN) model works. Figure 3.1 represents the general overview diagram of the NRN model. The following sections will detail each step in the process.

3.1.1 General Workflow of NRN Model

1. **Identify and Extract Numerical Values:** In the NRN model, the first step is to identify and extract relevant numerical values (e.g., population size, GDP, latitude, temperature) from the knowledge graph. To identify these numerical values, four distinct types of projections, namely: Relational Projection, Attribute Projection, Numerical Projection and Reverse Attribute Projection are used [1, 4].

2. **Encoding Method Selection:** Based on the query requirements, the model selects an appropriate encoding method i.e entity embedding or numerical encoding method. This is predefined by projection selection. If a query involves reasoning on any numerical value, the tailored handling of pre-processing of numerical data is needed. For the encoding of entities, the model utilizes existing encoding methods such as QGE, Q2P, Q2B [5–7].

3. **Numerical data preprocessing and Embeddings :** There are two steps in the numerical data pre-processing part. These are :

   (a) **Data Normalization:** Before applying the numerical encoding method, the data needs to be normalized to a common scale, which leads to improved comparison and interpretation results. The model uses two different data normalization methods: Log-scaling and Quantile scaling.

   (b) **Numerical Encoding Method:** After Normalization, the data is transferred to the Gaussian Mixture Model (Parameterized Distributions) and the Sinusoidal Encoding method to transform the raw numerical values into their embeddings [8].

4. **Integrate Embeddings:** Here, numerical embeddings are combined with entity embeddings to create a unified representation and to ensure that they form the same embedding space [9].

5. **Transformer Model and Numerical Reasoning:** In this stage, the combined embedding are fed into the transformer model as model input. The transformer model processes these embedding to perform reasoning and to answer the complex queries involving numerical conditions [10]. Finally the model provides answers to these queries.

3.1.2 Query Processing in NRN

This example illustrates how the NRN model answers a query such as "Find cities with a population greater than 1 million and average temperature above 25°C". To answer this specific query, the NRN model first extracts the population and temperature attributes for cities. Next, it normalizes these values using log-scaling and Quantile scaling. After normalization, it encodes the data using the Gaussian Mixture Model (e.g., parameterized distributions) and Sinusoidal Embedding [11]. The next steps involve calculating the aggregate population size and average temperature for each city, applying the conditions (e.g., greater than 1 million for population, above 25°C for temperature) and reasoning over these numerical representations to identify cities that meet the query criteria.

3.2 Numerical Data Pre-processing in NRN

Existing query encoding methods consider numerical values in the same way as entities [5], [7]. Although these methods provide good performance results on queries related to relations between entities, they are unable to reasonably deal with numerical values. This is due to the following reasons:

- They are unable to encode the semantic information of numerical values.
- They are unable to describe the distributional changes of values when there are computations and comparisons between numerical values.

According to [1], the NRN model is able to deal with such relations involving numerical values. They incorporate new numerical variables and special operations to answer queries involving numerical values to the existing problem scope.

3.2.1 Numerical Values Handling in NRN

The main focus of our project was to fully comprehend the pre-processing part of the NRN model because of its importance regarding the two seperate encoding phases. As discussed in the last sections, knowledge graphs consist of connections of multiple nodes. These nodes can be entities or numerical values. The approach of the NRN model is to encode entities and numerical values in different ways. Entities are encoded using one of the ”baseline” methods, which are existing models (e.g. GQE, Q2P, Q2B) while applying a more sophisticated technique for numerical values. The following example illustrates how the NRN model treats entity and numerical values in different ways:

“Find the Carbon Emission of X transportation company before the year of 2024”

To answer this query, the NRN model encodes the semantics of the year 2024 such that the model can compute the distributions of the years before 2024. Existing query encoding methods consider the year 2024 attribute as an entity (i.e. only learn the graph features of 2024) and ignore its numerical meaning or semantic meaning. As the existing query encoding methods only consider entities, they are insufficient to deal with the relations where reasoning over numerical values is needed. To provide better reasoning results on complex queries involving numerical values, the paper [1] introduces a model with four different projections for numerical data pre-processing. As the existing methods only make use of relational projections between entities, they are insufficient to deal with the relations involving numerical values.

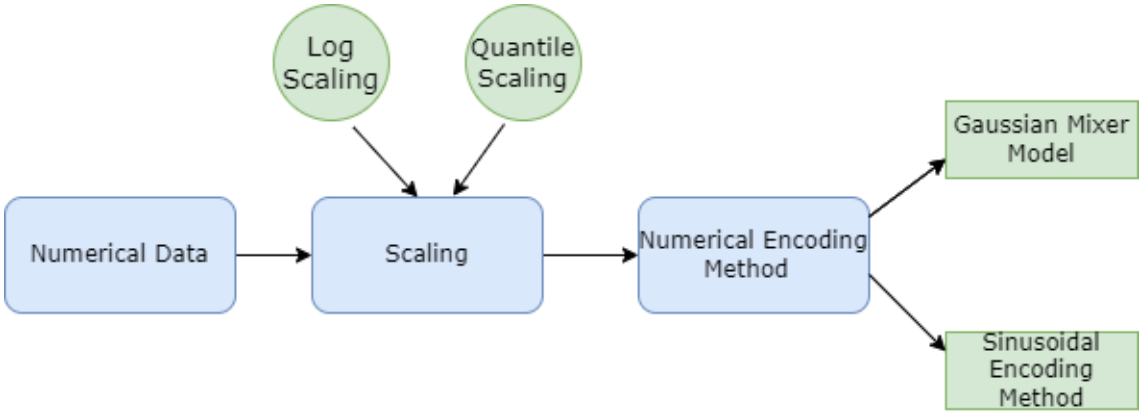

Figure 3.2: Numerical Value Handling Process

This diagram 3.2 shows the numerical value handling process in NRN. The following

section will describe the numerical data pre-processing steps and methods used in more detail.

3.2.2 Data Scaling Methods

Data scaling represents normalizing the data range to a common scale, making it easier to compare and interpret results [12]. In a knowledge graph, different types of numerical data (e.g., ages, salaries, years) can have varying ranges, which may lead to features with larger ranges having more influence during model training and reasoning. By applying data scaling, it is ensured that each feature contributes equally to the model. In the NRN model, two different data scaling methods are used. These are as follows:

1. Log Scaling

Log Scaling is used to normalize the range of data using logarithmic functions. Data with large differences in scales can have unstable variance. This variability can make it challenging for models to learn patterns. Log scaling helps to stabilize this variance by normalizing the data range. In addition to that, it helps to reduce the impact of outliers as extreme values are brought closer to the mean, making the model less sensitive to outliers and making the distribution more symmetric [12].

2. Quantile Scaling

Quantile scaling is a data scaling method used to transform raw input values into a normalized range based on its percentile rank within the dataset. This method is needed when data distribution is not normal or skewed and it has to be normalized based on the relative position rather than absolute values.

3.2.3 Numerical Encoding Method

As explained in section 2, the model uses two numerical encoding techniques (Sinusoidal Encoding and Gaussian Mixture Model) to project attribute values from R to R^D and obtain numerical embeddings. These methods convert numerical values into embeddings that capture their meaning and relationships. The semantic embeddings then serve as input

to the transformer model for reasoning. The paper [1] uses the following two numerical encodings methods.

1. Gaussian Mixture Model

Gaussian Mixture Model (GMM) is a probabilistic model used to represent the distribution of numerical data into multiple gaussian components with different mean and covariance matrices. It uses parameterized density functions (PDF) to model complex distributions in data and understand their underlying patterns. A PDF is a mathematical function that provides probabilities of a given numerical data point x, belonging to a certain cluster or component, where high probabilities are selected as the answer [13]. The NRN model makes use of the "Gaussian Mixture Model" function from the python library scikit-learn with 20 numbers of gaussian components [1] . For a numerical value x, the GMM equation is given by:

$$\mathcal{N}(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{3.1}$$

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k) \tag{3.2}$$

where:

- K is the number of Gaussian components.
- π_k is the weight of the k-th Gaussian component
- $\mathcal{N}(x \mid \mu_k, \Sigma_k)$ is the Gaussian distribution for the k-th component with mean μ_k and covariance matrix Σ_k.

Knowledge graph data often contains noise and variability, leading to incorrect reasoning. The Gaussian Mixture Model (GMM) helps address this uncertainty by representing data as probabilistic distributions rather than fixed points [14]. For instance, in a knowledge graph that contains information about carbon emissions (Co2) from various industries, GMMs can model how emissions are distributed across different sectors. By identifying clusters, GMMs can identify patterns such as industries with low emissions, moderate emissions, and high emissions, based on their carbon emission distributions. This allows for targeted sustainability efforts and more effective policy-making. Therefore, this approach allows for more robust reasoning and accurate query answering.

2. Sinusoidal Encoding

Sinusoidal encoding is often used in the context of positional encoding in transformer models introduced by [11] . It encodes positional information about numerical data within a dataset which assigns unique positional embedding by sine and cosine functions [15]. In the paper [1] , the authors discovered that this method is also applicable for encoding numerical values, as it preserves geometric properties from the original value space to the embedding space. Its formula is expressed by:

$$\psi(v)_d = \begin{cases} \sin\left(\frac{v}{n^{d/D}}\right) & \text{if } d \equiv 0 \pmod 2 \\ \cos\left(\frac{v}{n^{(d-1)/D}}\right) & \text{if } d \equiv 1 \pmod 2 \end{cases} \tag{3.3}$$

Here, v is the numerical value to be encoded, d and $(d-1)$ is the dimension index and D is the total dimensionality of the encoding. These encoded values allow the model to understand both the relative position and its relationship to other positions in the dataset. Numerical comparisons based on relative position are comparatively better than arbitrary indexing.

3.2.4 Entity Encoding Method

As mentioned in section 2, knowledge graph edges describe either the relationships between entities or their associated numerical values. During the entity encoding phase, the NRN model uses one of the existing query encoding methods (GQE, Q2B, Q2P) to encode entities. These methods convert entities into fixed-length real-valued vectors. Each method GQE, Q2B, and Q2P differs in its approach to encoding logical queries within the embedding space. The following provides a summary of these methods:

- **GQE:** This encoding method known as Graph Query Encoding, transforms logical queries into a single vector [5]. This single vector captures all the essential information of the query.
- **Q2B:** Query2Box encodes queries into hyper-rectangles (Boxes) within the embedding space [7].
- **Q2P:** Query2Particles is a query encoding method that encodes logical queries as multiple vectors in an embedding space [6].

3.3 Model Evaluation

The following section discusses the experimental results of the NRN model on FB15k, DB15k, and YAGO15k [16]. Two different metrics were used to compare the NRN models to the other baseline models. The first method is MRR (Mean Reciprocal Rank) and and the second is Hit@k. These metrics help to measure the quality of the rankings produced by the model in response to queries and compare the performance of different models. It is important to mention that the following results are only experimental and could not be reproduced from us. It is therefore difficult to make precise statements about the performance of the model.

3.3.1 Evaluation Metrics

1. MRR

Mean reciprocal rank (MRR) measures the number of triples predicted correctly. If the first predicted triple is correct, then 1 is added, if the second is correct ½ is summed, and so on [9]. It helps to decide which model is useful for practical applications where the ranking of retrieved information is important. Evaluation metrics of the test query q can be expressed with the following equation:

$$\text{Metric}(q) = \frac{1}{|[q]_{\text{test}}/[q]_{\text{val}}|} \sum_{v \in [q]_{\text{test}}/[q]_{\text{val}}} m(\text{rank}(v)) \tag{3.4}$$

Here, $m(r) = \frac{1}{r}$ and $\frac{1}{|[q]_{\text{test}}/[q]_{\text{val}}|}$ is the set of answers on the test graph but not in the validation graph.

2. Hit@K

The Hit@k is a another popular metric used to evaluate the accuracy of a model's predictions. It measures the proportion of queries for which at least one relevant item appears within the top K ranked predictions generated by the model [9]. Hit@1 indicates whether the correct answer is ranked as the top prediction, similarly Hit@3 if it's within the top 3 predictions, and Hit@10 if it's within the top 10 predictions. Therefore, this metric helps to evaluate and compare the performance of the model in terms of how well the system

can retrieve the correct answer. From equation 3.4, evaluation metrics of the test query q can be calculated as:

$$m(r) = \begin{cases} 1 & \text{if, } r \leq k \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

3.3.2 Benchmark Dataset Construction

Dataset

To construct the benchmark dataset, three different knowledge graphs FB15k, DB15k, and YAGO15 were used. Each graph differs in terms of the number of nodes, relationships, attributes, and edge types. The following figure 3.3 shows the statistics of these knowledge graphs:

Graphs	Data Split	#Nodes	#Rel.	# Attr.	#Rel. Edges	#Attr. Edges	#Num. Edges	#Edges
FB15k	Training	25,106	1,345	15	947,540	20,248	27,020	1,015,056
	Validation	26,108	1,345	15	1,065,982	22,779	27,376	1,138,916
	Testing	27,144	1,345	15	1,184,426	25,311	27,389	1,262,437
DB15k	Training	31,980	279	30	145,262	33,131	25,495	237,019
	Validation	34,191	279	30	161,978	37,269	25,596	262,112
	Testing	36,358	279	30	178,394	41,411	25,680	286,896
YAGO15k	Training	32,112	32	7	196,616	21,732	26,616	266,696
	Validation	33,078	32	7	221,194	22,748	26,627	293,317
	Testing	33,610	32	7	245,772	23,520	26,631	319,443

Figure 3.3: Statistics of Dataset Construction [1]

Distribution of Numerical Values in dataset

The knowledge graphs FB15k, DB15k, and YAGO15k contain various types of numerical values, including integers, floats, doubles, and degrees. Figure 3.4 shows the distribution of these numerical values across the graphs. Understanding this distribution helps us see how numerical values are represented and their frequency in each graph. Consequently, this affects the methods used for encoding and processing numerical data.

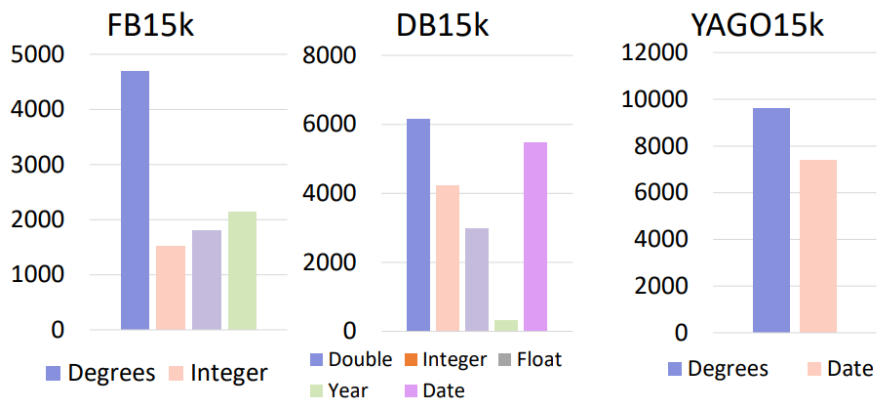

Figure 3.4: Types of numerical values in FB15k, DB15k, and YAGO15k [1]

3.3.3 Evaluation Results

In the experiment, they evaluated the generalization capability of the NRN model by computing the rankings of the answers that cannot be directly searched from an observed knowledge graph. Figure 3.5 represents four different evaluation metrics scores (e.g. Hit@1, Hit@3, Hit@10, and MRR). The experiment results show that the Number Reasoning network (NRN) provides better reasoning results compared to the baseline methods (GQE, Q2P, Q2B) on three different knowledge graphs. The main difference is that the NRN uses parameterized distributions to encode the numerical values instead of using a unified query encoding method. As a result, from the empirical results, they conclude that parameterized distributions are effective encoding structures for numerical values. On the other hand, Sinusoidal is an effective method to encode the magnitude of numerical attribute values. According to [1], the NRN performs better than the baseline methods on almost all of the three datasets.

Dataset	Entity	Value	H@1	H@3	H@10	MRR
FB15k	GQE	Baseline	13.56	23.47	34.80	20.80
		NRN + DICE	14.16	24.69	36.18	21.72
		NRN + Sinusoidal	**14.33**	**24.81**	**36.25**	**21.86**
	Q2B	Baseline	14.49	25.96	38.20	22.56
		NRN + DICE	15.55	27.31	39.24	23.72
		NRN + Sinusoidal	**15.78**	**27.65**	**39.54**	**23.96**
	Q2P	Baseline	15.54	25.50	36.88	22.84
		NRN + DICE	16.98	27.02	38.36	24.27
		NRN + Sinusoidal	**17.72**	**27.96**	**39.23**	**25.09**
DB15k	GQE	Baseline	6.86	12.98	21.28	11.73
		NRN + DICE	7.26	13.36	21.81	12.15
		NRN + Sinusoidal	7.30	13.63	**22.32**	**12.34**
	Q2B	Baseline	7.47	14.17	22.81	12.66
		NRN + DICE	**8.07**	**15.21**	**24.22**	**13.53**
		NRN + Sinusoidal	8.05	15.04	24.05	13.45
	Q2P	Baseline	8.41	14.57	22.84	13.31
		NRN + DICE	8.81	15.10	23.27	13.73
		NRN + Sinusoidal	**9.13**	**15.69**	**24.22**	**14.25**
YAGO15k	GQE	Baseline	10.56	18.12	27.65	16.35
		NRN + DICE	11.68	19.49	29.05	17.59
		NRN + Sinusoidal	**11.78**	**19.74**	**29.12**	**17.73**
	Q2B	Baseline	13.47	22.67	32.56	20.02
		NRN + DICE	13.94	23.74	33.56	20.77
		NRN + Sinusoidal	**14.41**	**23.96**	**33.78**	**21.13**
	Q2P	Baseline	6.70	11.98	20.11	11.29
		NRN + DICE	9.80	**16.98**	**26.74**	**15.53**
		NRN + Sinusoidal	**9.90**	16.83	26.43	15.49

Figure 3.5: Main experiment results of NRN on query answering [1]

Chapter 4

Model Challenges

Throughout the course of this project, we faced several issues that cost us a lot of time and affected our quality of results. The next section will provide an overview of the most critical problems we encountered.

4.1 Code comprehension

Many parts of the code lacked meaningful comments or explanations and made it difficult for us to comprehend important steps of the process. We spent a lot of time trying to figure out single lines of code to put them into context. We decided to therefore not get hung up on one line of code but rather continue and view parts of it as a “black box”.

4.2 Limited resources

Running the code that the authors used in their experiment on our personal laptops would have taken up to 72 hours and was not feasible for us. The hardware we had at hand was missing resources to cope with the resource-intensive application. In addition to that, the model had to be re-trained every time when starting the program code, which made it really hard to move forward. All in all, we were only able to run the code using small datasets that were additionally provided by the authors but could not provide meaningful insights.

4.3 Virtual Machine setup

Because of the resource-intensive code of the NRN-model, we requested a virtual machine from the university that theoretically had much more processing power at hand. However, setting up the code here was difficult due to the windows operating system not supporting the installation of Windows-Subsystem-Linux (WSL). When we converted the original shell-scripts in the code base to windows compatible powershell-scripts, the next problem occurred. The NVIDIA Graphics card that was available to the virtual machine could not be accessed by the code and the python cuda library. After many attempts of utilizing the virtual machine, we then resorted to working with the small datasets from our personal machines.

4.4 Evaluation metrics

One of our main goals was to understand the results of the authors evaluation and the metrics they used. While we gained a general impression of the mathematical basics of the evaluation, we failed to understand the meaning of the outcome of our tests and its comparison to the results in the paper. Not only was it hard to follow how the score of the performance of a query type or one specific query was calculated but because of the general obscurity of the data, we could not find a way to display any of the data in a human-readable format.

Chapter 5

Related Work

In recent years, knowledge graph reasoning has become a significant area of research, especially focusing on complex query answering (CQA) involving both entities and numerical values. This section reviews relevant works in the area, highlighting advancements and challenges in knowledge graph reasoning.

5.1 Knowledge Graph Query Encoding Methods

Traditional query encoding methods, such as Graph Query Embedding (GQE) [17], Query2Box (Q2B) [18], and Query2Particles (Q2P) [19], encode queries as geometric shapes like vectors or hyper-rectangles. These methods often treat numerical values like entities, limiting their ability to handle queries requiring numerical reasoning. For example, they struggle with queries involving numerical comparisons or computations, such as finding entities with attributes greater than a certain value.

5.2 Numerical Reasoning in Knowledge Graphs

The Number Reasoning Network (NRN) [20] addresses the limitations of traditional methods by using distinct encoding structures for entities and numerical values. NRN applies parameterized density functions to represent numerical value distributions, enhancing numerical reasoning. This approach improves performance on numerical complex query answering (Numerical CQA) tasks across knowledge graphs like Freebase, YAGO, and DBpedia. NRN's accurate representation of numerical values enables more

precise query answering.

5.3 Applications and Challenges

Knowledge graph reasoning is used in fields like finance, healthcare, and e-commerce, where numerical reasoning is essential. Despite advancements such as NRN [20], challenges persist in accurately encoding numerical semantics and describing their distribution changes during computations and comparisons. Overcoming these challenges is crucial for enhancing the practical use of knowledge graphs in real-world applications.

5.4 Comparative Analysis

NRN consistently outperforms baseline models like GQE, Q2B, and Q2P in metrics such as Mean Reciprocal Rank (MRR) and Hit@K. It achieves this by using parameterized distributions for numerical encoding, which enhances its effectiveness for complex numerical queries.

The table below summarizes the performance improvements of NRN over baseline models.

Model	Improvement in MRR	Improvement in Hit@1	Improvement in Hit@10
GQE	+5%	+5%	+5%
Q2B	+6%	+5%	+4%
Q2P	+7%	+6%	+5%

Table 5.1: Summary of Performance Improvements of NRN

Traditional query encoding methods have laid the groundwork for knowledge graph reasoning. However, innovations like NRN [20] are essential for advancing the field, particularly in handling numerical values.

Chapter 6

Conclusion

6.1 Future Work

This section outlines potential areas for improvement in the current work, focusing on enhancing the transparency of the evaluation process and providing pretrained models to improve efficiency.

6.1.1 Fine-Grained Evaluation and Scoring Transparency

Based on the observed metrics, such as Mean Reciprocal Rank (MRR) and Hit@k from our experiments, a key area for improvement is enhancing the transparency and interpretability of the scoring process. Currently, there is a lack of clarity regarding the specific queries and answers used in scoring, which limits our understanding of how these metrics are derived. To address this, we propose developing a transparent framework that includes both the specific queries and corresponding answers used in the evaluation process, as well as the criteria for assigning evaluation scores. This approach will clarify how metrics like MRR and Hit@k are calculated, assist in interpreting the results, and provide a better understanding of the model's performance compared to baseline methods.

6.1.2 Availability of Pretrained Models

To improve efficiency and reduce the long training time of 60-70 hours, it is essential to provide pretrained models. Our investigation of the NRN model revealed that the lack of

pretrained models extended the training period significantly. Having access to pretrained models can speed up the process by using existing knowledge and features, allowing for faster experimentation and model refinement. For future improvements, we recommend ensuring the availability of pretrained models to make the training process more efficient.

6.2 Conclusion

The NRN model's capacity to process numerical values in knowledge graphs is still under evaluation and requires further testing to confirm its practical effectiveness. Despite existing challenges and limitations, NRN demonstrates potential for effective numerical query processing. This capability could benefit fields such as finance and scientific research. For example, NRN could enhance risk assessment models by accurately analyzing historical financial data. Moving forward, efforts will focus on improving the model's functionality, expanding its range of tasks, and integrating features such as interpretability and pretrained models to fully realize its practical applications.

Bibliography

[1] Jiaxin Bai, Chen Luo, zheng li, Qingyu Yin, Bing Yin, and Yangqiu Song. Knowledge graph reasoning over entities and numerical values. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23)*, pages 57–68, New York, NY, USA, 2023. ACM.

[2] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514, 2022.

[3] Alberto Garcia-Duran and Mathias Niepert. KBLRN : End-to-End Learning of Knowledge Base Representations with Latent, Relational, and Numerical Features, September 2017.

[4] Yangqiu Song, HKUST Department of CSE, Haoran Li, Dadi Guo, Wei Fan, Mingshi Xu, Jie Huang, Fanpu Meng, Yangqiu Song, Pat Verga, Haitian Sun, Livio Baldini Soares, William W. Cohen, Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, Ming-Wei Chang, Zihao Wang, Hang Yin, and Yangqiu Song. *Complex Query Answering on Neural Knowledge Graphs with Rich Semantics*. 2023.

[5] William L. Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. Embedding logical queries on knowledge graphs. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 2030–2041, Red Hook, NY, USA, 2018. Curran Associates Inc.

[6] Jiaxin Bai, Zihao Wang, Hongming Zhang, and Yangqiu Song. Query2particles: Knowledge graph reasoning with particle embeddings. pages 2703–2714, 01 2022.

[7] Hongyu Ren, Weihua Hu, and Jure Leskovec. Query2box: Reasoning over knowledge graphs in vector space using box embeddings, 02 2020.

[8] Amine Dadoun. Knowledge Graph Embeddings 101 - towards Data science. 5 2023.

[9] Wikipedia contributors. Word embedding, July 26 2024. Accessed: July 26, 2024.

[10] Nisha Arya. What are Vector Databases and Why Are They Important for LLMs? - KDnuggets.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[12] Wikipedia contributors. Normalization (statistics), 2 2024.

[13] Aishwarya Singh. Build Better and Accurate Clusters with Gaussian Mixture Models, May 5 2024. Accessed: July 28, 2024.

[14] Guojia Wan and Bo Du. GaussianPath:A Bayesian Multi-Hop Reasoning Framework for Knowledge Graph Reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):4393–4401, 5 2021.

[15] Prudhviraju Srivatsavaya. What is Positional Encoding? - Prudhviraju Srivatsavaya - Medium, July 26 2023. Accessed: July 28, 2024.

[16] Vinh Thinh Ho, Daria Stepanova, Dragan Milchevski, Jannik Strötgen, and Gerhard Weikum. Enhancing knowledge bases with quantity facts. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 893–901, New York, NY, USA, 2022. Association for Computing Machinery.

[17] William L. Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. Embedding logical queries on knowledge graphs. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018*, pages 2030–2041, Montréal, Canada, 2018. Curran Associates Inc.

[18] Hongyu Ren, Weihua Hu, and Jure Leskovec. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In *8th International Conference on Learning Representations (ICLR 2020)*, 2020.

[19] Jiaxin Bai, Zihao Wang, Hongming Zhang, and Yangqiu Song. Query2particles: Knowledge graph reasoning with particle embeddings. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2703–2714, 2022.

[20] Jiaxin Bai, Chen Luo, Zheng Li, Qingyu Yin, Bing Yin, and Yangqiu Song. Knowledge graph reasoning over entities and numerical values, 2023.