Knowledge Graph Embeddings in Function Spaces

Vikrant Singh

University of Paderborn
DICE Group

July 1, 2024

Overview

Introduction to Knowledge Graphs

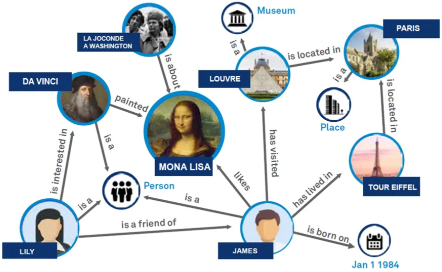


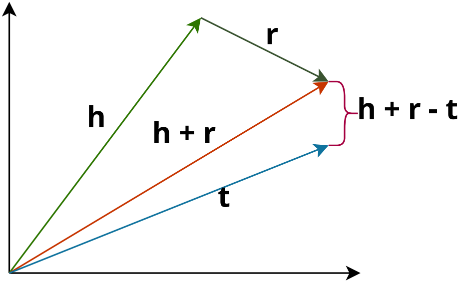
Figure: Knowledge Graph Example

- $\mathcal{G} := \{(h, r, t) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}\}$, where $\mathcal{E}$ and $\mathcal{R}$ represent a set of entities and relations respectively
- **Applications**: Enhance search engines, recommendation, and question-answering systems.

Knowledge Graph Embedding Models

- KGs are incomplete.
- KG Embedding models predict missing links, aiding in KG completion.
- Continuous vector representation for entities (h, t) and relations (r): $\mathbf{h} = [\theta_h]$, $\mathbf{r} = [\theta_r]$, $\mathbf{t} = [\theta_t]$.

Traditional Embedding Models



- TransE [1]: Models relationships as translations in the embedding space.
- Scoring function: $\psi_{TransE}(h, r, t) = \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_{l_1/l_2}$

Model Scoring Functions

Model	**Scoring Function** ψ
TransE [1]	$\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_{l_1/l_2}$
DistMult [2]	$\mathbf{h}^\top \mathrm{diag}(\mathbf{r})\mathbf{t}$
ComplEx [4]	$\mathrm{Re}(\mathbf{h}^\top \mathrm{diag}(\mathbf{r})\bar{\mathbf{t}})$

Table: Scoring functions of various KG embedding models.

Moving Beyond Vector Embeddings

- Function-based embeddings: Representing $(\mathbf{h}, \mathbf{r}, \mathbf{t})$ as functions $f(\theta, x)$.
- θ: Parameters for each triple component.
- Domain X: The function operates over a domain X.
- Goal: Utilize expressiveness of function spaces for more effective learning and optimization.
- Examples:
  - $\mathbf{h} = f(\theta_h, x)$
  - $\mathbf{r} = f(\theta_r, x)$
  - $\mathbf{t} = f(\theta_t, x)$

Process Pipeline

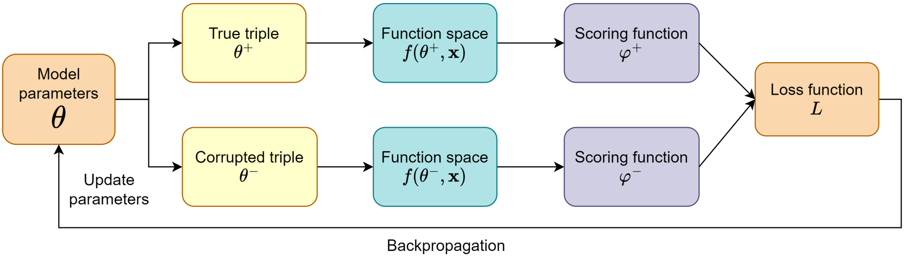


Figure: Schematic overview of the embedding as functions KG model.

Function Spaces Explored

- Polynomial Function Space.
- Complex Number Function Space.
- Neural Network Function Space.

Function Spaces Explored

- Polynomial Function Space
- Complex Number Function Space
- Neural Network Function Space $\Leftarrow\Leftarrow$ Neural Architecture Search.

Polynomial Function Space

Embedding as a Polynomial Functions

$$f_{\text{poly}}(\theta, x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_{d-1} x^{d-1}. \quad (1)$$

Example

- Consider head entity parameters $\theta_h = \{1, -2, 3\}$ over the domain X.

$$f_{\text{poly}}(\theta_h, x) = 1 - 2x + 3x^2$$

Complex Number Function Space

Embedding as a Complex Number Function

$$f_{complex}(\theta, x) = \sum_{k=0}^{d-1} \theta_k \cdot \cos(x) + i \cdot \sum_{k=0}^{d-1} \theta_k \cdot \sin(x), \tag{2}$$

Example

- Consider head entity parameters $\theta_h = \{2, -1\}$ over the domain X.

$$f_{complex}(\theta_h, x) = (2-1) \cdot \cos(x) + i \cdot (2-1) \cdot \sin(x)$$

Neural Network Function Space Equation

- Parameter set:

$$\theta = [\theta_1, \theta_2, \ldots, \theta_d], \quad \text{where } d \text{ is the dimension.} \tag{3}$$

- Network function space :

$$f(\theta, X) = \sigma(\theta^{(n)}(\sigma(\theta^{(n-1)}(\ldots\sigma(\theta^{(2)}(\sigma(\theta^{(1)}X))))))). \tag{4}$$

- Layer-specific matrix size:

$$\theta^n = \sqrt{\frac{d}{n}} \times \sqrt{\frac{d}{n}}, \quad \text{for } n \text{ layers.} \tag{5}$$

Optimized Network Configuration

How to find Optimal Values:

- Dimension (d), number of layers (n), and Activation function (σ)?
- Network architecture type?
- Other network hyperparameters?

Neural Architecture Search

Overview of Neural Architecture Search (NAS)

Figure: The general framework of Neural Architecture Search.

- NAS aims to automate the design of neural network architectures, optimizing for performance on a given task.

Overview of Neural Architecture Search (NAS)

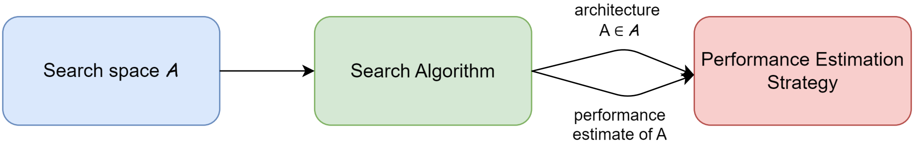


Figure: The general framework of Neural Architecture Search.

- NAS aims to automate the design of neural network architectures, optimizing for performance on a given task.
- **Search Space:** layer types, hyperparameters, and connectivity patterns.
- **Search Algorithm:** Uses techniques like reinforcement learning, Bayesian optimization, and evolutionary or gradient-based methods for efficient search.

NAS Frameworks

A variety of frameworks support NAS, each with unique features and methodologies:

- Microsoft's NNI (Neural Network Intelligence) [5].
- UniNAS [6].
- Auto-PyTorch [7].

NAS Frameworks

A variety of frameworks support NAS, each with unique features and methodologies:

- **Microsoft's NNI (Neural Network Intelligence)** [5].
- UniNAS [6].
- Auto-PyTorch [7].

NNI Visualization

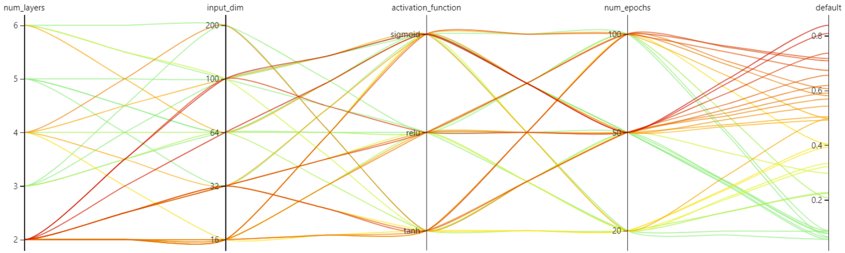
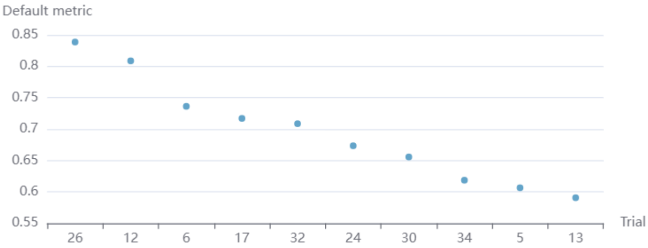


Figure: Parameters as the different axis in NNI

NNI Visualization

Top trials

Max | Min | Display top 10



	Trial No.	ID	Duration	Status	Default metric
>	26	klecB	1m 38s	SUCCEEDED	0.838603
>	12	C1axP	1m 21s	SUCCEEDED	0.808598
>	6	x4NIb	1m 58s	SUCCEEDED	0.736218
>	17	ex8X2	1m 41s	SUCCEEDED	0.717057

Optimal Network Configuration

$$f(\theta, X) = \sigma(\theta^{(n)}(\sigma(\theta^{(n-1)}(\ldots\sigma(\theta^{(2)}(\sigma(\theta^{(1)}X))))))). \tag{6}$$

The most effective configuration identified through NAS

- Two layers
- Layer normalization after each layer
- Residual connections
- Alternate relu activation function

Process Pipeline

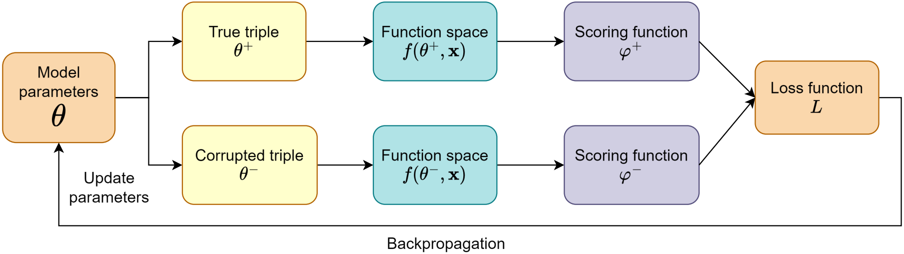


Figure: Schematic overview of the embedding as functions KG model.

Scoring Functions

- Compositional Scoring Function.
- Trilinear Scoring Function.
- Vector Triple Product Scoring Function.

Scoring Function

$$\psi_{compositional} = \int_X f(\theta_r, f(\theta_h, x)) \circ f(\theta_t, x)\, dx. \tag{7}$$

$$\psi_{trilinear} = \int_X f(\theta_h, x) \circ f(\theta_r, x) \circ f(\theta_t, x)\, dx. \tag{8}$$

$$\begin{aligned} \psi_{vtp} = &- \int_X f(\theta_h, x)\, dx \times \int_X f(\theta_t, x) \circ f(\theta_r, x)\, dx \\ &+ \int_X f(\theta_r, x)\, dx \times \int_X f(\theta_t, x) \circ f(\theta_h, x)\, dx. \end{aligned} \tag{9}$$

Scoring Function

$$\psi_{compositional} = \int_X f(\theta_r, f(\theta_h, x)) \circ f(\theta_t, x)\, dx. \tag{7}$$

$$\psi_{trilinear} = \int_X f(\theta_h, x) \circ f(\theta_r, x) \circ f(\theta_t, x)\, dx. \tag{8}$$

$$\begin{aligned} \psi_{vtp} = &- \int_X f(\theta_h, x)\, dx \times \int_X f(\theta_t, x) \circ f(\theta_r, x)\, dx \\ &+ \int_X f(\theta_r, x)\, dx \times \int_X f(\theta_t, x) \circ f(\theta_h, x)\, dx. \end{aligned} \tag{9}$$

Process Pipeline

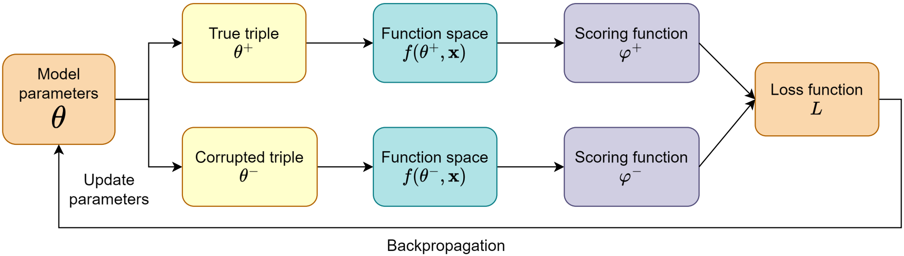


Figure: Schematic overview of the embedding as functions KG model.

Loss Functions Overview

- L2 Norm Loss.
- Margin Ranking Loss.
- Binary Cross-Entropy Logits Loss.

Loss Functions Overview

$$\mathcal{L}_{\text{l2}} = \sqrt{(1 - \psi(h, r, t) + \psi(h, r, \hat{t}))^2} \tag{10}$$

$$\mathcal{L}_{\text{margin}} = \max(0, \gamma - (\psi(h, r, t) - \psi(h, r, \hat{t}))) \tag{11}$$

$$\mathcal{L}_{\text{BCE}} = -y \cdot \log(\sigma(p)) - (1 - y) \cdot \log(1 - \sigma(p)) \tag{12}$$

Loss Functions Overview

$$\mathcal{L}_{\text{l2}} = \sqrt{(1 - \psi(h, r, t) + \psi(h, r, \hat{t}))^2} \tag{10}$$

$$\mathcal{L}_{\text{margin}} = \max(0, \gamma - (\psi(h, r, t) - \psi(h, r, \hat{t}))) \tag{11}$$

$$\mathcal{L}_{\text{BCE}} = -y \cdot \log(\sigma(p)) - (1 - y) \cdot \log(1 - \sigma(p)) \tag{12}$$

Loss Functions Overview

$$\mathcal{L}_{\mathrm{l2}} = \sqrt{(1 - \psi(h, r, t) + \psi(h, r, \hat{t}))^2} \tag{10}$$

$$\mathcal{L}_{\mathrm{margin}} = \max(0, \gamma - (\psi(h, r, t) - \psi(h, r, \hat{t}))) \tag{11}$$

$$\mathcal{L}_{\mathrm{BCE}} = -y \cdot \log(\sigma(p)) - (1 - y) \cdot \log(1 - \sigma(p)) \tag{12}$$

Result and Analysis

Evaluation Overview

Dataset	E	R	G^{Train}	$G^{\text{Val.}}$	G^{Test}
KINSHIP	104	25	8,544	1,068	1,074
UMLS	135	46	5,216	652	661
FB15K	14,951	1,345	483,142	50,000	59,071
WN18	40,943	18	141,442	5,000	5,000
NELL-995-h100	22,411	43	50,314	3,763	3,746
NELL-995-h50	34,667	86	72,767	5,440	5,393
NELL-995-h25	70,145	172	122,618	9,194	9,187

Dataset statistics.

Metrics Used:

- Mean Reciprocal Rank (MRR)
- Hits @n where $n = \{1, 3, 10\}$

UMLS Dataset Result

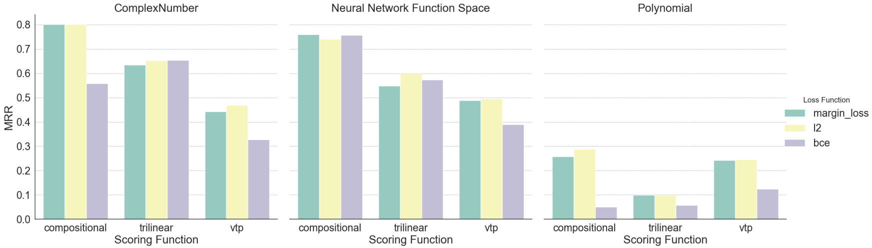


Figure: MRR performance of different function spaces with different loss and scoring functions for 64-dim vector on UMLS test dataset

KINSHIP Dataset Result

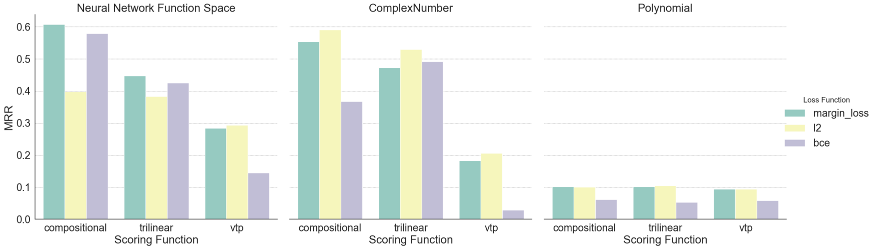


Figure: MRR performance of different function spaces with different loss and scoring functions for 64-dim vector on KINSHIP test dataset

Combined MRR Results for UMLS and KINSHIP Datasets

Model	MRR (Kinship)	MRR (UMLS)
TransE [1]	0.0545	0.6011
DistMult [2]	0.2299	0.1646
ComplEx [4]	0.1001	0.1142
QMult [3]	0.0670	0.1280
OMult [3]	0.0600	0.1043
Keci [8]	0.2193	0.1659
Complex Number FS	0.5236	0.6396
NNFS (NAS)	**0.7783**	**0.8798**

Table: MRR Comparison of Various KG Embedding Models Utilizing BCE Loss with a 1024-Dimensional Vector on the MLS and KINSHIP Test Dataset.

MRR for NELL-995 with h50 and h100

Model	MRR (h50)	MRR (h100)
TransE [1]	0.109	0.100
DistMult [2]	0.115	0.105
ComplEx [4]	0.066	0.044
QMult [3]	0.028	0.020
OMult [3]	0.035	0.019
Keci [8]	0.085	0.055
ComplexNumber FS	**0.172**	**0.158**
NNFS	0.099	0.130

Table: MRR Comparison of Various KG Embedding Models Utilizing BCE Loss with a 2048-Dimensional Vector on the NELL-995 Test Dataset

Combined MRR Results for WN18 and FB15K

Model	MRR (WN18)	MRR (FB15K)
TransE [1]	0.258	0.226
DistMult [2]	0.821	**0.479**
ComplEx [4]	**0.893**	0.466
QMult [3]	0.481	0.363
OMult [3]	0.637	0.380
Keci [8]	0.075	0.348
ComplexNumber FS	0.197	0.298
NNFS	0.826	0.360

Table: MRR Comparison of Various KG Embedding Models Utilizing BCE Loss with a 2048-Dimensional Vector on the WN18 and FB15K Test Dataset

Effect of embedding dimension

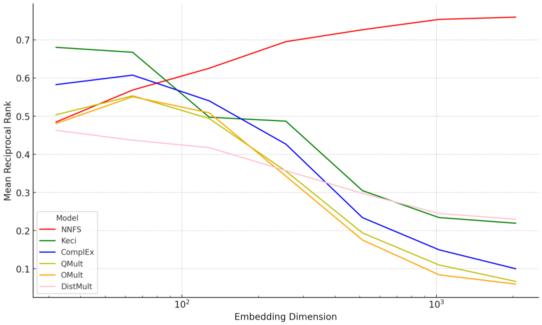


Figure: Comparison of Mean Reciprocal Rank for Various Models on the Kinship Test Dataset across Parameter Dimensions Ranging from 32 to 2048.

Effect of number of Negative Samples

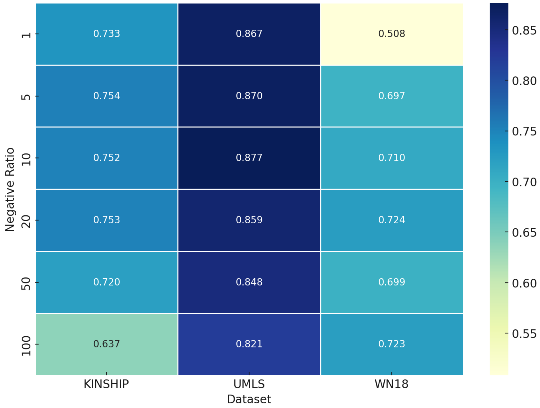


Figure: Heatmap reflecting the impact of varying negative ratios (k) on Mean Reciprocal Rank (MRR) scores across the KINSHIP, UMLS, and WN18 datasets.

Conclusions and Future Work

Conclusions:

- Compositional scoring function outperforms other.
- Different function spaces exhibit optimal performance with specific loss functions.
- NAS for automated discovery of optimal configurations.
- FS performs well at higher dimensions.
- Scalability issues on large datasets.

Future Directions:

- Enhance NNFS with NAS for scalability.
- Broaden scoring function research.
- Address large-scale dataset challenges.

The End

Questions? Comments?

Acknowledgements

Universität Paderborn

- Prof. Dr. Axel-Cyrille Ngonga Ngomo
- Jun.-Prof. Dr. Sebastian Peitz

References I

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.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng (2014).
Embedding Entities and Relations for Learning and Inference in Knowledge Bases.
International Conference on Learning Representations (ICLR).

Caglar Demir, Diego Moussallem, Stefan Heindorf, and Axel-Cyrille Ngonga Ngomo (2021).
Convolutional Hypercomplex Embeddings for Link Prediction.
Proceedings of the 13th Asian Conference on Machine Learning (ACML 2021).

References II

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard (2016).
Complex Embeddings for Simple Link Prediction.
International Conference on Machine Learning, 2071–2080.

Microsoft (2020).
Neural Network Intelligence.
https://github.com/microsoft/nni.

Kevin Alexander Laube (2020).
UniNAS.
GitHub repository, GitHub.
https://github.com/cogsys-tuebingen/uninas.

Lucas Zimmer, Marius Lindauer, and Frank Hutter (2021).
Auto-PyTorch Tabular: Multi-Fidelity MetaLearning for Efficient and Robust AutoDL.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 3079-3090.
Also available under https://arxiv.org/abs/2006.13799.

Caglar Demir and Axel-Cyrille Ngonga Ngomo (2023).
Clifford Embeddings – A Generalized Approach for Embedding in Normed Algebras.
Machine Learning and Knowledge Discovery in Databases: Research Track, Springer.
DOI: 10.1007/978-3-031-43418-1_34.
Available at
https://www.researchgate.net/publication/

References IV

373987609_Clifford_Embeddings_-_A_Generalized_
Approach_for_Embedding_in_Normed_Algebras.