

Knowledge Graph Completion Using Deep Learning

Aditi Sharma, Omprakash Yadav

Students, Associate Professor

Department of Knowledge-Based Systems

PSGR Krishnammal College for Women, Coimbatore

Email: *Aditi25sharma@yahoo.com, yadavomprakash72@gmail.com*

Abstract

Knowledge graphs (KGs) are structured representations of entities and their relationships, widely used in domains like natural language processing, recommendation systems, and semantic search. Despite their utility, knowledge graphs are often incomplete, containing missing entities or relations, which can reduce their effectiveness. Knowledge graph completion (KGC) aims to infer these missing links and enhance the graph's coverage. Recently, deep learning approaches have shown significant improvements in KGC by effectively modeling complex patterns and semantic information within large-scale graphs. This paper provides a comprehensive review of KGC methods leveraging deep learning, including embedding-based models, graph neural networks (GNNs), and hybrid architectures. Comparative analyses, challenges, and future research directions are discussed. The paper also presents key datasets, evaluation metrics, and illustrative results through tables and figures to highlight trends and performance in KGC research.

Keywords: *Knowledge Graph Completion, Deep Learning, Graph Neural Networks, Embedding Models, Link Prediction, Semantic Knowledge Graphs*

INTRODUCTION

Knowledge graphs (KGs) represent real-world entities and relationships in structured graph form, enabling machines to reason, answer queries, and provide intelligent recommendations. Examples of popular knowledge graphs include **DBpedia**, **Wikidata**, and **Freebase**, which are extensively used in AI applications. Despite their growth, KGs suffer from incompleteness due

to errors in extraction, data sparsity, or rapidly evolving domains. Incomplete KGs hinder downstream tasks like question answering, recommendation systems, and semantic search.

Knowledge graph completion (KGC) addresses this issue by predicting missing entities or relations to produce a more comprehensive graph. Traditional KGC approaches relied on rule-based or statistical methods; however, these approaches often fail to capture complex dependencies and large-scale semantics. Deep learning has emerged as a powerful tool in KGC because it can learn latent representations of entities and relations, enabling high-accuracy predictions even in sparse or noisy data scenarios.

This paper aims to review the state-of-the-art deep learning techniques for KGC, classify methods based on architectures, analyze performance on benchmark datasets, and discuss challenges and future research directions.

FUNDAMENTALS OF KNOWLEDGE GRAPHS

Knowledge graphs (KGs) are structured representations of real-world knowledge, capturing entities, their attributes, and relationships in a formal, machine-readable format. They are widely used in applications such as semantic search, recommendation systems, natural language understanding, and question answering. By organizing information as a graph, KGs allow both humans and machines to reason over entities and their interconnections.

1. Structure of Knowledge Graphs

A knowledge graph can be formally represented as:

$$G=(E,R,T) \quad G=(E, R, T) \quad G=(E,R,T)$$

Where:

- **EEE (Entities):** Entities are the nodes of the graph, representing objects, concepts, or instances in the real world. Examples include *Albert Einstein*, *Physics*, *Princeton University*. Each entity can have attributes or types, such as *Person*, *Field*, or *Organization*.
- **RRR (Relations):** Relations represent the edges in the graph and define the type of connection between entities. Relations are typically directional and labeled, indicating the nature of the relationship. Examples include *educated at*, *field of study*, *born in*.

- **TTT (Triples):** The set of triples $T \subseteq E \times R \times E$ represents factual knowledge in the KG. Each triple has a **head entity**, a **relation**, and a **tail entity**, forming a structured fact. For example:

(Albert Einstein, field of study, Physics)

Here, *Albert Einstein* is the head entity, *field of study* is the relation, and *Physics* is the tail entity.

Key characteristics of knowledge graphs:

1. **Heterogeneous Structure:** Entities can belong to different types (e.g., people, locations, events), and relations may be diverse and multi-typed.
2. **Directed and Labeled Edges:** Relations often have direction (from head to tail) and carry semantic meaning, allowing precise modeling of real-world interactions.
3. **Multi-relational Graphs:** Unlike traditional networks, entities can be connected through multiple relations, forming a rich, multi-dimensional structure.
4. **Semantic Context:** KGs preserve contextual meaning by linking related entities, enabling reasoning across the graph.

Example:

Head Entity	Relation	Tail Entity
Albert Einstein	born in	Ulm
Albert Einstein	educated at	ETH Zurich
Albert Einstein	field of study	Physics
Physics	part of	Natural Science

This table shows how a knowledge graph encodes structured facts that can be queried or used for inference.

2. Knowledge Graph Incompleteness

Despite the benefits, most real-world knowledge graphs are incomplete due to limitations in data collection, extraction errors, or dynamic knowledge domains. The incompleteness manifests in several ways:

1. **Missing Entity Links:** Relationships that should exist between known entities are absent. For example, a KG might include *Albert Einstein* and *Physics*, but the triple (*Albert Einstein, field of study, Physics*) may be missing.
2. **Missing Entities:** Some entities may not be included at all, either because they are newly emerging concepts or were overlooked during KG construction. For example, a newly discovered particle in physics may not yet be represented in the graph.
3. **Noisy or Incorrect Data:** Some triples may contain incorrect or contradictory information due to errors in extraction or integration from multiple sources. For example, a triple stating (*Albert Einstein, born in, Berlin*) is factually incorrect.
4. **Sparse Relations:** Certain types of relations may be underrepresented, making it difficult for models to learn their patterns.

These issues reduce the utility of knowledge graphs in downstream applications such as recommendation, search, or question answering.

Knowledge Graph Completion (KGC) is the task aimed at addressing these challenges. Formally, KGC seeks to predict missing triples (h, r, t) , where h is a head entity, r is a relation, and t is a tail entity, such that the predicted triple is likely to be true. This process involves:

- **Link Prediction:** Inferring missing edges between existing entities.
- **Entity Prediction:** Identifying previously missing entities and linking them correctly.
- **Relation Prediction:** Discovering the appropriate relation type between entities when ambiguous.

By completing the graph, KGC enhances the richness, consistency, and usability of knowledge graphs, enabling more accurate reasoning and intelligent application outcomes.

.

DEEP LEARNING FOR KNOWLEDGE GRAPH COMPLETION

Deep learning models for KGC can be broadly classified into three categories: embedding-based models, graph neural networks, and hybrid approaches.

EMBEDDING-BASED MODELS

Embedding-based models represent entities and relations as dense vectors in a continuous latent space, capturing semantic and relational patterns.

1. Translational Models

TransE is a foundational translational embedding model. It represents relationships as translations in vector space:

$$\mathbf{h} + \mathbf{r} \approx \mathbf{t}$$

Where $\mathbf{h}, \mathbf{r}, \mathbf{t}$ are embeddings for head, relation, and tail.

Variants include **TransH**, **TransR**, and **TransD**, which handle complex relations such as one-to-many or many-to-many.

2. Semantic Matching Models

Semantic matching models use similarity-based scoring functions between entity and relation embeddings.

- **DistMult**: Bilinear scoring: $f(\mathbf{h}, \mathbf{r}, \mathbf{t}) = \mathbf{h}^T \text{diag}(\mathbf{r}) \mathbf{t}$
- **Complex**: Extends DistMult to complex space to handle asymmetric relations.

3. Neural Tensor Networks (NTN)

NTNs model multiplicative interactions between entity embeddings, enabling high-order relational representations.

Table 1. shows the comparison of popular embedding-based models.

Model	Approach	Pros	Cons	Example Use Cases
TransE	Translational	Simple, efficient	Struggles with complex relations	Link prediction
TransH	Translational	Handles one-to-many/many-to-many	Slightly more complex	Knowledge graph refinement
DistMult	Semantic	Simple bilinear scoring	Symmetric relations only	Recommendation systems
Complex	Semantic	Handles asymmetry	Complex space computations	Question answering

Model	Approach	Pros	Cons	Example Use Cases
NTN	Neural	Captures high-order patterns	High computational cost	Advanced link prediction

GRAPH NEURAL NETWORKS (GNNS)

GNNS leverage graph topology and feature information to generate node embeddings, which are used for KGC tasks.

1. Graph Convolutional Networks (GCNs)

GCNs propagate information between neighboring nodes to encode structural and semantic context:

$$H^{(l+1)} = \sigma(D^{-1/2} A D^{-1/2} H^{(l)} W^{(l)})$$

Where A is the adjacency matrix with self-loops, D is the degree matrix, $H^{(l)}$ the node embeddings at layer l , and $W^{(l)}$ the learnable weights.

2. Relational Graph Convolutional Networks (R-GCNs)

R-GCNs extend GCNs to handle multiple types of relations in KGs:

$$h_i^{(l+1)} = \sigma \left(\sum_{r \in R} \sum_{j \in N_i^r} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)} + W_0^{(l)} h_i^{(l)} \right)$$

Where N_i^r is the set of neighbors of node i under relation r , and $c_{i,r}$ is a normalization constant.

3. Graph Attention Networks (GATs)

GATs assign attention weights to neighbors, allowing the model to prioritize more relevant relational information:

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(a^T [W h_i || W h_j]))}{\sum_{k \in N_i} \exp(\text{LeakyReLU}(a^T [W h_i || W h_k]))}$$

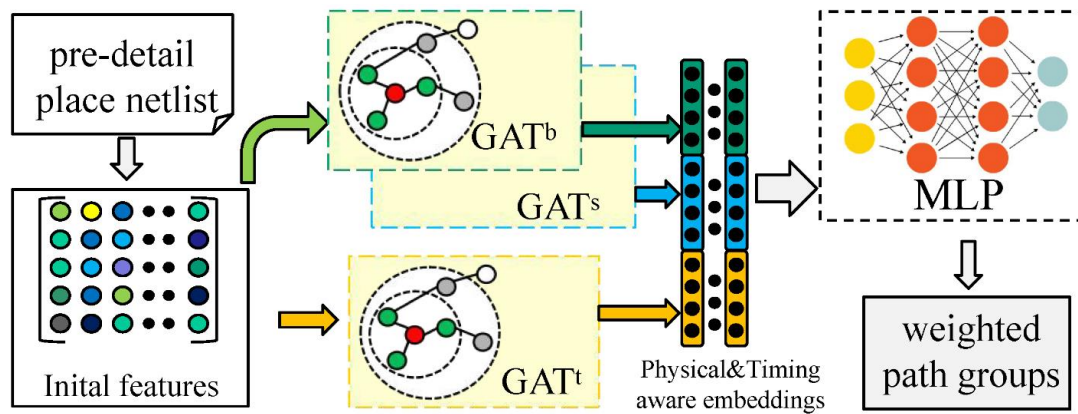


Figure 1. illustrates a GNN-based KGC framework.

HYBRID MODELS

Hybrid models combine embedding and graph neural network approaches for improved KGC performance.

- **ConvE**: Applies convolutional layers to 2D reshaped embeddings for interaction modeling.
- **CompGCN**: Combines relation and entity embeddings in a GCN framework to improve relational reasoning.
- **STaGCN**: Integrates semantic information with graph convolutions for knowledge inference.

DATASETS AND EVALUATION METRICS

For research in knowledge graph completion (KGC), having standardized datasets and evaluation metrics is essential to benchmark model performance and enable fair comparisons. Knowledge graphs in practice vary widely in size, domain, and complexity, which affects the selection of datasets and the design of experiments.

1. Benchmark Datasets

Several datasets have become standard benchmarks for evaluating KGC models. They differ in scale, entity/relation types, and domain focus. These datasets provide pre-defined train, validation, and test splits to allow reproducibility of experiments.

Table 2. Summarizes popular KGC datasets.

Dataset	Domain	Triples	Entities	Relations
FB15k-237	General	310k	15k	237
WN18RR	Lexical/NLP	93k	40k	11
YAGO3-10	General/Entities	1.2M	123k	37
DBpedia	General	5M	500k	500

EVALUATION METRICS

Evaluation metrics are critical for assessing the performance of knowledge graph completion (KGC) models. They quantify how accurately a model can predict missing triples, rank the correct entity, or infer relations in a knowledge graph. Common metrics include **Mean Reciprocal Rank (MRR)**, **Hits@k**, and **Mean Rank (MR)**.

1. Mean Reciprocal Rank (MRR)

Definition:

MRR measures the average of the reciprocal ranks of the correct entities in a ranked list of predictions. For each test triple (h, r, t) , the model predicts a ranked list of candidate tail entities (or head entities) based on a scoring function. The reciprocal rank is the inverse of the position at which the correct entity appears in the ranked list.

Formula:

$$MRR = \frac{1}{|T_{test}|} \sum_{i=1}^{|T_{test}|} \frac{1}{rank_i}$$

Where:

- $|T_{test}|$ is the number of test triples.
- $rank_i$ is the rank of the correct entity for the i -th triple.

Interpretation:

- A higher MRR indicates that the correct entity is ranked closer to the top for most predictions.
- MRR ranges from 0 to 1, with 1 being perfect ranking (the correct entity is always ranked first).

Example:

Suppose the model predicts the following ranked tail entities for the triple (AlbertEinstein,fieldofstudy,?)(Albert Einstein, field of study, ?)(AlbertEinstein,fieldofstudy,?):

- Mathematics
- Physics ✓
- Chemistry

The correct tail entity (*Physics*) is at **rank 2**, so the reciprocal rank is $1/2=0.51/2 = 0.51/2=0.5$. If another triple had the correct entity at rank 1, its reciprocal rank would be 1. The MRR is the average of these reciprocal ranks across all test triples.

2. Hits@k

Definition:

Hits@k measures the fraction of correct entities that appear in the top-k predictions generated by the model. It evaluates how often the correct entity is among the top-ranked candidates.

Formula:

$$\text{Hits@k} = \frac{1}{|T_{\text{test}}|} \sum_{i=1}^{|T_{\text{test}}|} \mathbf{1}(\text{rank}_i \leq k)$$

Where:

- $\mathbf{1}(\text{rank}_i \leq k)$ is an indicator function equal to 1 if the correct entity is within the top-k predictions, otherwise 0.
- $|T_{\text{test}}|$ is the total number of test triples.

Interpretation:

- A higher Hits@k indicates better predictive performance.
- Typical values used are k=1,3,10k = 1, 3, 10k=1,3,10, e.g., Hits@10 indicates the percentage of test triples where the correct entity is in the top 10 predictions.

Example:

For three test triples:

Test Triple	Correct Entity Rank	Hits@1	Hits@3	Hits@10
(A, r, ?)	2	0	1	1
(B, r, ?)	1	1	1	1

Test Triple	Correct Entity Rank	Hits@1	Hits@3	Hits@10
(C, r, ?)	5	0	0	1

$$\text{Hits@1} = \frac{0+1+0}{3} = 0.33$$

$$\text{Hits@3} = \frac{1+1+0}{3} = 0.66$$

$$\text{Hits@10} = \frac{1+1+1}{3} = 1.0$$

This shows the model often ranks the correct entity within the top 3 or 10 predictions.

3. Mean Rank (MR)

Definition:

Mean Rank computes the average rank of the correct entities across all test triples. It measures how far, on average, the model places the correct entity in its predictions.

Formula:

$$\text{MR} = \frac{1}{|T_{\text{test}}|} \sum_{i=1}^{|T_{\text{test}}|} \text{rank}_i$$

Interpretation:

- Lower MR values indicate better performance, as correct entities are ranked closer to the top.
- Unlike MRR, MR is sensitive to extreme ranks and outliers.

Example:

For three test triples with ranks 2, 1, and 5:

$$\text{MR} = \frac{2+1+5}{3} = 2.67$$

This means the correct entity is, on average, ranked around the 3rd position.

4. Additional Notes

- **Filtered vs Raw Evaluation:** In practice, the ranking is often "filtered" to remove other valid triples from consideration, preventing penalization for predicting an alternative correct entity.
- **Complementary Metrics:** MRR provides a robust average ranking measure, Hits@k evaluates practical top-k performance, and MR gives insight into absolute ranking positions.

Summary Table:

Metric	Definition	Interpretation	Desirable Value
MRR	Mean reciprocal rank	Measures average top-ranked accuracy	High (closer to 1)
Hits@k	Fraction of correct entities in top-k	Measures top-k retrieval accuracy	High (closer to 1)
MR	Mean rank of correct entities	Average position of correct entity	Low (closer to 1)

These metrics are widely used to benchmark embedding-based, GNN-based, and hybrid KGC models across standard datasets like FB15k-237, WN18RR, and YAGO3-10.

COMPARATIVE ANALYSIS

Recent studies show that GNN-based methods outperform traditional embedding approaches on datasets with complex relational patterns. **Table 3** presents recent results for MRR and Hits@10.

Table 3.

Model	FB15k-237 (MRR)	WN18RR (MRR)	Hits@10 FB15k-237	Hits@10 WN18RR
TransE	0.294	0.226	44.1	48.2
DistMult	0.338	0.430	49.2	53.1
ComplEx	0.339	0.440	50.1	54.7
ConvE	0.325	0.430	51.0	52.5
R-GCN	0.349	0.435	52.3	55.0
CompGCN	0.362	0.447	54.0	56.2

The table indicates that hybrid and GNN-based models excel in relational reasoning, especially in multi-hop and sparse scenarios.

CHALLENGES IN KNOWLEDGE GRAPH COMPLETION

Despite advances, KGC using deep learning faces several challenges:

- **Scalability:** Large-scale KGs with millions of entities are computationally intensive.
- **Dynamic Graphs:** Real-world KGs evolve over time, requiring incremental learning.
- **Explainability:** Deep models lack transparency in predictions.
- **Data Sparsity:** Some relations are underrepresented, leading to biased embeddings.
- **Heterogeneous Data Integration:** Integrating multimodal data (text, images) remains challenging.

FUTURE DIRECTIONS

1. **Self-Supervised Learning:** Pretraining KG embeddings with self-supervised objectives to improve generalization.
2. **Multimodal KGs:** Incorporating images, text, and structured data for richer representations.
3. **Temporal Graphs:** Modeling temporal dynamics for time-aware link prediction.
4. **Explainable KGC:** Developing interpretable models for industry adoption.
5. **Edge and Node Augmentation:** Leveraging knowledge transfer and data augmentation to combat sparsity.

CONCLUSION

Knowledge graph completion is vital for enhancing KG quality, enabling improved reasoning, search, and recommendation capabilities. Deep learning methods, particularly GNN-based and hybrid models, have substantially improved link prediction and triple inference over traditional methods. Despite the significant progress, challenges like scalability, dynamic updates, and explainability remain. Future research is expected to focus on multimodal integration, temporal reasoning, and self-supervised approaches to build more robust and interpretable KGC systems.

REFERENCES

1. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., & Yakhnenko, O. (2013). Translating Embeddings for Modeling Multi-relational Data. *NeurIPS*.
2. Wang, Z., Zhang, J., Feng, J., & Chen, Z. (2014). Knowledge Graph Embedding by Translating on Hyperplanes. *AAAI*.

3. Nickel, M., Tresp, V., & Kriegel, H. P. (2011). A Three-Way Model for Collective Learning on Multi-Relational Data. *ICML*.
4. Schlichtkrull, M., Kipf, T. N., Bloem, P., van den Berg, R., Titov, I., & Welling, M. (2018). Modeling Relational Data with Graph Convolutional Networks. *ESWC*.
5. Dettmers, T., Minervini, P., Stenetorp, P., & Riedel, S. (2018). Convolutional 2D Knowledge Graph Embeddings. *AAAI*.
6. Vashishth, S., Sanyal, S., Nitin, V., Talukdar, P. (2020). Composition-based Multi-Relational Graph Convolutional Networks. *ICLR*.
7. Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE TKDE*, 29(12), 2724-2743.
8. Wang, Q., Mao, Z., Wang, B., & Guo, L. (2017). Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE TKDE*, 29(12), 2724–2743.
9. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A Comprehensive Survey on Graph Neural Networks. *IEEE TNNLS*, 32(1), 4-24.
10. Xie, R., Liu, Z., Jia, J., & Sun, M. (2016). Representation Learning of Knowledge Graphs with Entity Descriptions. *AAAI*.