Link Predictions using Graph Embeddings

Ahmad Ahmad, Rishika A. Gupta
Erasmus Mundus Joint Master's Degree in Big Data Management and Analytics
Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya, Barcelona, Spain
ahmad.ahmad@estudiantat.upc.edu, rishika.gupta@estudiantat.upc.edu

***Abstract*—Knowledge Graphs (KGs) have gained significant attention in both industry and academia, leading to extensive research on extracting information from diverse sources. However, even the most advanced KGs suffer from incompleteness, which has prompted research efforts in the field of Link Prediction (LP). LP aims to predict missing facts within a KG and has emerged as a promising task to address KG incompleteness. KG embedding-based mainly - Geometric, Matrix Factorization and Deep Learning techniques have shown promising performance in LP benchmarks. In this analysis, we present a comprehensive understanding of 6 state-of-the-art LP methods namely - Geometric: TransE & RotatE, Matrix Factorization: DistMult & TuckER, Deep Learning: ConvE & InteractE. Furthermore, these models are compared on common large datasets - Freebase FBk-257 and WordNet WN18RR using the evaluation metrics of Mean Reciprocal Ratio (MRR) and Hits@10 (H@10). Results show that both RotatE and TuckER has proven to perform the best in terms of inferring these missing links.**

***Index Terms*—Graph Embeddings, Link Prediction, Knowledge Graphs**

I. Introduction

A. *Why Graph Embeddings?*

Graph networks are essentially frameworks used for representing and manipulating complex data and its relationships in the form of graphs. In current times, with the growing availability of large-scale graph-structured data, graph networks have become essential for understanding complex systems and extracting valuable insights. In social networks, graphs can represent social connections between people, enabling analysis of communities, influence spread and recommendation systems. At the same time, they can also model molecular structures, protein-protein interactions, gene regulatory networks, and metabolic pathways, aiding in drug discovery and understanding of biological systems. Their ability to capture relationships and dependencies within the data makes them valuable for tasks such as data integration, clustering, link prediction, anomaly detection, and visualization [1].

However, why do we need graph embeddings? One of the primary reasons is to enhance machine learning on graphs [2]. Traditional machine learning techniques are basically designed for vector data (mathematical or textual), limiting their direct application to graph-structured data. Graph embeddings transform the discrete and complex nature of graphs into continuous vector representations, allowing the utilization of a broader range of mathematical, statistical, and machine-learning approaches. This enables the application of powerful machine learning algorithms on graph data for tasks like node classification, link prediction, and recommendation systems. Secondly, embeddings are an efficient way of representing graph data. Graphs can be large and complex, with a high number of nodes and edges [2]. Storing and processing graphs directly using their adjacency matrix can be computationally expensive and memory-intensive. Graph embeddings provide a compressed representation of graph data by mapping nodes and edges to lower-dimensional vector spaces. These embeddings capture the structural and semantic information of the graph while reducing the memory footprint and computational complexity. And finally, they also aid in performing simple and faster computations. Vector operations on embeddings are more efficient compared to comparable operations on graphs. Once nodes and edges are embedded into continuous vector representations, various mathematical operations, such as distance calculations, similarity measurements, and transformations, can be performed more efficiently. This allows for faster computations and enables scalable analysis of large graphs. Thus, by leveraging graph embeddings, we can transform the discrete and complex nature of graphs into a continuous vector space, where mathematical operations can be applied and patterns can be discovered more easily. Further, by embedding graph elements into a continuous space, graph structures can be analyzed and compared using traditional machine learning and data mining techniques [2].

B. *Why Link Prediction?*

Knowledge graphs are large-scale repositories of structured as well as unstructured data that capture entities, their attributes and the relationships between them. However, even in comprehensive knowledge graphs, the available information is often incomplete and there are missing or undiscovered relationships between entities. A very simplistic use-case of Link Prediction is shown in the figure 1 and the following points detail why link prediction commonly occurs in the knowledge graph.

- **Data Incompleteness:** Knowledge graphs are constructed from various sources, including curated databases, text extraction, and user-generated content. Despite efforts to create comprehensive knowledge graphs, the information available is inherently incomplete. There may be entities

that are not yet included, and relationships between existing entities may be missing. Link prediction helps infer these missing connections and provides a more comprehensive view of the knowledge graph [3].

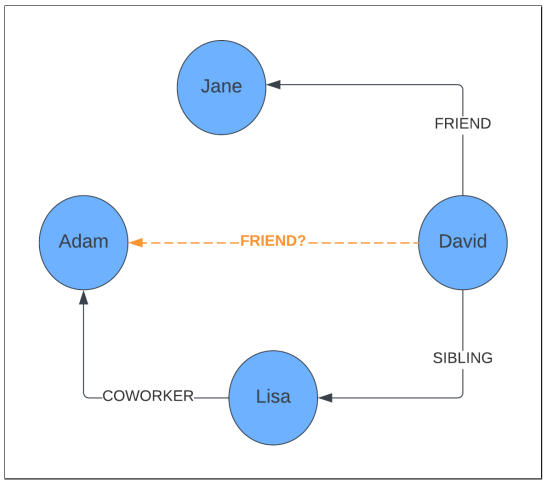

Fig. 1: Example of Link Prediction

- **Semantic Gap:** Knowledge graphs aim to represent real-world knowledge and capture complex relationships between entities. However, the process of constructing a knowledge graph involves significant semantic interpretation, mapping, and integration of diverse data sources. As a result, there can be a semantic gap between the available information and the true underlying relationships. Link prediction helps bridge this gap by inferring relationships that might not have been explicitly captured in the knowledge graph [4].

- **Knowledge Discovery:** Knowledge graphs are valuable resources for knowledge discovery and exploration. Link prediction plays a critical role in uncovering hidden or latent relationships between entities in the knowledge graph. It helps in identifying potential connections that were not previously known, supporting tasks such as completing the knowledge graph, discovering new facts, and supporting reasoning and inference [5].

- **Knowledge Graph Evolution:** Knowledge graphs are dynamic, evolving structures that are continuously updated with new information. As new data becomes available, link prediction assists in predicting future links and relationships in the evolving knowledge graph. This is especially important for maintaining up-to-date knowledge graphs and supporting applications that rely on the most current information [6].

Due to these reasons, the link prediction problem is particularly relevant yet challenging in the context of knowledge graphs. Thus, by addressing the link prediction problem in KGs, we can enhance the completeness, accuracy, and usability of these knowledge repositories. Link prediction enables knowledge discovery, supports reasoning and inference, facilitates data integration, and assists in maintaining up-to-date and high-quality knowledge graphs.

In social networks, link prediction can be used to suggest new friendships or connections between individuals in social networks. It can help recommend potential friends, identify community structures, detect hidden relationships, and enhance social network analysis [7]. On the other hand in the case of recommendation systems, it can improve recommendation algorithms by predicting links between users and items. By estimating the likelihood of a user liking or purchasing a particular item, personalized recommendations can be made. This is valuable in e-commerce, movie recommendations, music streaming platforms, and social media platforms [8]. Continuing on the use case of building recommendations, it can also be applied in collaborative filtering scenarios to predict missing or future interactions between users and items, aiding in suggesting relevant items based on user preferences and item similarities [8], [9]. In the domain of biology and bioinformatics, it is very useful in predicting protein-protein interactions [10], gene-disease associations, and metabolic pathways, as by inferring potential links between biological entities, link prediction can assist in understanding molecular interactions, drug discovery, and disease analysis [11]. Additionally, for information retrieval systems and search engines, it suggests related documents, web pages, or resources that are not explicitly linked and thus assists in discovering hidden relationships, thereby improving the quality of search results. Predicting future citations between scientific papers, link prediction proves its importance in citation networks by identifying influential papers, understanding research trends, and recommending related publications [12]. Not limited to this, in the sector of cybersecurity, it can identify potential fraudsters or malicious entities by detecting abnormal link patterns and predicting suspicious connections in networks [13]. Summing up, the essence of link prediction includes predicting relationships between entities in knowledge graphs to complete knowledge graphs, identify missing links, and enhance semantic search and knowledge discovery.

II. Graph Embedding Approaches for Link Prediction

In this section, we will discuss the commonly used datasets (Section A), evaluation metrics (Section B) as well as summary of methods employed (Section C onwards) to convert graphs into embeddings for tackling the link prediction problem.

A. *Datasets*

Typically, benchmark datasets for link prediction are derived by sampling real-world knowledge graphs (KGs) and subsequently dividing the collected facts into training, validation, and test sets. In our analysis, we outline the five most widely recognized datasets in the field of link prediction

[14]. Table I presents key properties of these datasets that are of particular importance.

Dataset	Entities	Relations	Train	Valid	Test
FB15k	14951	1345	483142	50000	50971
WN18	40943	18	141442	5000	5000
FB15k-237	14541	237	272115	17535	20466
WN18RR	40943	11	86835	3034	3134
YAGO3-10	123182	37	1079040	5000	5000

TABLE I: Link Prediction Datasets Summary [14]

Freebase (FB15k): Among the available benchmark datasets, FB15k stands out as one of the most widely utilized. In its creation, the authors of the dataset [15] specifically selected FreeBase entities that had over 100 mentions and were present in the Wikilinks database. They extracted all the associated facts involving these entities, including their lower-degree neighbours, while excluding facts with literal such as dates and proper nouns. Additionally, they transformed n-ary relations represented with reification into cliques of binary edges, significantly impacting both the graph structure and semantics.

WordNet (WN18): WN18 which was also introduced by the authors of TransE [15], originates from WordNet3, a linguistic KG ontology designed to serve as a dictionary and thesaurus for NLP and automatic text analysis purposes. In WordNet, entities correspond to synsets or word senses, and relations represent their lexical connections, such as hypernym. To construct WN18, the authors utilized WordNet as a foundation and progressively filtered out entities and relationships with insufficient occurrences through iterative refinement.

Freebase (FB15k-237): FB15k-237 is a modified version of FB15k created by Toutanova and Chen [16], addressing the issue of test leakage observed in FB15k. Test leakage occurs when the test data is exposed to models during the training phase. In the case of FB15k, this problem arises due to the presence of closely related or inverse relations. To evaluate the extent of this issue, Toutanova and Chen demonstrated that a simple model based on observable features could achieve impressive performance on FB15k. To provide a more challenging dataset, FB15k-237 was developed. The authors selected facts from FB15k that involved the 401 largest relations and removed any equivalent or inverse relations. Additionally, to eliminate trivial triples, they ensured that entities connected in the training set were not directly linked in the validation and test sets.

WordNet (WN18RR): WN18RR is a modified version of WN18 developed by Dettmers et al. [17] to address the issue of test leakage observed in WN18. The authors highlight the severity of this leakage by demonstrating that a simple rule-based model, known as the Inverse Model, achieves state-of-the-art performance on both WN18 and FB15k by detecting inverse relations. To mitigate this problem, they construct the more challenging WN18RR dataset using a similar approach employed for FB15k-237 [16].

YAGO3 KG (YAGO3-10): YAGO3-10, derived from the YAGO3 KG [36], was introduced by Dettmers et al. [17]. It was generated by selecting entities that have a minimum of 10 different relations and collecting all the associated facts, including their neighboring entities. Unlike FB15k and FB15k-237, YAGO3-10 also retains facts related to textual attributes found in the KG. Consequently, as stated by the authors, a significant portion of the triples in YAGO3-10 pertains to descriptive properties of individuals, such as citizenship or gender. The relatively poor performance of the Inverse Model [17] on YAGO3-10 suggests that this benchmark does not suffer from the same test leakage issues observed in FB15k and WN18.

For this work, we have considered two datasets as benchmarks - FB15k-237 and WN18RR (avoiding test leakage datasets).

B. Evaluation Metrics

The most frequently used metrics to evaluate the test scores in Link Prediction stated in [14] are discussed below. Note that these metrics can be calculated either individually for subsets of predictions or collectively by considering all test predictions as a whole.

Mean Rank (MR): The score as described in eq.1 has a value between 1 and $|N|$, where N are the set of nodes representing entities in the graph. The MR metric represents the average rank position of correct predictions. Lower values of MR indicate better performance, as it means that the correct links are ranked higher on average. Due to its

$$MR = \frac{1}{|Q|} \sum_{q \in Q} q \tag{1}$$

sensitivity to outliers, researchers have recently opted to avoid using Mean Rank (MR) as a metric in link prediction. Instead, they have shifted towards employing Mean Reciprocal Rank (MRR) as a more robust alternative. The value lies between 0 and 1, and the higher the better.

Mean Reciprocal Rank (MRR): It is defined as the average of the inverse of the obtained ranks as displayed in eq.2.

$$MRR = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{q} \tag{2}$$

Hits@K (H@K): This metric H@K - eq.3 represents the ratio of predictions with a rank equal to or lower than a given threshold K. Common values for K include 1, 3, 5, and 10. Hits@K ranges from 0 to 1, with a higher value indicating better performance. For example, Hits@1 measures

the accuracy of the top-ranked prediction, while Hits@5 measures the accuracy of the top five predict.ions H@1

$$H@K = \frac{|q \in Q : q \leq K|}{|Q|} \tag{3}$$

and Mean Reciprocal Rank (MRR) are often closely related because these predictions correspond to the most relevant components in the MRR formula.

In this paper, we have considered two evaluation metrics as benchmarks - MRR and Hits@10.

C. *Approaches*

The three major approaches implemented to solving the Link Prediction problem are - Geometric (Section D), Matrix-Factorization (Section E) and Deep Learning (Section F). Also, the models under each of these approaches are summarized in the following sections as well as in Figure 2.

1) Geometric: In this type of approach, relations are understood as geometric operations performed in the latent space. The head embedding undergoes a spatial transformation τ, which is determined by the relation embedding's values. The resulting vector is then compared to the tail vector using a distance function, δ (such as L1 or L2 norm), to compute the fact score [14]. General representation can be denoted as

$$\phi(h, r, t) = \delta(\tau(h, r), t)$$

2) Matrix Factorization: Link prediction is treated as a tensor decomposition task in this approach. The knowledge graph can be viewed as an incomplete three-dimensional (3D) adjacency matrix, represented by a three-way tensor. This can be decomposed into a collection of low-dimensional vectors, specifically the embeddings of entities and relations. In practice, the embeddings are learnt afterwards using the available facts, but they are expected to generalize and assign high scores to unseen true facts. To calculate the score of a fact, the combination operation is applied to the embeddings of its components. These models typically have very few or no shared parameters, which makes them lightweight and easy to train [14].

3) Deep Learning: These models employ deep neural networks to carry out link prediction. Neural networks acquire parameters like weights and biases, which, when combined with input data, enable the recognition of significant patterns. Deep networks organize these parameters into distinct layers, often accompanied by non-linear activation functions. Over time, various types of layers have been developed. For instance, dense layers simply combine input data X with weights W and incorporate a bias $b : W \cdot X + b$. To maintain simplicity, the subsequent formulas will implicitly omit the mention of bias. More advanced layers perform intricate operations, such as convolutional layers that learn convolution kernels to apply to input data or recurrent layers that handle sequential inputs recursively [14].

In LP research, knowledge graph embeddings are typically learned concurrently with the weights and biases of the layers. These shared parameters enhance the expressiveness of the models, but they may result in longer training times and increased susceptibility to overfitting [14].

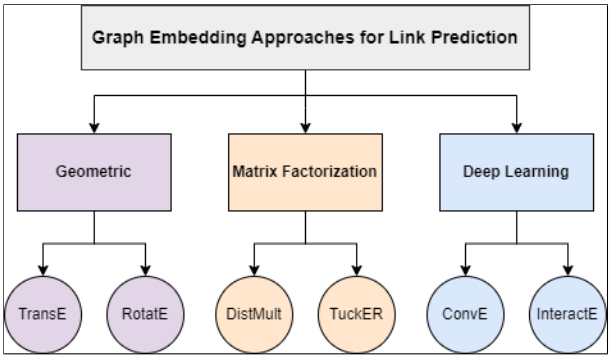

Fig. 2: Graph Embedding Approaches Summary

D. *Geometric Method*

Within this method, we have considered two mainly used approaches: TransE and RotatE.

1) TransE: This is one of the novel embedding methods that has led to many new techniques (based on it) employed by other scientists over the years. Bordes, A., et. al. [15] proposed a method for learning low-dimensional embedding of entities through an energy-based model that is simplistic to train, contains few parameters, and scales well for large graphs. In this case, relationships are expressed as translations within the embedding space. Each triplet *(h,l,t)* that belongs to a set *S*, includes a head and tail entity which is denoted by *h* and *t* respectively.

The two entities are then linked through a relationship *l*. The main approach behind the embedded translations is that if there exists a relationship, then *t* should be a close neighbour for *h+l*, or else they should be far apart. In addition, the L_1 or L_2 norm is applied as a dissimilarity measure *d* which represents the energy for each triplet as *d(h+l, t)*. In order to learn the embeddings, the model tries to reduce a margin-based ranking criterion within the training set denoted as $[\gamma + d(h + l, t) - d(h' + l, t')]_+$ where γ is a margin hyperparameter and $[x]_+$ represents the positive part of x. Next, either the head or tail (but not both) is substituted with a random entity to form a set of corrupted triplets through the equation $S'_{(h,l,t)} = \{(h', l, t)|h' \in E\} \cup \{(h, l, t')|t' \in E\}$. Lower values of energy within the training triplets are preferred by the loss function as compared to the corrupted triplets.

To implement the optimization, stochastic gradient descent is applied in minibatch mode across the different *h*,

l, *t*. A constraint is also added, such that the $L_2 - norm = 1$ across the entity embeddings except for *l*. This ensures that the loss function does not artificially inflate the norms of entity embeddings easily during the training process. Throughout the algorithm, for each given iteration, the embedded vectors are initially normalized. Next, a portion of the triplet set is taken and utilized for the minibatch training triplets. Furthermore, for each of those triplets, one corrupted triplet is sampled as well. Finally, the algorithm takes the next gradient step (same learning rate) and updates its parameters. The stopping criteria for the algorithm is then dependent on either the performance or a validation set.

2) RotatE: In this approach, the authors [18] provide a different method, where each relation is represented as a rotation from the source to its target entity within the complex vector space. The model is said to be able to infer multiple types of relational patterns which include symmetry/antisymmetry, inversion, as well as composition. Furthermore, they also propose a novel self-adversarial negative sampling technique to improve the efficiency of training the RotatE model which is also able to scale very well.

Given a triple (h, r, t), the head (h) and tail (t) is mapped to the complex embeddings as $h, t \in C^k$. An element-wise rotation is applied from h to t denoted as $t = h \circ r$, where $|r_i| = 1$, and $\circ$ is the Hadmard product. The modulus for each element of r is constrained in the embeddings as $|r_i| = 1$ for $t_i = h_i r_i$. This results in a counter-clockwise rotation by $\theta_{r,i}$ radians from the complex plane origin.

A distance function is also defined for each triple (h, r, t), denoted as $d_r(h, t) = ||h \circ r - t||$. By implementing this rotation, the model is able to infer the three kinds of patterns mentioned previously.

As part of the optimization technique, a loss function is used that is similar to the negative sampling loss [19]. The self-adversarial negative sampling approach is used to deal with the issue of uniform negative sampling where there is an inefficiency due to false samples as the training progresses, which does not provide any additional benefit.

E. Matrix-factorization Method

Within this method, we have considered two mainly used approaches: DistMult and TuckER.

1) DistMult: Firstly, in the DistMult model, real values are used to represent the embedding vectors. Next, the bilinear product of subject, predicate, and object embeddings is employed which behaves as an interaction function to encode knowledge facts, resulting in the learning of symmetric representations for all predicates, as the product operator on real numbers possesses inherent symmetry. Thus, this state-of-the-art model's results as described in [20], [21] are known to be obtained in linear time and space complexity, indicating their high scalability and efficiency, which is a significant improvement over the scalability issues faced by the traditional approaches of Markov-logic networks [22] or Bayesian clustering framework [23]. And even using the low-dimensional representation for entities, these methods strongly perform well in terms of reasoning and validating unseen facts in the Knowledge Bases. DistMult is far superior to other distance-based embedding techniques such as TransE [15], however, encounters challenges in modelling facts with asymmetric predicates due to its underlying design.

In the context of a knowledge base represented as a list of relation triplets (e1, r, e2), where e1 represents the subject, e2 represents the object, and r denotes the relationship between them [20], the objective of the graph embeddings is to assign high scores (i.e., low energies) for a valid set of triplets. This process is achieved utilizing a neural network, with the first layer of the network mapping a pair of input entities to low-dimensional vectors, while the second layer combines these vectors into a scalar value. This scalar is then compared with relation-specific parameters using a scoring function. Associating each entity with a high-dimensional vector (either one-hot encoded or n-hot feature vector), entities e_1 and e_2 (x_{e1}, x_{e2} - input entity vectors) can be expressed as the learnt entity representation y_{e1}, y_{e2} as $y_{e1} = f(W x_{e1})$ and $y_{e2} = f(W x_{e2})$, where f is a linear or non-linear function and W is the projection matrix of the first layer. For representing relations, $g_r(y_{e1}, y_{e2}) = y_{e1}^T * M_r * y_{e2}$ is used, where $M_r \in R^{n*n}$, which is also used in [24]. Further, to learn the parameters of the model, negative triplets are introduced by corrupting either one of the relational arguments and minimizing the margin-based ranking loss. Formulating link prediction as a task to rank entities, in the testing dataset for each of the triplets, each entity is predicted by the model and scores are calculated for all corrected entities as well as corrupt entities further ranking them in descending order.

2) TuckER: As stated by the authors in [25], TuckER (E - entities, R - relations) is a very simple yet powerful linear model that utilizes Tucker decomposition of the binary tensor representation of knowledge graph triples, demonstrating full expressiveness despite bounding dimensionalities on the embeddings. In addition to the traditional triple representation (subject, predicate, object), knowledge graphs can be represented using a binary tensor. This tensor assigns values of 1 to true facts and 0 to unknown facts (false or missing).

Using Tucker decomposition, an entity embedding matrix E, equivalent for both subject and object entities $E = A = C \in R^{n_e * d_e}$ and relation matrix $R = B \in R^{n_r * d_r}$, where n_e, n_r represent the number of entities and relations and d_e, d_r dimensionality of the entity and relation embedding vectors. Thus, in the context of link prediction, using the true triples, a score is obtained, indicated by:

$$\phi(e_s, r, e_o) = W x_1 e_s x_2 w_r x_3 e_o$$

wherein $e_s, e_o \in R^{d_e}$, $w_r \in R^{d_r}, W \in R^{d_e * d_r * d_e}$ and W is the core tensor, in order to accurately score all the missing triples. Then, their corresponding probability is obtained $p = \sigma(s) \in [0, 1]$, where σ is the logistic sigmoid function. Also, the number of parameters in TuckER increases linearly with the entity and relation embedding dimensionalities, rather than depending on the number of entities or relations. Unlike simpler models such as DistMult, ComplEx, and SimplE, TuckER does not encode all the learned knowledge solely into the embeddings. Instead, some knowledge is stored in the core tensor (W) and shared among all entities and relations through multi-task learning. Thus, the core tensor of TuckER can be viewed as a shared pool of prototype relation matrices, which are linearly combined based on the parameters in each relation embedding. For training the model 1-N scoring like in [17] is used and is optimized by minimizing the Bernoulli negative log-likelihood loss function.

F. Deep Learning

Within this method, we have considered two mainly used approaches: ConvE and InteractE.

1) ConvE: In contrast to the matrix factorization models mentioned in [20] [24], which prioritize speed and simplicity at the cost of expressive features, scaling these models by increasing the feature set becomes impractical for larger graphs. One solution to address the scaling issue of shallow architectures and the overfitting problem of fully connected deep architectures is to employ parameter-efficient and fast operators that can be combined into deep networks, which aligns with the objective of this approach [17]. Operating spatially over graph embeddings, 2D-convolutions (ConvE) stand out as a straightforward multi-layer convolutional architecture designed for link prediction, comprising of a single convolution layer, a projection layer that reduces dimensions to the embedding size, and an inner product layer. In the field of natural language processing, numerous convolutional models have been proposed to tackle diverse tasks like semantic parsing [26], [27], but majorly relying only on 1D-convolutions.

Any knowledge graph is represented as (s,r,o) $\subset$ E*R*E can be defined as a collection of triples [17]. Each triple comprises a relationship r $\in$ R and two entities s & o $\in$ E, representing the subject and object of the triple, respectively. By expressing (s,r,o), we indicate that there exists a relationship of type r between the entities s and o. The problem of link prediction can be formulated as a point-wise ranking problem, wherein the goal is to learn a scoring function $\psi : E * R * E \rightarrow R$. For a given input triple x = (s,r,o), the score $\psi(x) \in R$ is indicative of the likelihood that the fact represented by x is true. For training the model's parameters, the logistic sigmoid function is used, for the subject and object $p = \sigma(\psi_r(e_s, e_o))$. In addition to this, the binary cross-entropy loss is minimized and rectified linear units are employed for faster training. Further, batch normalization is applied after each layer for stability, regularization, and faster convergence. In contrast to alternative link prediction models that consider a triplet (s,r,o) consisting of an entity pair and a relation, scoring it using a one-to-one (1-1) approach, this method involved a single (s,r) pair and simultaneously scoring it against all entities o $\in$ E, employing a one-to-many (1-N) scoring approach.

2) InteractE: Addressing the drawbacks of ConvE, i.e. capturing only limited rich interactions between the embeddings, InteractE as proposed in [30] relies on three main notions - feature permutation, reshape features and employing circular convolution. It also highlights how increasing the interaction between the features benefits the performance of the link prediction problem. Other than ConvE, there are variants like ConvKB [32], ConvTransE, SACN [33], etc. that employed approaches based on convolutional neural networks but suffered performance issues across various datasets and metrics. Instead of relying on a single fixed order of input, the authors [30] employed multiple permutations in the approach to capture a wider range of potential interactions. Further, checkered reshaping is used in place of simple feature reshaping (as opposed to ConvE) and like ConvE, circular convolution is used to capture additional feature interactions.

In InteractE [30], the model learns a d-dimensional vector represented as ($e_s, e_r \in R^d$) for each entity as well as relation in the knowledge graph, where d = $d_w d_h$. To encompass diverse types of heterogeneous interactions, InteractE initiates the process by generating t-random permutations of both e_s and e_r, resulting in a set of transformed entities and relations denoted by $P_t = [(e_s^1, e_r^1); ...; (e_s^t, e_r^t)]$ and $\phi(e_s^i, e_r^i)$ are the sets of interactions for different *i* which occur with high probability. Next, it applies the reshaping operator along with circular convolution $\phi_{chk}(e_s^i, e_r^i)$, which effectively captures a wide range of heterogeneous interactions between entity and relation features. It also employs a stacking approach, treating each reshaped permutation as an individual channel and to convolve the permutations, circular convolution is applied in a depth-wise fashion [34]. In the scoring function, the results of each circular convolution are flattened and combined into a vector. Subsequently, InteractE projects this vector into the embedding space. The results of each circular convolution operation are flattened and concatenated to form a vector. This vector is then projected into the embedding space (R^d) by InteractE. Specifically, the score function in InteractE is defined as follows:

$$\psi((s, r, o) = g(vec(f(\phi(P_k) \circledast w))W)e_o$$

where $\circledast$ represents the depth-wise circular convolution, vec($\cdot$) denotes the concatenation of vectors, e_o represents the embedding matrix for the object entity, and W is a trainable weight matrix. The functions f and g are selected as the rectified linear unit (ReLU) and sigmoid, respectively. During

training, the standard binary cross-entropy loss with label smoothing is utilized.

III. Approach Scalability

Model	Scoring Function	Space Complexity
TransE	$\|\|\|e_s + e_r - e_o\|\|$	$\mathcal{O}(\|\varepsilon\|d + \|R\|d)$
RotatE	$\|\|\|e_s \circ e_r - e_o\|\|^2$	$\mathcal{O}(\|2\varepsilon\|d + 2\|R\|d)$
DistMult	$< e_s, e_r, e_o >$	$\mathcal{O}(\|\varepsilon\|d + \|R\|d)$
TuckER	$W x_1 e_s x_2 w_r x_3 e_o$	$\mathcal{O}(\|\varepsilon\|d_e + \|R\|d_r + d_e d_r d_e)$
ConvE	$f(vec(f([e_s; e_r] \star w))W)e_o$	$\mathcal{O}(\|\varepsilon\|d + \|R\|d)$
InteractE	$g(vec(f(\phi(P_k) \circledast w))W)e_o$	$\mathcal{O}(\|\varepsilon\|d + \|R\|d + 2d^2)$

TABLE II: Scoring & complexity comparison [14], [20]

Table II summarizes the six models with their scoring functions and space complexities. $|\varepsilon|$ denotes the number of entities (subjects as well as objects), $|R|$ denotes the number of relations, d is the dimension on embedding with d_e, d_r being the dimensions of entity embedding and relation embedding respectively. The geometric and matrix-based methods consume linear space while the deep learning method of InteractE consumes quadratic space. Interestingly, the model that performs the best on FB15k-237 as discussed in Section IV - TuckER, has a higher space complexity than others to capture richer interactions among the entities and relations. On the other hand, RotateE performs exceptionally well in both datasets while occupying linear space, which is pretty impressive.

IV. Performance and Results

Dataset		FB15k-237		WN18RR	
Approach ↓	**Metric →**	**MRR**	**Hits@10**	**MRR**	**Hits@10**
Geometric	TransE	0.294	0.354	0.226	0.501
	RotatE	0.338	0.533	**0.476**	**0.571**
Matrix Factor.	DistMult	0.241	0.419	0.430	0.490
	TuckER	**0.358**	**0.544**	0.470	0.526
Deep Learning	ConvE	0.325	0.501	0.430	0.520
	InteractE	0.354	0.535	0.463	0.528

TABLE III: Consolidated Results [18], [25], [30]

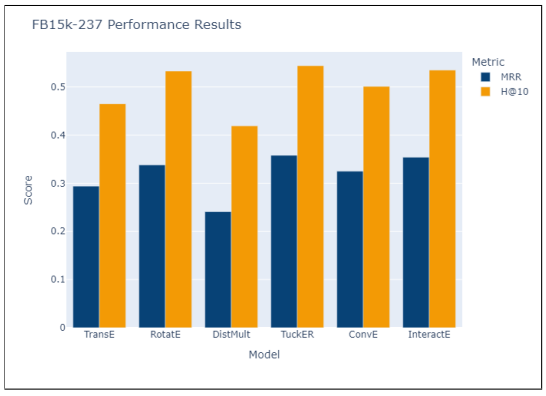

Fig. 3: FB15k-237 Performance Results

Starting with the FB15k-237, the main relation patterns involved composition patterns. In this case, we can see that the older approaches: TransE and DistMult do not perform as well as the newer approaches in terms of both MRR and H@10. This is possibly due to the fact that when these two models were tested on the older versions of this dataset, the issue with data leakage by having inverse patterns in training may have masked its true potential of properly inferring unseen data. On the other hand, it can be observed from Table III and Figure 3, that the newer approach TuckER edges out InteractE and RotatE in terms of inferring these composition patterns well (both MRR and H@10).

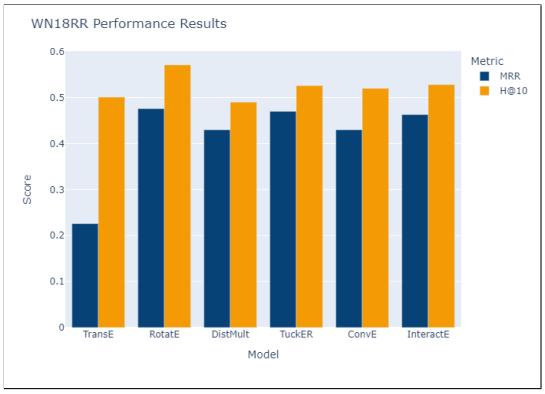

Fig. 4: WN18RR Performance Results

For WN18RR, the results differ slightly, whereas TransE suffers a lot here as compared to the other models, due to its inability to deal with symmetrical patterns [18]. DistMult and ConvE perform comparably, but RotatE infers the best here, with both TuckER and InteractE doing exceptionally well too.

Overall, in terms of both scalability and the ability to infer higher-ranked missing links, RotatE demonstrates its advantages in both of these areas, indicating a very well thought and designed model by the authors.

V. Conclusion

To conclude, we have presented six different state-of-the-art LP techniques that are used to infer missing links in KGs, as well as a performance analysis on two well-known datasets (FB15k-237 and WN18RR) that do not include data leakage. Our priority was to benchmark the different models based on their ability to infer the higher-ranked missing links, thus using the evaluation metrics, MRR and Hits@10. Using approaches such as binary tensor decomposition (TuckER), complex vector spaces with rotations (RotatE), or deep learning approaches with increased feature interactions (InteractE) has proven to improve the ability to infer these missing links. Furthermore, RotatE has shown the capability of performing exceptionally well, while maintaining a linear space complexity as compared to TuckER and InteractE.

VI. Limitations and Future Work

Firstly, given that each paper may have a variety of implementation methods, it was essential for us to identify

approaches that were state-of-the-art and at the same time using a common dataset and metric evaluation. In fact, some papers have additional evaluation methods such as a comparison of different relation types in terms of multiplicities (1-1, 1-N, N-1, N-N), but given that they are not outlined in other publications, we were unable to place more emphasis on them to keep the comparison fair for each model. Furthermore, due to the complex nature and variety of each approach, we did not get the chance to conduct a full experimental setup, to confirm the claims of the authors and add our own experimentation to it as well. Finally, some papers do not fully elaborate on the extent to which components contributed mainly to the performance of their model, thus making it more difficult to fully understand the distinction factors within the chosen approaches.

As part of our future work, we would like to implement a full experimental setup to explore various evaluation methods and test on additional datasets that could include heterogeneous graphs, imbalanced datasets, and YAGO3-10, a very large-scale dataset as explained in Section II.A. In addition, we aim to explore more domains, to provide suggestions on which type of approach works better in different scenarios. Lastly, in each type of approach, we can explore more base models like HolE, ComplEX, etc. or an extension or even a combination of models such as RSN, Trans4E, etc. to further extend our comprehensive analysis and contribute more to the research community.

REFERENCES

[1] Goyal, P., & Ferrara, E. (2018). Graph embedding techniques, applications, and performance: A survey. Knowledge-Based Systems, 151, 78-94.

[2] Godec, P. (2019, June 26). Graph embeddings - the summary. Medium. https://towardsdatascience.com/graph-embeddings-the-summary-cc6075aba007

[3] Nayyeri, M., Cil, G. M., Vahdati, S., Osborne, F., Rahman, M., Angioni, S., ... & Lehmann, J. (2021). Trans4E: Link prediction on scholarly knowledge graphs. Neurocomputing, 461, 530-542.

[4] Chen, J., He, H., Wu, F., & Wang, J. (2021, May). Topology-aware correlations between relations for inductive link prediction in knowledge graphs. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 35, No. 7, pp. 6271-6278).

[5] Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., & Bouchard, G. (2016, June). Complex embeddings for simple link prediction. In International conference on machine learning (pp. 2071-2080). PMLR.

[6] Bhowmik, R., & de Melo, G. (2020). Explainable link prediction for emerging entities in knowledge graphs. In The Semantic Web–ISWC 2020: 19th International Semantic Web Conference, Athens, Greece, November 2–6, 2020, Proceedings, Part I 19 (pp. 39-55). Springer International Publishing.

[7] Daud, N. N., Ab Hamid, S. H., Saadoon, M., Sahran, F., & Anuar, N. B. (2020). Applications of link prediction in social networks: A review. Journal of Network and Computer Applications, 166, 102716.

[8] Li, J., Zhang, L., Meng, F., & Li, F. (2014). Recommendation algorithm based on link prediction and domain knowledge in retail transactions. Procedia Computer Science, 31, 875-881.

[9] Huang, Z., Li, X., & Chen, H. (2005, June). Link prediction approach to collaborative filtering. In Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries (pp. 141-142).

[10] Cai, B., Wang, H., Zheng, H., & Wang, H. (2013). Integrating domain similarity to improve protein complexes identification in TAP-MS data. Proteome science, 11(1), 1-14.

[11] Long, Y., Wu, M., Liu, Y., Fang, Y., Kwoh, C. K., Chen, J., ... & Li, X. (2022). Pre-training graph neural networks for link prediction in biomedical networks. Bioinformatics, 38(8), 2254-2262.

[12] Taskar, B., Wong, M. F., Abbeel, P., & Koller, D. (2003). Link prediction in relational data. Advances in neural information processing systems, 16.

[13] Sanna Passino, F., Bertiger, A. S., Neil, J. C., & Heard, N. A. (2021). Link prediction in dynamic networks using random dot product graphs. Data Mining and Knowledge Discovery, 35(5), 2168-2199.

[14] Rossi, A., Barbosa, D., Firmani, D., Matinata, A., & Merialdo, P. (2021). Knowledge graph embedding for link prediction: A comparative analysis. ACM Transactions on Knowledge Discovery from Data (TKDD), 15(2), 1-49.

[15] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., & Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. Advances in neural information processing systems, 26.

[16] Toutanova, K., & Chen, D. (2015, July). Observed versus latent features for knowledge base and text inference. In Proceedings of the 3rd workshop on continuous vector space models and their compositionality (pp. 57-66).

[17] Dettmers, T., Minervini, P., Stenetorp, P., & Riedel, S. (2018, April). Convolutional 2d knowledge graph embeddings. In Proceedings of the AAAI conference on artificial intelligence (Vol. 32, No. 1).

[18] Sun, Z., Deng, Z. H., Nie, J. Y., & Tang, J. (2019). Rotate: Knowledge graph embedding by relational rotation in complex space. arXiv preprint arXiv:1902.10197.

[19] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 26.

[20] Yang, B., Yih, W. T., He, X., Gao, J., & Deng, L. (2014). Embedding entities and relations for learning and inference in knowledge bases. arXiv preprint arXiv:1412.6575.

[21] Mohamed, S. K., & Nováček, V. (2019). Link prediction using multi part embeddings. In The Semantic Web: 16th International Conference, ESWC 2019, Portorož, Slovenia, June 2–6, 2019, Proceedings 16 (pp. 240-254). Springer International Publishing.

[22] Getoor, L., & Taskar, B. (Eds.). (2007). Introduction to statistical relational learning. MIT press.

[23] Kemp, C., Tenenbaum, J. B., Griffiths, T. L., Yamada, T., & Ueda, N. (2006, July). Learning systems of concepts with an infinite relational model. In AAAI (Vol. 3, p. 5).

[24] Nickel, M., Rosasco, L., & Poggio, T. (2016, March). Holographic embeddings of knowledge graphs. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 30, No. 1).

[25] Balažević, I., Allen, C., & Hospedales, T. M. (2019). Tucker: Tensor factorization for knowledge graph completion. arXiv preprint arXiv:1901.09590.

[26] Yih, W. T., Toutanova, K., Platt, J. C., & Meek, C. (2011, June). Learning discriminative projections for text similarity measures. In Proceedings of the fifteenth conference on computational natural language learning (pp. 247-256).

[27] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. Journal of machine learning research, 12(ARTICLE), 2493-2537.

[28] Kazemi, S. M., & Poole, D. (2018). Simple embedding for link prediction in knowledge graphs. Advances in neural information processing systems, 31.

[29] Zhang, W., Paudel, B., Zhang, W., Bernstein, A., & Chen, H. (2019, January). Interaction embeddings for prediction and explanation in knowledge graphs. In Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining (pp. 96-104).

[30] Vashishth, S., Sanyal, S., Nitin, V., Agrawal, N., & Talukdar, P. (2020, April). Interacte: Improving convolution-based knowledge graph embeddings by increasing feature interactions. In Proceedings of the AAAI conference on artificial intelligence (Vol. 34, No. 03, pp. 3009-3016).

[31] Guo, L., Sun, Z., & Hu, W. (2019, May). Learning to exploit long-term relational dependencies in knowledge graphs. In International conference on machine learning (pp. 2505-2514). PMLR.

[32] Nguyen, D. Q., Nguyen, T. D., Nguyen, D. Q., & Phung, D. (2017). A novel embedding model for knowledge base completion based on convolutional neural network. arXiv preprint arXiv:1712.02121.

[33] Shang, C., Tang, Y., Huang, J., Bi, J., He, X., & Zhou, B. (2019, July). End-to-end structure-aware convolutional networks for knowledge base completion. In Proceedings of the AAAI conference on artificial intelligence (Vol. 33, No. 01, pp. 3060-3067).

[34] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1251-1258).