A Comparative Study on Deep Learning methods for Link Prediction

Abhirama Subramanyam
penamakuri.1@iitj.ac.in

Asha Rani
rani.1@iitj.ac.in

Harsha M
harsha.1@iitj.ac.in

November 16, 2020

1 Introduction

Since the advent deep learning has gained enormous popularity in may fields, owing to which researches have looked at solving some traditional problems in the literature using deep learning. One such problem is Link Prediction. Through this project we look at few the then State of the Art to current State of the Art Deep learning techniques, noting the methodological changes and advances.

The very basic definition of link prediction is to predict the missing link between the nodes in a incomplete graph (knowledge graph/social network/web graphs etc) or to predict future links.

Few Basic Methods of Link Prediction Include:-

1. Common Neighbours
2. Jaccard Index
3. Distance Based Metrics like Cosine Similarity
4. Page Rank
5. SimRank
6. Adar Index
7. Shortest Path

2 Brief Recapitulation of Definitions:

Link Prediction Task related experiments and results are generally tend to be shown on **Multi Relational Graphs**. As many real world networks like product network for example is a multi relational graph. To explain it with an example, in a product graph, users, products, reviews are nodes, and the relations between them, like this {product} *"is bought*

by" {user}, {review} **"is given to"** {product}, here **"is bought by"** and "is given to" are the relations between nodes.

Any Multi Relational Graph can be represented as set of facts, for e.g. Consider a Graph **G {V,E,R}** where **V** is set of vertices, **E** is set of edges and **R** is set of relations. Every pair of nodes $u, v \in V$ and the edge $u \rightarrow v \in E$, can be represented as (u,r,v) which can be read as u is connected to v under the relation "**r**". This terminology varies from paper to paper. Though, we explain most commonly used nomenclature of the same, they use triplets **(s,r,o)** where **s** is **subject**, r is **relation** and o is **object.**

2.1 Redefinition of Task

Given **s** and **o**, the task is to predict **r**, which upon slight modification can be defined as the following: Given **s** and **r** predict **o**, all the papers in our project tend to follow the second definition.

2.2 Mathematical Definition

Knowledge Graph $\mathcal{G} = (s, r, o) \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$

Knowledge Graph Embeddings:- e_s subject embedding and e_o object embedding,and problem is to find the Scoring function $\Psi(s, r, o) = \Psi_r(e_s, e_o)$

3 Papers

3.1 ConvE

This paper contributes two critical things:

1. Multilayer convolutional network for link prediction
2. It quantifies the presence of test set leakage in the datasets used until then for link prediction task, i.e., FB-15k and WN18.

The paper proposes a model ConvE in Fig 1, which uses the concept of 2D reshaping. The components of the graph are transformed into vectors to represent them in the latent space. The embeddings of entity and relation vectors are reshaped into 2D to increase interaction, then concatenated, followed by the dropout of 0.2.The output is then convolved, which produces the feature map, then again dropout is used. The vectorized output is projected with the required dimension, and then sigmoid is applied to get probabilistic value, i.e., the result is the value of score function.

The scoring function used in this paper:

$$\psi(e_s, e_o) = f(vec(f([\bar{e_s}; \bar{r_r}] * w))W)e_o \tag{1}$$

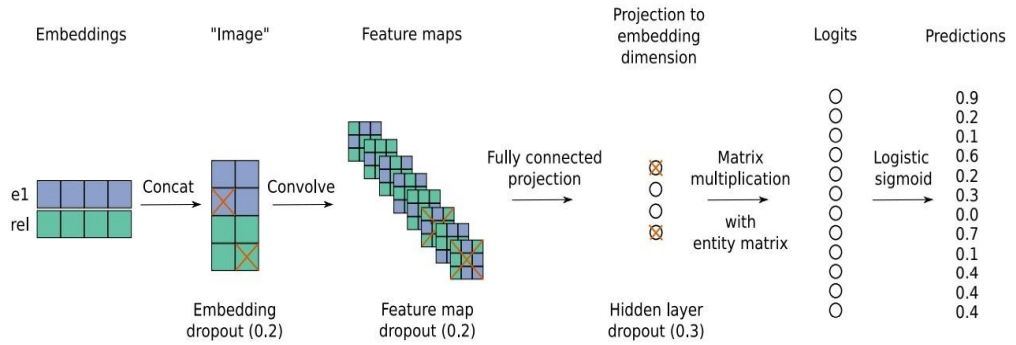


Figure 1: ConvE Model

Until this paper, leakage in the test set was suspected, but no quantified results were presented. To prove this, the authors have proposed an inverse model, which investigates the inverse pairs present in the test and train set. Example, a tuple (s,r1,o) is present in the test set and another tuple (s,r2,o) in the train set.So, the other tuple can be produced by applying the inverse. In such scenario, model will not learn from the representations but will use these inverse relationships to predict the links. The authors have contributed two datasets WN18RR and FB15k-237, after removing such inverse relation tuples from the datasets.

3.2 HypER

ConvE was able to produce impressive results and also work well work large scale real-world data. But the approach was not well explained. HypER presents the approach to model the entity and relationship well, according to the type of relationship. Contrary to ConvE, which uses a global set of 2D filters, HypER uses filters according to the type of relationships, showed in Fig 2. This idea was inspired from the concept of computer vision.In this model , authors have used series of tensor multiplications.For the training ,authors have used 1-N scoring methods ,Adam optimizer and binary cross entropy loss, while ConvE uses 1-1 scoring method. The number of parameters used in HypER is fewer

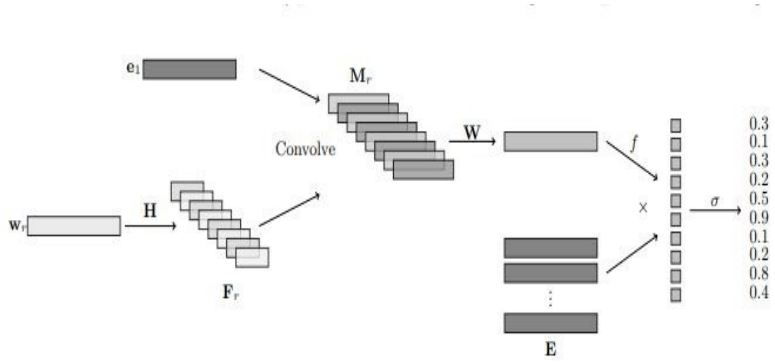


Figure 2: HypER Model

in comparison to that in ConvE, but it performs better on the same datasets.This suggest 1D interaction in HypER with relation specific filters convolves the embedding better than over reshaping the embeddings as 2D.To show the influence of Hyper network component authors have experimented by removing the hypernetwork which shows with hypernetwork model performs better, along with learning from multiple relationship benefits.HypER is faster than ConvE , this we observed even while running the code of these papers.
The scoring function is represented as:

$$\phi_r(e_1, e_2) = f(vec(e_1 * vec^{-1}(w_r H))W)e_2 \tag{2}$$

3.3 ConvKB

Knowledge bases are usually databases of triples representing the relationships between entities in the form of fact. Facts are represented as (head entity, relation, tail entity) denoted as (h, r, t), e.g., (Jodhpur, city Of, Rajasthan). However in knowledger bases, many valid triples will be missing. That means that the knowledge bases are still incomplete. This paper focuses on knowledge base completion and link prediction to predict whether a triple (h, r, t) is valid or not. The model **ConvKB** proposed in this paper generalizes transitional characteristics in transition-based embedding models. ConvKB models the relationships among same dimensional entries of the subject and relation embeddings.

In ConvKB, each triple (head entity, relation, tail entity) is represented as a 3- column matrix where each column vector represents a triple element. This 3-column matrix is then fed to a convolution layer where multiple filters are operated on the matrix to generate different feature maps. These feature maps are then concatenated into a single feature vector representing the input triple. The feature vector is multiplied with a weight vector via a dot product to return a score.
Embedding models aim to define a score function f giving an implausibility score for each triple (h, r, t) such that valid triples receive lower scores than invalid triples.

- feature map $\mathcal{V} = [v_1, v_2, ..., v_k]\ \epsilon R_k$ is generated as $v_i = g(w \cdot A_{i,:} + b)$
- ConvKB score function is $f(h, r, t) = concat(g([v_h, v_r, v_t]) * \Omega) \cdot w$
- Loss function is as follows:

$$\mathcal{L} = \sum_{(h,r,t)\epsilon\{\mathcal{G}\mathcal{G}'\}} log(1 + exp(l_{(h,r,t)} \cdot f(h, r, t))) + \frac{\lambda}{2}||w||^2 \tag{3}$$

  where $l_{(h,r,t)}$ is 1 if it belongs to valid triple set $\mathcal{G}$, -1 if it belongs to invalid triple set $\mathcal{G}'$.

3.4 ConvR

In this paper, the main objective is to model multi-relational graphs. Multi-relational graphs consists of directed graphs, where nodes are considered as entities (subject and object) and the different relationships between the pair of nodes can be considered as the relations. The model has been designed such that it produces maximum interactions between subject and relation data. In Statistical Relational Learning (SRL), multi-relational graphs are modelled and are used in various applications such as predicting missing facts in knowledge bases and link prediction in social networks. The design consists of adaptive convolutional neural network which will captures interaction on both sides more naturally and effectively.

3.4.1 Comparison with ConvE

ConvE was the one of the earliest methods to model such kind of graphs using CNNs. But, it fails to capture sufficient interactions for multi-relational data, since it uses global filters and both subject and relation are applied as inputs. However in ConvR performs adaptive convolution where relation embeddings is itself reshaped into filters and convolved along with the subject embeddings. This helps in capturing better relationships. This can be explained using the following figure:

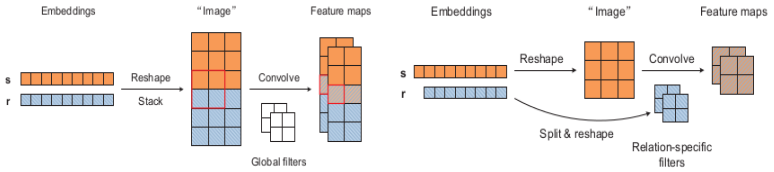


Figure 3: ConvE (left) versus convR (right)

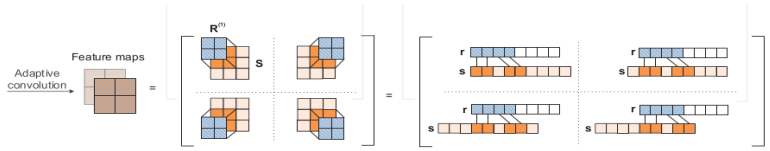


Figure 4: Adaptive Convolution by ConvR

The scoring function is calculated using :

$$\psi(s, r, o) = f(Wc + b)^T o \tag{4}$$

- Each dimension of this score vector corresponds to an entity o $\epsilon\mathcal{E}$, calculated as

${p_o}^{s,r} = \sigma(\psi(s,r,o))$ where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function. Hence, cross-entropy loss function is used.

- For link prediction task, it is evaluated on **WN18RR** and **FB15K-237** datasets and it performs better than convE.
- ConvR enables entity-relation interactions at diverse regions, and all the convolutional features are able to capture such interactions better than convE.
- On comparison with ConvE, this model has increase in MRR and a 6% increase in Hits@10, while saving 12% in parameter storage.

3.5 R-GCN

All the previous approaches, used only node related information as input to the model, so the model is not structure aware, i.e., model is not aware of the structure of the graph. To overcome this shortcoming, researchers have come up with Graph Neural Network models. These models make use of neighbourhood of the node in calculating node's embedding. These models are shown to learn the structure and proven to be effective than the earlier methods. A brief introduction to these models is given below.

3.5.1 Introduction to Graph Neural Networks

Embedding of node u at step k is updated by aggregating (averaging) the information (embeddings) of its neighbors in the graph $\mathcal{G}$.

$$h_v^0 = x_v \tag{5}$$

$$h_v^k = \sigma\left(W_k \sum_{u \in \mathcal{N}(v)} \frac{h_u^{k-1}}{|N(v)|} + B_k h_v^{k-1}\right), \forall k > 0 \tag{6}$$

where h_v^{k-1} is previous layer embedding of v, $|N(v)|$ is the average of neighbor's previous layer embeddings, σ is the non-linearity (e.g. ReLU or tanh), h_v^k is kth layer embedding of v, x_v is the initial layer 0 embeddings.

3.5.2 Introduction to Graph Convolutional Neural Networks

GCN's are slight variation on the neighborhood aggregation idea as these GCN's use per neighborhood normalization. They tend to use same weight matrices for self and neighbor embeddings.

$$h_v^k = \sigma\left(W_k \sum_{u \in \mathcal{N}(v)} \frac{h_u^{k-1}}{\sqrt{|N(v)||N(u)|}} + B_k h_v^{k-1}\right), \forall k > 0 \tag{7}$$

3.5.3 R-GCN

R-GCN is similar to vanilla GCN, which we have introduced in the last section, the only difference is R-GCN works for Multi-Relational Graphs wherein vanilla GCN's doesn't. The aggregation function of R-GCN is similar to that of traditional GCN, but it will have a different weight matrix for each relation, say W_r, i.e.,

$$h_i^{k+1} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}W_r^k h_j^k + W_0^k h_i^k\right) \tag{8}$$

Through this proposed update, R-GCN is able to solve the problem that GCN has on Multi-Relational Graphs. Though this addresses the problem, but sufferes from over-parameterization, i.e., the number of weights (learnable parameters) will increase with increase in the number of relations. Therefore, this is not suitable for real time graphs, like knowledge graphs, friend networks, product graphs etc. as they tend to have many relations and also given the dynamic nature of real time networks, relations tend to increase over time.

3.6 Comp-GCN

To address the over-parameterization problem in the previous R-GCN, authors of this paper have proposed Relation Based Composition, i.e., relations are also represented by embeddings (vectors). Relational Embeddings and their respective weights can also be learned, thereby for one layer of Comp-GCN aggregation we need only four Weight Matrices, which are

1. W_r (for nodes with relations)
2. W_{i_r} (for nodes inverse relations)
3. W_o (for self loops)
4. W_{rel} (for updating relational embeddings)

$$h_v = \sigma\left(\sum_{(u,r)\in\mathcal{N}(v)} W_{\lambda(r)}\phi(x_u, z_r)\right) \tag{9}$$

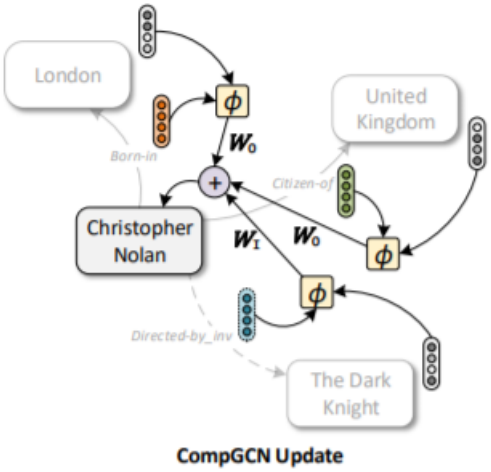
where x_u,z_r denotes initial features for node u and relation r respectively, h_v denotes the updated representation of node v, and $\mathbf{W}_{\lambda(r)} \in \mathcal{R}^{d_1\times d_0}$ is a relation specific parameter. In this paper, they used direction specific weights, i.e., $\lambda(r) = dir(r)$, which is defined as:

$$\mathbf{W}_{dir(r)}\begin{cases}\mathbf{W}_O, r\in\mathcal{R}\\ \mathbf{W}_I, r\in\mathcal{R}_{inv}\\ \mathbf{W}_S, r = T(selfloop)\end{cases} \tag{10}$$

For updating relational embeddings z_r,

$$h_r = \mathbf{W}_{rel} z_r \tag{11}$$

where $\mathbf{W}_{rel}$ is a learnable weight matrix which projects all relational embeddings to the same space as that of node embeddings, so that they can be utilized in the further CompGCN layers. Here's the quick figure, explaining the step in which the embedding of the node is calculated, here in this figure we can see that, to calculate the embedding of Node "Christopher Nolan", unlike GCN, CompGCN has used relational embeddings along with neighbor node embeddings via composition operator ϕ which was experimented with simple operations like subtraction, multiplication, correlation. Node embeddings that are ob-



tained using CompGCN are more robust and very well structure aware, thereby improving link prediction numbers. The final node embeddings and the learned relation embeddings are used to generate top-N object entries with a linear layer at the end, similar to earlier models like ConvE, ConvR, HypER etc.

4 Data

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	Train	Valid	Test
WN18RR	40,943	11	86,835	3,034	3,134
FB15k-237	14,541	237	272,115	17,535	20,466

5 Metrics

- **Mean Rank (MR) :** The average rank assigned to the true triple, over all test triples.
- **Mean Reciprocal Rank (MRR) :** The average of the reciprocal rank assigned to the true triple.

- **Hits@k :** measures the percentage of cases in which the true triple appears in the top k ranked triples.
  - **Example :** Hits@1, Hits@3, Hits@10, ..

  Overall, the aim is to achieve high mean reciprocal rank and hits@k and low mean rank.

6 Results

These results are obtained, by reproducing the code (not from the paper)

Paper	Dataset	MR	MRR	Hits@10	Hits@3	Hits@1
HypER	**FB15k-237**	389.98	0.327	0.508	0.359	0.238
ConvKB	**WN18RR**	1525.98	0.255	0.540	0.418	0.063

7 Summary

1. Link Prediction has been one of the important task in the recent times especially in product networks, friend networks, among other social networks.
2. Deep Learning Community has started to look at solving this problem, by bringing convolution operations (networks) to operate on source and relation to predict the object.
3. With the advancement of Graph Representation Learning techniques like Graph Neural Networks (GNN's) and Graph Convolutional Neural Networks, node embeddings are rich in information, structure aware and are robust. These have improved the accuracy of the link prediction.
4. Through this project, we have studied the deep learning techniques that are being proposed overtime to link prediction starting from simple Convolution based methods to latest Comp GCN.

8 References

1. CompGCN - https://arxiv.org/pdf/1911.03082.pdf
2. R-GCN - https://arxiv.org/pdf/1703.06103.pdf
3. GNN/GCN Slides - Machine Learning - 2 (IIT Jodhpur) Course Slides.

4. ConvE - https://arxiv.org/abs/1707.01476
5. HypER - https://arxiv.org/abs/1808.07018
6. ConvKB - https://www.aclweb.org/anthology/N18-2053.pdf
7. ConvR - https://www.aclweb.org/anthology/N19-1103.pdf