Automatic Edge Prediction:An Application on Knowledge Graphs for Relationship Extraction

Sruthimol Rajesh
(Roll No:47921014)

Under the Guidance of

Dr.Shailesh S
Assistant Professor

Department of Computer Science
Cochin University of Science Technology

Contents

1. Introduction
2. Problem Statement
3. Literature Review
4. Methodology
5. Tools
6. Implementation and Results
7. Conclusion
8. References

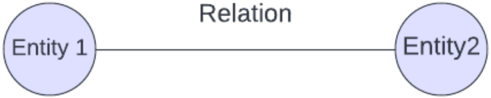
Introduction

Overview

- Knowledge management is important due to the rapid growth of data and the need for efficient storage and organization.
- Knowledge graphs are structured representations of knowledge that enable efficient storage and retrieval of information.
- Nodes and edges in knowledge graphs represent entities and their relationships, and labels capture domain-specific meaning and context.
- Knowledge graphs can integrate information from various sources, enabling better search and analysis of data and the discovery of new insights and relationships.

Knowledge Graph

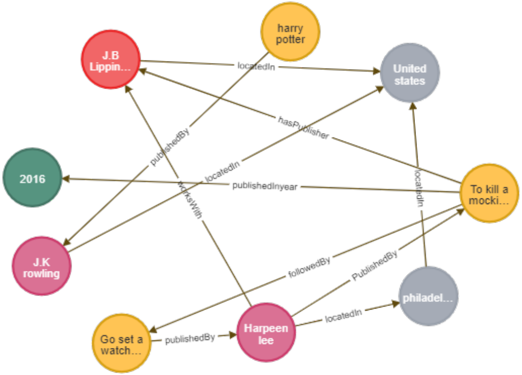
- A **Graph** is a connected Data
- **Knowledge graph** is a collection of interlinked descriptions of entities,relationships and events from the abstract concepts.
- We can create specific instances of each of our ontological relationships



Knowledge Graph

For creating a node and relationship in Neo4J

- create(n:publisher(name:’J.B Lippincott’))
- MATCH (a:book), (b:author) WHERE a.name = ”Go set a watchman” AND b.name = ”Harpeen lee” CREATE (a)-[r:publishedBy]->(b) RETURN r



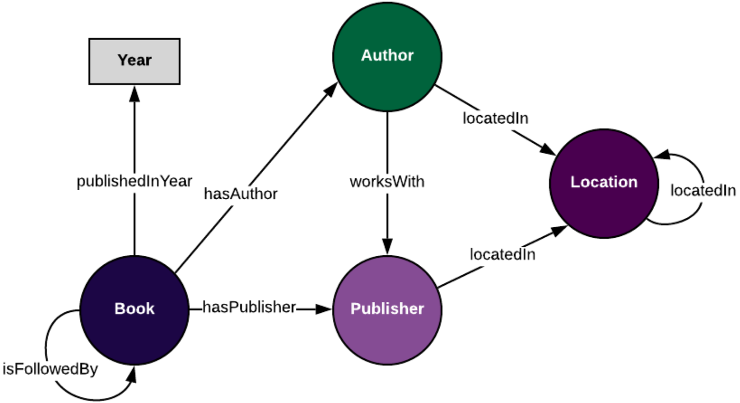
Knowledge graph

- **Knowledge Graph** (KG) is a collection of triples(h,r,t)
- **h,t :** Are the two entities
- **r :** Predicate (also called relation)between them
- One of its pillars are ontologies

Ontology

- Ontologies are abstract representations of knowledge graphs that define concepts and relationships in a formal way.
- Ontologies ensure consistency and logical coherence of knowledge representation
- Used to infer new knowledge and validate and correct existing knowledge.

Ontology

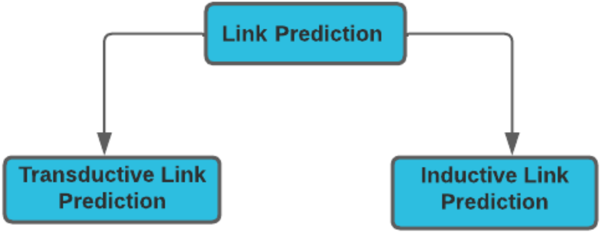

Year
Author
locatedIn
publishedInYear
hasAuthor
worksWith
Location
locatedIn
locatedIn
Book
hasPublisher
Publisher
isFollowedBy

Reasoning

- **Reasoning :**A knowledge graph can provide additional information that is not explicitly stated
- A graph database engine can infer additional facts and relationships, which can then be added to the knowledge graph.
- Discover multi-hop relationships or other patterns
- It can be applied to a set of logical rules to the data to derive a new information is the Link prediction.

Link prediction

- Task of predicting missing or future links in a complex network
- Link prediction is a task to estimate the probability of links between nodes in a graph.
- Link prediction falls into two categories



Link Prediction

Transductive Link Prediction

Inductive Link Prediction

Motivation

- The unceasing growth of knowledge in real life raises the necessity to enable the inductive reasoning ability on expanding KGs.
- Existing inductive work assumes that new entities all emerge once in a batch, which oversimplifies the real scenario that new entities continually appear. This study dives into a more realistic and challenging setting where new entities emerge
- Knowledge graph is more popular and it can be used to different areas
- I am more interested to know about the area of graphs because it can be easily represented
- Graphical representations are more simpler than theory

Problem Statement

Problem Statement

To develop an Inductive learning based Automatic Edge Prediction in different Application on Knowledge Graphs for Relationship Extraction by using feature extraction and neural networks for accurately predict missing links in complex networks.

Literature Review

Literature Review

Author	Year	Technique Adopted	Features Used	Proposed Method
Zhang et al.	2021	Graph neural network (GNN)	Node features and graph topology	Multi-scale Graph Neural Network (MSGNN)
Sun et al.	2021	Deep learning	Node and edge features	Graph neural network with edge attention mechanism
Wu et al.	2021	Network embedding	Node and edge features	Multi-task Network Embedding (MTNE)
Jiang et al.	2020	Graph convolutional network (GCN)	Node and edge features	Edge Weighted Graph Convolutional Network (EW-GCN)

Literature Review

Author	Year	Technique Adopted	Features Used	Proposed Method
Zhang et al.	2020	Deep learning	Node and edge features	Hierarchical Attention Graph Convolutional Network (HAGCN)
Wang et al.	2020	Graph neural network (GNN)	Node and edge features	Relation-aware Graph Convolutional Network (RGCN)
Liu et al.	2019	Deep learning	Node and edge features	Attention-based Graph Convolutional Network (AGCN)
Xu et al.	2019	Network embedding	Node and edge features	Structural Deep Network Embedding (SDNE)

Literature Review

- Transductive link prediction: Previous works on link prediction have focused on transductive learning, where the model is trained on the observed graph and tested on the unobserved edges.
- Graph neural network (GNN) based link prediction: GNNs have shown promising results for link prediction tasks by learning node representations and capturing graph structures.
- Machine learning-based link prediction: Traditional machine learning methods such as logistic regression and SVM have also been used for link prediction tasks.

Inductive Learning

- Inductive Learning based deep learning because it allows the model to generalize to new nodes and edges that were not present in the training data.
- An advantage of Inductive Learning is that the model can be used to predict links in larger graphs without retraining the model every time new nodes or edges are added.

Methodology

Proposed System

- This method is a deep learning-based model for edge prediction in knowledge graphs.
- It utilizes a Deep neural network to learn semantic representations and predict new relationships in different applications.
- The system has advantages in handling complex, large-scale knowledge graphs and extracting implicit relationships.

Computational Framework

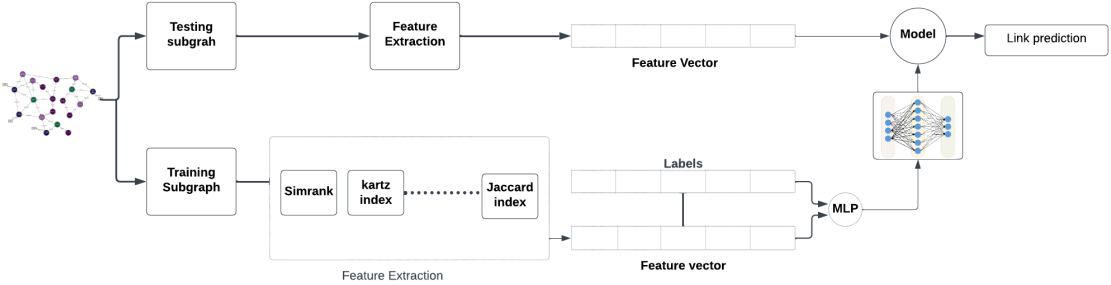


Figure: Computational Framework

Work Flow

1. Network representation
2. Feature extraction
3. Feature Vector
4. Training by Deep Neural Network
5. Link Prediction
6. Evaluation metrics

Dataset

Dataset Name	Nodes	Edges	Directed/Undirected
Karate Club Network	34	78	Undirected
Cora Knowledge Graph	2,708	2,708	Directed
Drug-Gene Interaction	1560	9833	Directed
Facebook Network	4039	88234	Undirected
Tourism Knowledge Graph	2250	5200	Directed

Table: Dataset details

Network Representation

- Networks are often represented as graphs, which are mathematical structures that consist of nodes (also called vertices) and edges (also called links). In a graph representation of a network.
- Which can be created as Knowledge Graphs,and the can be directed and Undirected

Network Representation

- Undirected graphs: In an undirected graph, edges do not have a direction and represent a symmetric relationship between two nodes.
  Eg: Karate Club Network,Drug-Gene Interaction

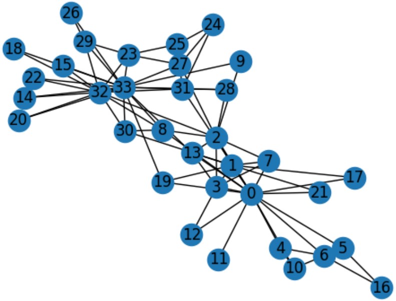


Figure: Karate club network

Network Representation

- Directed graphs: In a directed graph, edges have a direction and represent an asymmetric relationship between two nodes. That is, if node A is connected to node B, it does not necessarily mean that node B is connected to node A. Directed graphs are often used to represent information flow networks, web networks, and biological networks.

Network Representation

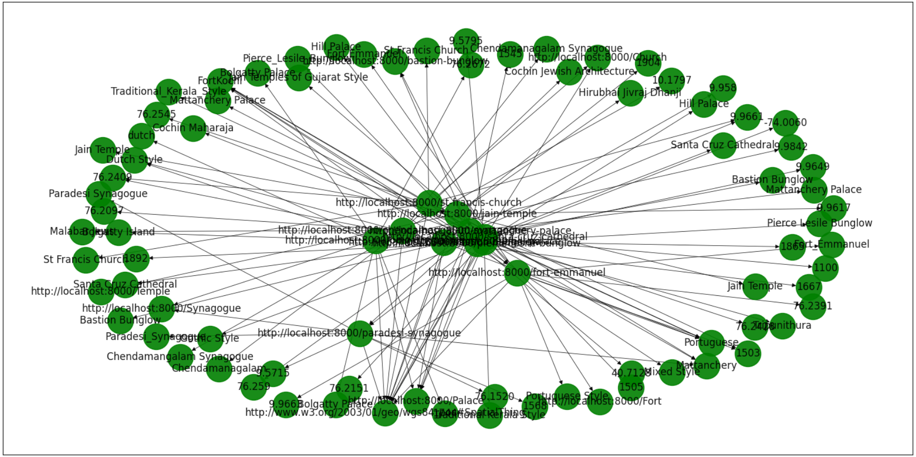


Figure: Tourism Knowledge Graph

Feature Extraction

Feature extraction in knowledge graphs is the process of representing node characteristics as a set of features. This can be done using different methods depending on the specific use case and available data.

1. Structure-based Similarity Index
2. Node-based Similarity Measure
3. Path based Simliliarity Measure

Structure-based Similarity Index

- **Jaccard coefficient:** Jaccard coefficient gives the normalized score to the common neighbor’ s predicted score.

$$LPJC(x,y) = \frac{|N(x) \cap N(y)|}{|N(x) \cup N(y)|} \tag{1}$$

- **Adamic Adar Index:** In Adamic Adar, the node pairs are assigned with a high score

$$LPAA(x,y) = \sum_{z \in N(x) \cap N(y)} \frac{1}{\log(N(z))} \tag{2}$$

Structure-based Similarity Index

- **Katz Index:** It has been observed that the chances of establishing links between nodes having higher degree is more compared with pair of nodes having smaller degree.

$$LPPA(x, y) = N(x) \times N(y) \tag{3}$$

Node-based Similarity Measure

Node-based similarity measures are used in knowledge graph analysis to measure similarity between nodes on their attributes, properties or features.

- **Cosine Similarity:** The cosine similarity measures the similarity between two nodes based on their feature vectors. It is calculated as the dot product of the feature vectors divided by the product of their Euclidean norms.

$$\text{cosine similarity}(x, y) = \frac{\mathrm{x} \cdot \mathrm{y}}{\|\mathrm{x}\| \, \|\mathrm{y}\|} \tag{4}$$

Node-based Similarity Measure

- **Euclidean Distance:** The Euclidean distance measures the dissimilarity between two nodes based on the distance between their feature vectors in Euclidean space.

$$\text{Euclidean distance}(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{5}$$

Path based Simliliarity Measure

Path-based similarity measures are algorithms used in knowledge graph analysis to measure the similarity between nodes based on the paths connecting them in the graph.

- **Katz Measure:** Useful measure to quantify hidden links between two nodes, which is based on the number of paths between the nodes.

$$LPKZ(x,y) = \sum_{i=1}^{\infty} \beta^i |path_i(x,y)| \tag{6}$$

Path based Simliliarity Measure

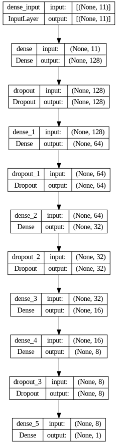
- **Sim-Rank:** This measure considers the similarity score between the neighboring nodes. In this measure, the probability of establishing links for a node pair is high, if they are having more number of similar neighbors.

$$LPSR(x,y) = \gamma \frac{\sum_{a \in N(x)} \sum_{b \in N(y)} LPSR(a,b)}{|N(x)||N(y)|} \tag{7}$$

Deep Neural Network

- A deep neural network is used to predict links in a graph by training on a feature matrix representing the graph and predicting the presence of links between nodes.
- The model uses multiple hidden layers to transform the input data and produce an output.

Architecture



dense_input | input: | [(None, 11)]
InputLayer | output: | [(None, 11)]

dense | input: | (None, 11)
Dense | output: | (None, 128)

dropout | input: | (None, 128)
Dropout | output: | (None, 128)

dense_1 | input: | (None, 128)
Dense | output: | (None, 64)

dropout_1 | input: | (None, 64)
Dropout | output: | (None, 64)

dense_2 | input: | (None, 64)
Dense | output: | (None, 32)

dropout_2 | input: | (None, 32)
Dropout | output: | (None, 32)

dense_3 | input: | (None, 32)
Dense | output: | (None, 16)

dense_4 | input: | (None, 16)
Dense | output: | (None, 8)

dropout_3 | input: | (None, 8)
Dropout | output: | (None, 8)

dense_5 | input: | (None, 8)
Dense | output: | (None, 1)

Figure: MLP Architecture

Architecture

- The model is a Multi-Layer Perceptron (MLP) neural network.
- It consists of five fully connected (Dense) layers of neurons
- The first layer has 128 neurons, the second layer has 64 neurons, the third layer has 32 neurons, the fourth layer has 16 neurons, and the fifth layer has 8 neurons.
- The input layer has a number of neurons equal to the number of features in the training data.

Architecture

- Hidden layers use the ReLU activation function, which is a popular choice for neural networks.
- The output layer has a single neuron with a sigmoid activation function, which is suitable for binary classification problems.
- Three Dropout layers have been added after the first, second and third dense layers, respectively, to reduce overfitting and improve generalization performance.

Activation Functions

- **ReLu:** It is a simple non-linear function that maps any negative input to zero and passes any positive input through unchanged.
  f(x) = max(0, x)
- **Sigmoid:** It maps any input to a value between 0 and 1, which can be interpreted as a probability.

$$f(x) = \frac{1}{1+e^{-x}} \tag{8}$$

Loss Function

- **Binary Cross Entropy:**Binary cross-entropy loss is a commonly used loss function in machine learning for binary classification problems. It is a measure of the difference between the predicted probabilities of a binary classification model and the true binary labels of the data.

$$L(y, y') = -(y \cdot \log(y') + (1 - y) \cdot \log(1 - y')) \tag{9}$$

Tools

Tools

1. **Google Colab**
2. **Neo4J**
3. **Python**
4. **PyTorch**
5. **TensorFlow**
6. **NetworkX**
7. **Stellargraph**

Implementation and Results

Experimental Setup

- Proposed Deep Learning model with standard link prediction algorithms on four different knowledge graphs.
- The feature extraction techniques used include node degree, clustering coefficient, and node centrality measures. Various normalization and scaling techniques are used to prepare the data for training the MLP models.
- Performance evaluation is done using metrics such as precision, recall, and F1 score.
- Experiments with different hyperparameters and optimization algorithms are conducted to determine the best configuration for each model.

Implementation

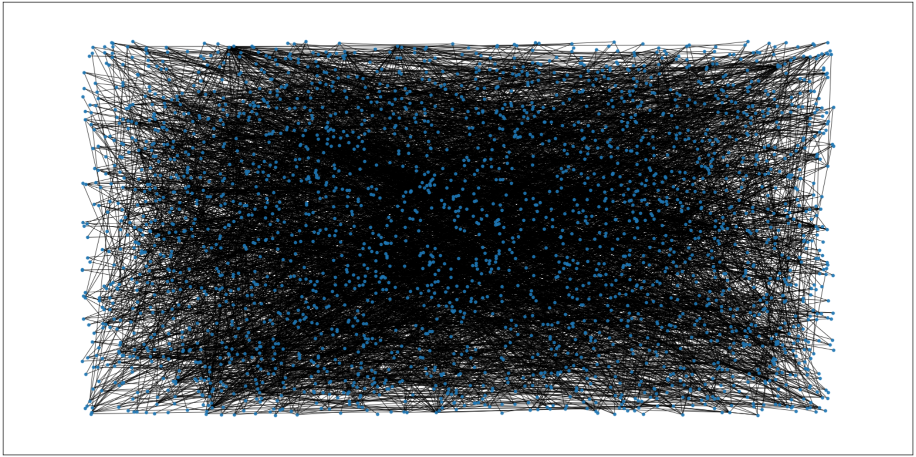


Figure: Training Subgraph

Implementation

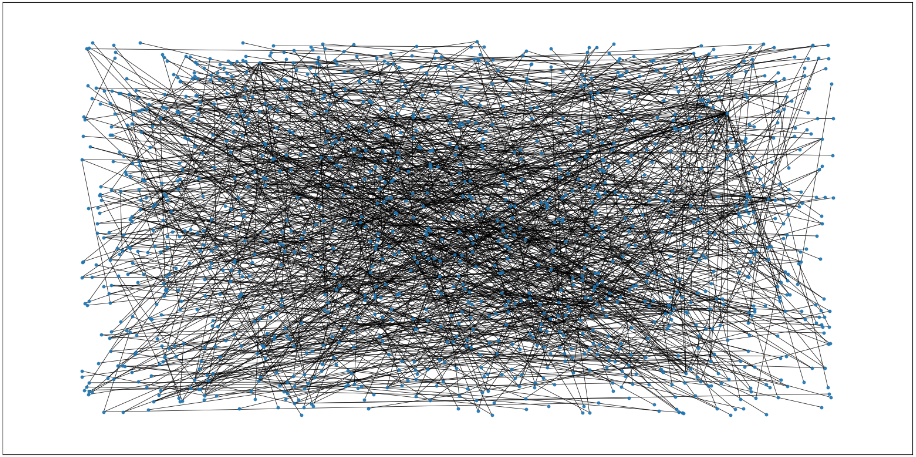
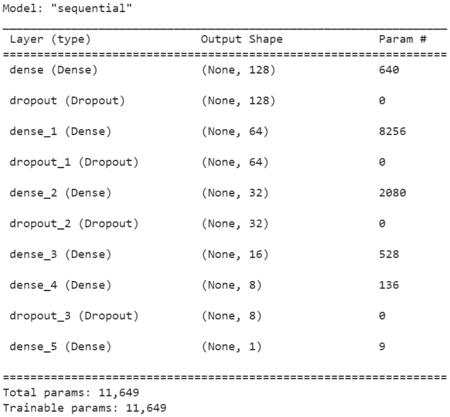


Figure: Testing Subgraph

Implementation



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense (Dense)               (None, 128)               640       
                                                                 
 dropout (Dropout)           (None, 128)               0         
                                                                 
 dense_1 (Dense)             (None, 64)                8256      
                                                                 
 dropout_1 (Dropout)         (None, 64)                0         
                                                                 
 dense_2 (Dense)             (None, 32)                2080      
                                                                 
 dropout_2 (Dropout)         (None, 32)                0         
                                                                 
 dense_3 (Dense)             (None, 16)                528       
                                                                 
 dense_4 (Dense)             (None, 8)                 136       
                                                                 
 dropout_3 (Dropout)         (None, 8)                 0         
                                                                 
 dense_5 (Dense)             (None, 1)                 9         
                                                                 
=================================================================
Total params: 11,649
Trainable params: 11,649
```

Figure: Implementation of model

Evaluation Metrics

Accuracy: The accuracy of the model is defined as the ratio of number of links that are actually predicted correctly to the total number of nonexistence links.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

Knowledge Graph	**Accuracy**
Wiki Data	0.9920
Tourism	0.9800
Cora	0.9987
Facebook	0.9100

Table: Accuracy of Knowledge Graphs

Evaluation Metrics

- **Precision:** Precision is the ratio of TP to the summation of TP and FP.
- **Recall:** Recall is the ratio of TP to the summation of TP and FN.

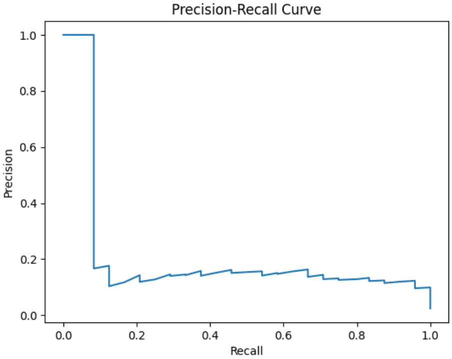


Figure: Precision Recall Curve

Evaluation Metrics

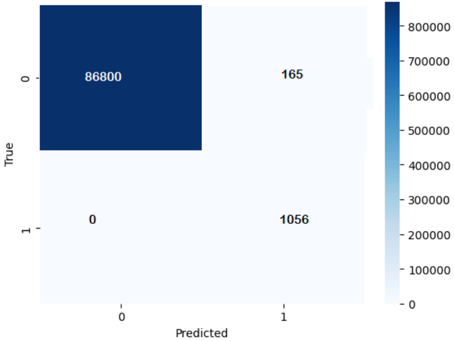


Figure: Confusion Matrix

Result

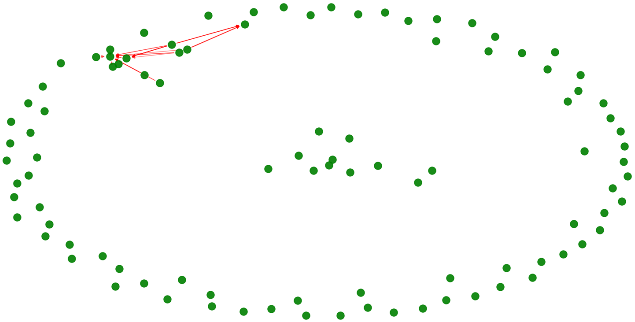


Figure: Predicted links

Conclusion

Conclusion

- Link prediction in complex networks is an emerging research domain in social network analysis.
- In this work I proposed a new framework for Inductive Link Prediction in Knowledge Graphs help us for the effective implementation of models to discover hidden groups or the absent relationships in the groups
- techniques in conjunction with Multi-Layer Perceptron for link prediction has the potential to improve performance.This work have achieved high accuracy.

References

References

1] Bordes, Antoine, et al. ”Translating embeddings for modeling multi-relational data. ” *Advances in neural information processing systems 26 (2016).*.

2] Elfaki, Abdelrahman, Amer Aljaedi, and Yucong Duan. ”Mapping ERD to knowledge graph.” 2019 IEEE World Congress on Services (SERVICES). Vol. 2642. IEEE, 2019.

3] Yang, Bishan, et al. ”Embedding entities and relations for learning and inference in knowledge bases.” arXiv preprint arXiv:1412.6575 .

4] Trouillon, Théo, et al. ”Complex embeddings for simple link prediction.” International conference on machine learning. PMLR.

References

5] Ding, Lianhong, and Shengchang Gao. ”Multi-Relational Graph Convolutional Network Based on Relational Correlation for Link Prediction.” 2021 2nd International Conference on Computer Science and Management Technology (ICCSMT). IEEE, 2021.

6] Zhang, Daokun, et al. ”Attributed network embedding via subspace discovery.” Data Mining and Knowledge Discovery 33.6 (2019): 1953-1980.

7] Bordes, Antoine, et al. ”Translating embeddings for modeling multi-relational data.” Advances in neural information processing systems 26 (2013).

8] McCoy, Kevin, et al. ”Biomedical text link prediction for drug discovery: a case study with COVID-19.” Pharmaceutics 13.6 (2021): 794.