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

Dissertation Submitted in partial fulfilment of the requirements of the degree of

Master of Technology

in

Computer Science and Engineering with specialization in Data Science and Artificial Intelligence

by

Sruthimol Rajesh

47921014

Under the guidance of

Dr. Shailesh S

DEPARTMENT OF COMPUTER SCIENCE

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

April 2023

DEPARTMENT OF COMPUTER SCIENCE
COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY
ERNAKULAM, KOCHI-682022

CERTIFICATE

This is to certify that the dissertation work entitled **Automatic Edge Prediction: An Application on Knowledge Graphs for Relationship Extraction** is a bonafide record of work carried out by Sruthimol Rajesh (47921014) submitted to the Department of Computer Science in partial fulfilment of the requirements for the award of the degree of Master of Technology in Computer Science and Engineering with specialization in Data Science and Artificial Intelligence at Cochin University of Science and Technology during the academic year 2023.

Project Guide	Head of the Department
Dr. Shailesh S	**Dr. Philip Samuel**
Assistant Professor	Professor
Dept. of Computer Science	Dept. of Computer Science
CUSAT	CUSAT

DATE : 25 April 2023

Office Seal

DECLARATION

I declare that the work presented in this dissertation titled **Automatic Edge Prediction: An Application on Knowledge Graphs for Relationship Extraction** represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Signature:

Name:	Sruthimol Rajesh	Date:	25 April 2023
Reg No:	47921014	Place:	Ernakulam

ACKNOWLEDGEMENT

Foremost, my sincere thanks to the Almighty for granted me the opportunity, the strength, and the guidance to complete this project.
I would like to express my sincere gratitude to my project guide **Dr.Shailesh S,** Assistant Professor,Department of Computer Science,for their invaluable guidance, support, and encouragement throughout the course of this project. Their expertise, insights, and constructive feedback have been instrumental in shaping my ideas and improving the quality of my work. I am deeply grateful for their patience, understanding, and willingness to go above and beyond to help me succeed. Thank you for being an exceptional mentor and for sharing your knowledge and experience with me.
I am honored to acknowledge the contributions of **Dr.Philip Samuel,** The Head of the Department,Assistant Professor,Department of Computer Science, for their unwavering support and guidance throughout my academic journey. Their visionary leadership, dedication, and commitment to excellence have set the standard for academic excellence at our institution. I am grateful for their encouragement, motivation, and inspiration, which have been instrumental in shaping my academic and personal growth. Thank you for your invaluable support and for creating an environment that fosters innovation, creativity, and learning.
Sruthimol Rajesh

ABSTRACT

The work presented here focuses on the automatic edge prediction in different applications on knowledge graphs for entity relationship extraction using a neural network model.

The extraction of entity relationships from a knowledge graph is a crucial task in natural language processing, information retrieval, and machine learning. This task involves identifying the semantic connections between different entities in the graph, which can be challenging due to the complexity and size of the graph. Traditional approaches to this problem involve manual annotation or rule-based techniques, which can be time-consuming and error-prone. However, recent advances in deep learning and neural networks have opened up new opportunities for automating this task.

The approach presented in this work focuses on using a neural network model to predict the presence of an edge between two nodes in the graph. The model is trained on a set of features extracted from the graph, including indegree, outdegree, degree centrality, betweenness centrality, union neighbors, and similarity measures. These features are carefully selected to capture different aspects of the graph structure and semantics, and are normalized to ensure that they are comparable across different graphs.

The neural network model used in this approach is a deep learning model with multiple hidden layers, which allows it to learn complex relationships between the input features and the target output. The model is trained on a large dataset of labeled examples, and achieves high accuracy in predicting the presence of edges in the graph. This approach has several advantages over traditional techniques, including its ability to handle large and complex graphs, its efficiency and scalability, and its ability to learn from data and improve over time.

Overall, this work presents a promising approach to automating the task of entity relationship extraction in a knowledge graph, which has important applications in many areas of artificial intelligence and natural language processing. By using a neural network model trained on graph features, this approach allows for efficient and accurate extraction of entity relationships, enabling more effective analysis and understanding of large and complex knowledge graphs.

Keywords: Knowledge Graph,neural network,entity relation extraction,Supervised learning.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years,the field of knowledge management has gained significant attention due to the rapid growth of data and the need to efficiently store and organize it. Knowledge graphs are one of the key developments in this field, which are structured representations of knowledge that allow the efficient storage and retrieval of information. These graphs are composed of nodes and edges, where nodes represent entities such as people, places, and concepts, and edges represent the relationships between these entities. The labels associated with nodes and edges capture the specific domain-specific meaning and context of the entity or relationship.

One of the benefits of knowledge graphs is that they can be used to integrate information from various sources, such as databases, social media, and the internet, into a single structured format. This enables better search and analysis of information, as well as the discovery of new insights and relationships. Moreover, knowledge graphs can be used in various applications such as recommendation systems, chatbots, and question answering systems.

Ontology is an abstract representation of the knowledge graph, which defines the concepts and relationships between them in a formal way. Ontologies are typically constructed using a set of rules and axioms, which ensure consistency and logical coherence of the knowledge representation. They provide a shared vocabulary for communicating knowledge across different systems and domains. Ontologies can be used to automatically infer new knowledge based on existing relationships and axioms, as well as to validate and correct existing knowledge. Therefore, ontology plays a critical role in the development and maintenance of knowledge graphs.

They are typically built using semantic web technologies and can be used to capture complex relationships between entities, events, and concepts. KGs provide a flexible way of representing large amounts of structured and unstructured information, which can be used for a wide range of applications, such as information retrieval, question answering,

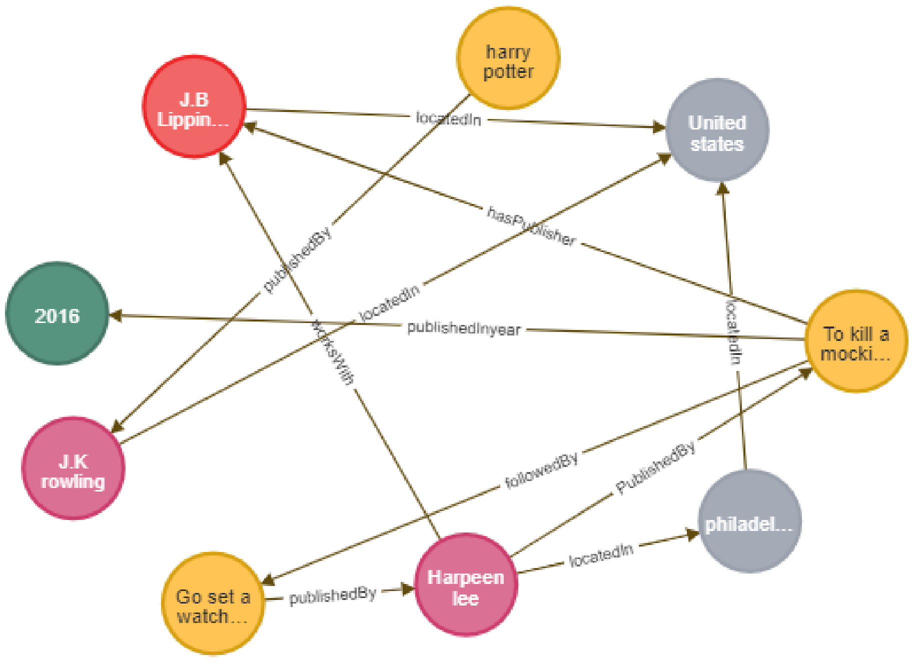


Figure 1.1: Knowledge Graph

and recommendation systems. One of the key challenges in KGs is predicting the existence of new edges, or links, between entities, which is known as link prediction. Link prediction is a fundamental problem in KGs that aims to identify the missing links between entities, which can be used to enrich the KG and improve its quality. Link prediction is a type of machine learning task that involves predicting the likelihood of a link between two entities. The entities could be anything, such as people, products, or even concepts. The prediction is based on existing links and attributes of the entities, such as their past interactions or shared characteristics. This task has many applications in different fields, such as e-commerce, drug discovery, and academic research.

One of the applications of link prediction is in e-commerce, where it can be used to recommend new items to users. By predicting the likelihood of a link between a user and an item, the system can suggest new items that the user might be interested in. This approach is commonly used in online marketplaces such as Amazon, where recommendations are based on the user's browsing and purchasing history.

Link prediction is also used in drug discovery to predict the interactions between drugs and their target proteins. By predicting these interactions, researchers can identify potential drug candidates and test their efficacy in vitro and in vivo. This approach can save time and resources compared to traditional methods of drug discovery, which can be time-consuming and expensive.

Finally, link prediction is also used in academic research to predict co-authorship relationships between researchers. By predicting these relationships, researchers can identify potential collaborators and build research networks. This can lead to new collaborations and research projects, which can advance scientific knowledge and innovation. Several ap-

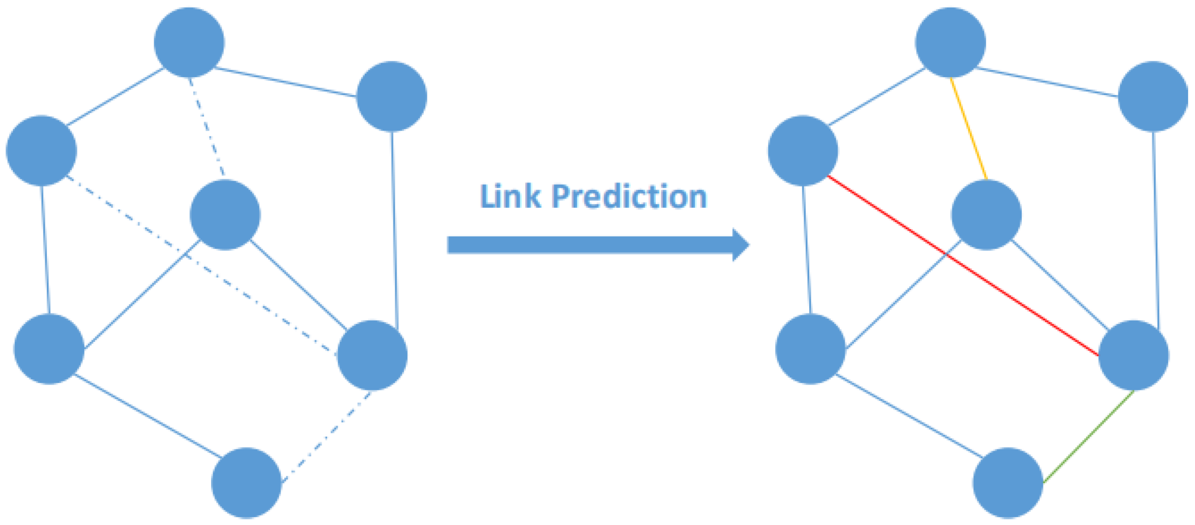


Figure 1.2: Link Prediction [1]

proaches have been proposed for link prediction in KGs, ranging from rule-based methods to graph based methods, and from similarity-based methods to embedding-based methods. Rule-based methods use handcrafted rules to identify patterns and regularities in the data, while graph-based methods rely on graph algorithms to identify the most important nodes and edges in the graph.Similarity based methods use measures of similarity to predict the likelihood of a link between two entities, while embedding-based methods learn low-dimensional representations of the entities and relations that capture the semantic and structural features of the graph.
In the context of knowledge graphs, link prediction refers to the task of predicting missing or potential relationships between entities in the graph. There are two common approaches to link prediction: Transductive and Inductive. Transductive link prediction assumes that the entire graph is available during the training phase, and the goal is to predict all the missing links in the graph. Transductive methods are based on the idea that the existing relationships in the graph contain enough information to make accurate predictions. The transductive approach involves partitioning the graph into training and test sets, where the training set contains all the known relationships, and the test set contains the missing links to be predicted. The model is then trained on the training set, and the missing links in the test set are predicted based on the learned patterns.

Inductive link prediction, on the other hand, involves predicting new links based on patterns learned from the existing relationships in the graph. Inductive methods are designed to make predictions for subsets of the graph without requiring the entire graph

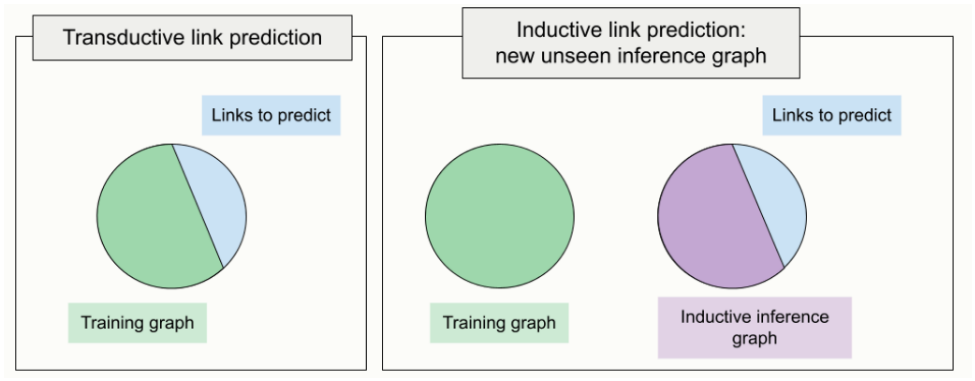

Transductive link prediction
Links to predict
Training graph
Inductive link prediction: new unseen inference graph
Links to predict
Training graph
Inductive inference graph

Figure 1.3: Transductive and Inductive Link Prediction [2]

to be available during the training phase. In inductive learning, the model is trained on a subset of the graph, and the goal is to learn a function that can generalize to new data. The inductive approach involves using features that describe the entities and their relationships to learn a function that can predict missing links based on the learned patterns.

The choice between transductive and inductive learning depends on the specific problem and the available data. Transductive methods are useful when the goal is to predict all the missing links in the graph, and the entire graph is available during the training phase. Inductive methods are more flexible and can be used to predict missing links for subsets of the graph, making them useful when the graph is large or the goal is to predict a specific subset of missing links. However, inductive methods may require additional features or metadata to be collected to make accurate predictions, which can be challenging in some domains.

Both transductive and inductive link prediction approaches have been widely used in the field of knowledge graph completion. Transductive methods such as matrix factorization, random walk-based methods, and neural network based methods have been used to predict all the missing links in the graph. Inductive methods such as graph convolutional networks (GCNs), graph attention networks (GATs), and graph neural networks (GNNs) have been used to predict links for specific subsets of the graph. The main goal is to predict the links for specific subsets of the graph and here the entire graph is not available during training.

link prediction is a fundamental problem in KGs that has many applications in various domains. In recent years, several approaches have been proposed for link prediction, ranging from rule-based methods to embedding-based methods. Despite the progress made in link prediction, there are still several challenges that need to be addressed, such as the sparsity of the data, the scalability of the algorithms, and the interpretability of the results. In this work,provide a comprehensive review of the state-of-the-art methods for link prediction and present a set of experiments that compare the performance of different methods on a range of datasets.

1.1 Motivation

Link prediction is a crucial task in knowledge graph completion, which involves predicting missing links between entities in a knowledge graph. There are several motivations for performing link prediction in knowledge graphs, 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 theoryincluding: Incompleteness of knowledge graphs: Knowledge graphs are often incomplete, meaning that there are missing links between entities. Link prediction can help fill in these missing links and provide a more comprehensive view of the relationships between entities.

Improved accuracy of knowledge graphs: Link prediction can help improve the accuracy of knowledge graphs by identifying and correcting errors in existing links, as well as suggesting new links that were previously unknown.

Enhanced entity search and recommendation: Link prediction can help enhance entity search and recommendation by identifying related entities that may be of interest to users. This can improve the overall user experience and provide more relevant results.

Better understanding of complex relationships: Link prediction can help uncover complex relationships between entities in a knowledge graph that may not be immediately apparent. This can lead to new insights and discoveries in various fields, such as biology, finance, and social sciences.

Applications in various fields: Link prediction has many practical applications in various fields, such as social network analysis, recommendation systems, and fraud detection. By accurately predicting links between entities, these applications can be more effective and efficient.

Overall, link prediction is a valuable tool for improving the accuracy and completeness of knowledge graphs, as well as for enhancing entity search and recommendation, understanding complex relationships, and enabling applications in various fields.

1.2 Problem Statement

To extract the relationship between nodes and predicts the missing links in knowledge graphs in complex networks To extract the relationship between nodes and predict the missing links in knowledge graphs in complex networks, we can use a variety of methods and techniques from the field of network analysis and machine learning.

In recent years, there has been an explosion in the amount of data available to us, and with this increase, there has been a need for tools to help us make sense of this data. One area where this is particularly challenging is in knowledge graphs in complex networks, where we need to extract the relationship between nodes and predict missing links. In this problem statement, we will explore the methods and techniques used in network analysis and machine learning to solve this problem.

Knowledge graphs are a type of graph database that stores information about entities and their relationships. They are commonly used in a wide range of applications, such as recommendation systems, search engines, and fraud detection. The graph structure of knowledge graphs allows us to represent complex relationships between entities, but it also creates challenges when it comes to extracting relationships and predicting missing links.

One approach to extracting relationships between nodes in a knowledge graph is to use network analysis techniques. Network analysis involves the study of the structure and behavior of networks, and it can be used to identify important nodes and relationships in a graph. For example, degree centrality is a measure of the number of edges connected to a node, and it can be used to identify nodes that are highly connected in the graph. Betweenness centrality is a measure of the number of shortest paths that pass through a node, and it can be used to identify nodes that are important for information flow in the graph.

Once we have identified important nodes and relationships in the graph, we can use machine learning techniques to predict missing links. Feature extraction is a common approach in machine learning that involves identifying relevant features from the nodes and edges in the network. For example, we can extract features such as the similarity of the nodes' attributes, the frequency of co-occurrence between pairs of nodes, or the length of the shortest path between nodes.

Once we have extracted relevant features from the graph, we can use classification models such as logistic regression, decision trees, or neural networks to predict the likelihood of a missing link between a pair of nodes. These models can be trained on a subset of the graph, where the known relationships are used as the training set, and the missing links are used as the test set.

Evaluating the performance of these models is an important step in this problem statement. Metrics such as precision, recall, and F1 score can be used to measure the accuracy of the model's predictions. These metrics can help us understand how well the model is able to predict missing links in the graph and identify areas where improvements can be made.

extracting relationships between nodes and predicting missing links in knowledge graphs in complex networks is a challenging problem that requires a combination of network analysis and machine learning techniques. By identifying important nodes and relationships in the graph and using relevant features to train classification models, we can make accurate predictions about missing links in the graph. This can help us improve our understanding of the relationships between entities in the graph and enable us to make more informed decisions in a wide range of applications.

1.2.1 Objectives

The main objective of link prediction using feature extraction and MLP (Multilayer Perceptron) is to predict the likelihood of a link forming between two entities in a network based on their attributes and the existing network structure.

Here are some specific objectives of this approach:

- Feature extraction: Extracting relevant features from nodes in a network to represent them in a machine-readable format. This involves identifying which features are relevant for link prediction and how to represent them mathematically.

- MLP training: Training an MLP algorithm to learn the patterns in the extracted features and predict the likelihood of a link forming between two entities. This involves defining the architecture of the MLP, selecting appropriate activation functions, and optimizing the parameters using training data.

- Evaluating link prediction performance: Evaluating the performance of the MLP algorithm in predicting links between entities in the network. This involves using metrics such as precision, recall, and F1 score to assess the accuracy of the predictions

- Improving link prediction accuracy: Identifying factors that can improve the accuracy of link prediction using feature extraction and MLP, such as selecting the right set of features, optimizing the MLP architecture and parameters, and incorporating additional information from the network.

Overall, the main objective of link prediction using feature extraction and MLP is to develop a robust and accurate method for predicting links in a network, which can be useful in a wide range of applications such as social network analysis, recommender systems, and bioinformatics.

Problem Statement

1.3 Major Contribution of the Dissertation

The major contribution of a link prediction work using MLP (Multilayer Perceptron) is to develop an accurate and effective method for predicting missing links in a network using machine learning. Specifically, the major contributions of such work can include:

- Developing a novel approach: The use of MLP for link prediction is a novel approach that can provide new insights into the underlying network structure and improve the accuracy of link prediction.
- Feature engineering: MLP-based link prediction methods typically involve extracting relevant features from the nodes and edges in the network to represent them mathematically. A significant contribution of such work can be in identifying which features are relevant for link prediction and how to represent them in a machine-readable format.
- Model training: The process of training an MLP algorithm involves optimizing the model's parameters using training data. A major contribution of such work can be in defining the architecture of the MLP, selecting appropriate activation functions, and optimizing the parameters to achieve the best possible performance.
- Evaluation of link prediction performance: A significant contribution of such work can be in evaluating the performance of the MLP-based link prediction approach using metrics such as precision, recall, and F1 score. This can provide insights into the accuracy of the predictions and help to identify areas for improvement.
- Improving link prediction accuracy: Another significant contribution of such work can be in identifying factors that can improve the accuracy of link prediction using MLP, such as selecting the right set of features, optimizing the MLP architecture and parameters, and incorporating additional information from the network.

1.4 Thesis Outline

The rest of the thesis organised as follows. Chapter 2 provides the background section of thesis and provides context for the research topic and explains why the research is important. This section typically includes a brief overview of the history of the Link Prediction.Chapter 3 provides an overview of the existing literature in the field of Knowledge Graphs in different applications.Additionally it will introduce the formulation of Deep Neural Network based Link Prediction.Chapter 4 presents an outline of the proposed work and Architecture. Chapter 5 contains the results and discussion of the system. Finally , conclusions and future works are discussed in Chapter 6.

Chapter 2

Background

2.1 Ontology

Ontologies are structured frameworks that represent knowledge about a particular domain of interest. They provide a shared understanding of the concepts, relationships, and rules within a specific field of study, enabling better communication and collaboration among stakeholders. Ontologies are used in various applications, including information retrieval, natural language processing, semantic web, and knowledge management.

An ontology consists of a set of concepts, properties, and relationships that define the structure of the domain. Concepts are the basic building blocks of an ontology, representing the entities or things that exist in the domain. Properties describe the attributes or characteristics of the concepts, such as color, size, weight, and shape. Relationships define the connections between the concepts, such as is-a, part-of, has-a, and contains. Together, these components provide a formal representation of the domain that can be used to reason, query, and infer knowledge.

Ontologies are generalized data models that capture the commonalities and variations within a domain, rather than specific instances. They are designed to be reusable and extensible, allowing new concepts and relationships to be added as the domain evolves. Ontologies provide a flexible and scalable approach to modeling knowledge, enabling interoperability and integration across different systems and applications. However, building an ontology requires a deep understanding of the domain and careful consideration of the trade-offs between generality and specificity, expressiveness and simplicity, and precision and ambiguity.

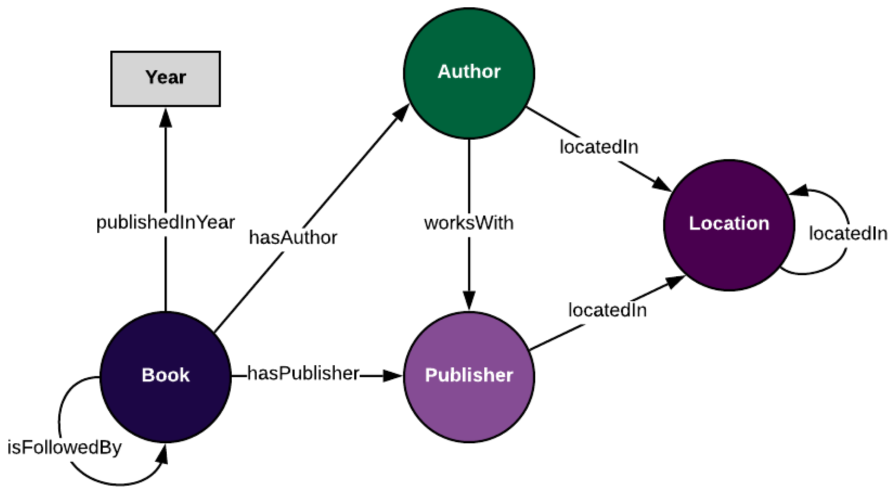


Figure 2.1: Ontology [3]

2.2 Knowledge Graph

Building on the ontology as a framework, a knowledge graph provides a way to represent real data as a web of interconnected entities and their attributes. By representing data in this way, we can uncover hidden relationships and patterns, and discover new insights that may not be apparent when data is viewed in isolation.

For example, in a book publishing domain, a knowledge graph could connect authors, books, publishers, and locations based on various relationships. For instance, an author could be linked to a book through authorship, and a book could be linked to a publisher through publishing. Similarly, a location could be linked to a publisher based on its headquarters or to a book based on the setting of the story.

Using a knowledge graph, we can explore various aspects of the domain, such as the distribution of books across locations, the popularity of authors, the influence of publishers on certain genres, and so on. We can also apply machine learning algorithms to the graph to discover patterns and predict future trends.

Overall, a knowledge graph is a powerful way to represent and analyze complex data in a structured and meaningful way, allowing us to make better-informed decisions and gain new insights into our domain.
ontology + data = knowledge graph

2.3 Link Prediction

Link prediction is a task in network analysis that aims to predict the presence or absence of edges (or links) between nodes in a network. This task is often used in various fields,

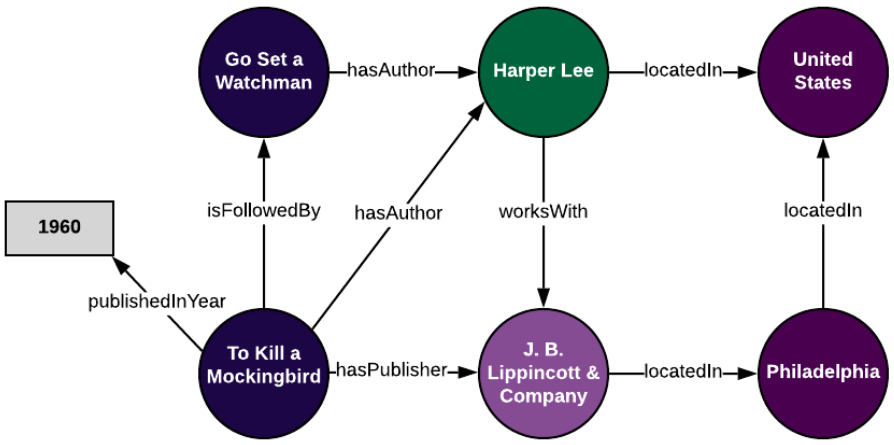


Figure 2.2: Knowledge Graph [3]

including social network analysis, recommender systems, biological networks, and knowledge graphs. The goal of link prediction is to identify missing or potential connections in the network, which can be useful for various purposes, such as improving network performance, identifying missing data, and understanding the underlying structure and dynamics of the network.

There are several approaches to link prediction, including heuristic-based methods, graph-based methods, and machine learning-based methods. Heuristic-based methods often rely on simple rules or metrics to estimate the likelihood of a link between two nodes, such as the number of common neighbors or the Jaccard coefficient. Graph-based methods often use random walks or matrix factorization techniques to capture the network structure and identify missing links. Machine learning-based methods often use features extracted from the network topology or node attributes, and train a predictive model to predict the presence or absence of edges.

Overall, link prediction is an important task in network analysis, with various applications in different domains. The choice of method depends on the specific problem and data characteristics, and often requires careful evaluation and validation to ensure its effectiveness and usefulness.

2.4 Neural Network

A neural network is a type of machine learning algorithm that is modeled after the structure and function of the human brain. It consists of a large number of interconnected processing nodes, or neurons, organized into layers.

Each neuron is designed to receive one or more inputs, perform some computations on those inputs, and generate an output. The output of one neuron can then be used as input for other neurons in the network, forming a complex web of interconnected nodes.

The network learns to perform a specific task by adjusting the strength of the connections between the neurons based on examples of inputs and outputs provided during the training phase. This process is called backpropagation, where the error between the actual output and the expected output is propagated back through the network to adjust the weights of the connections.

2.5 Multilayer Perceptron

A multilayer perceptron (MLP) is a type of neural network that consists of three or more layers of neurons: an input layer, one or more hidden layers, and an output layer.

In an MLP, each neuron in a layer is connected to every neuron in the following layer. Each connection has an associated weight, which determines the strength of the connection. During training, the weights are adjusted so that the network can learn to map inputs to outputs.

MLPs are widely used for supervised learning problems such as classification and regression. They are especially effective for problems with non-linear decision boundaries, where simple linear models such as logistic regression or linear regression would not be able to capture the complex patterns in the data.

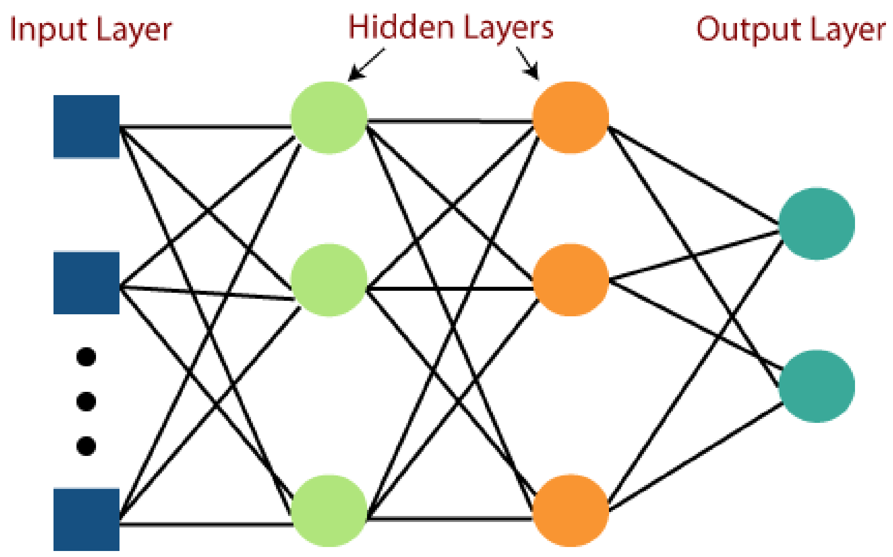


Figure 2.3: Multilayer Perceptron [4]

2.6 Graph Neural Network

Graph neural network (GNN) is a type of neural network that is designed to work with graph data structures, where nodes represent entities, and edges represent relationships between those entities. GNNs are particularly useful for analyzing data that has complex relationships, such as social networks, chemical compounds, or knowledge graphs.

GNNs work by propagating information through the graph structure, where each node in the graph updates its representation based on the representations of its neighboring nodes. This process is repeated multiple times, with each iteration refining the node representations by incorporating more information from the graph. In this way, GNNs are able to capture the complex relationships between entities in the graph and use this information to make predictions or perform other tasks.

Overall, GNNs provide a powerful tool for analyzing and making predictions on graph data, allowing us to leverage the rich relationships between entities to gain insights and solve complex problems.

2.7 Node Classification

Node classification is a task in machine learning that involves assigning a label or category to a node in a graph. The graph can represent a variety of structures, such as a social network, a biological network, or a knowledge graph. The goal of node classification is to predict the label of a node based on its features and the labels of its neighboring nodes.

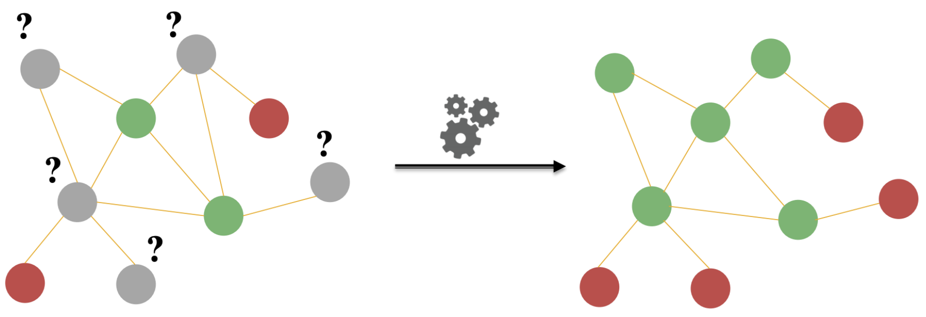


Figure 2.4: Node Classification [5]

In graph-based machine learning, node classification is a fundamental problem that has many applications. For example, it can be used to predict the function of a gene in a biological network, the sentiment of a user in a social network, or the category of a product in a knowledge graph.

2.8 Graph Embedding

Graph embedding refers to the process of mapping the nodes in a graph to low-dimensional vectors, while preserving the graph structure and node features. The resulting vector representations can be used for various tasks such as node classification, link prediction, and clustering.

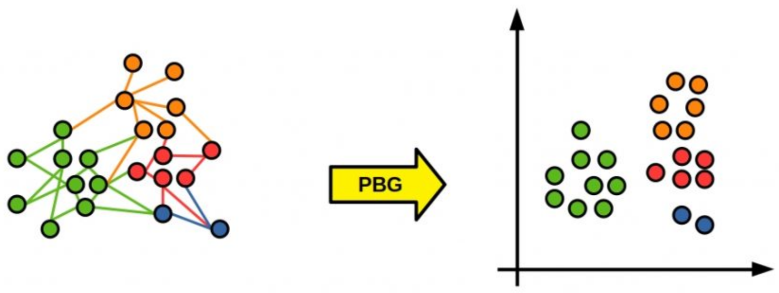


Figure 2.5: [6]

The goal of graph embedding is to capture the complex relationships and dependencies between nodes in a graph, and represent them in a way that is amenable to machine learning algorithms. Traditional approaches to graph analysis and machine learning often rely on explicit feature engineering and graph algorithms, which can be time-consuming and computationally expensive. Graph embedding, on the other hand, provides a way to automatically learn meaningful representations of nodes in a graph, which can then be used for downstream tasks.

There are various methods for graph embedding, including random walk-based methods, matrix factorization-based methods, and deep learning-based methods.
Random walk-based methods involve simulating random walks on the graph and using the resulting node sequences to learn the node embeddings. Matrix factorization-based methods involve factorizing the graph adjacency matrix into low-rank matrices, and using the resulting factors to learn the node embeddings. Deep learning-based methods involve using neural networks to learn the node embeddings directly from the graph structure and node features.

Chapter 3

Literature Review

Knowledge graphs have been a hot research topic in the field of artificial intelligence and machine learning over the past few years. Recent trends on knowledge graphs are the Knowledge graph embeddings,incorporating uncertainty,integration with natural language processing,developing interoperable knowledge graph and methods for scaling knowledge graphs.
Knowledge graph embedding is an area of research that learns low dimensional representations of entities and relations.The previous methds are the transductive link prediction and inductive link prediction.Transductive link predcition is a type of link prediction that aims to predict the presence or absence of links between a given set of nodes in a network.In transductive link prediction, the algorithm is given a set of nodes in the network, and it predicts the links that exist among them. This is typically done by partitioning the network into a training set and a test set. The algorithm is trained on the training set, which contains a subset of the nodes and their links, and then it is evaluated on the test set, which contains the remaining nodes and their links.There are many transductive link prediction methods such as
TransE (Bordes et al., 2013) model is athe first simple and efficient embedding method that represents relations as translations between entity embeddings. It has been widely used as a baseline in many subsequent worksMoreover TransE[7][8] model DistMult (Yang et al., 2014): has the better perfomance that is the bilinear model that performs a multiplicative interaction between entity and relation embeddings [9][10].ComplEx (Trouillon et al., 2016): ComplEx is an extension of DistMult that models both symmetric and asymmetric relations by using complex-valued embeddings. It has been shown to be effective in capturing more complex relations in knowledge graphs [11].There is convolutional neural network model that represents entities and relations in a 2D matrix form and performs convolutional operations to capture the interaction between them [12][13]. RotatE is a method that represents relations as rotations in the complex plane. [14]. TuckER is a

tensor factorization-based method that uses a bilinear interaction model and decomposes the knowledge graph into a tensor form [15]. KG-BERT is a method that integrates BERT (a widely used pre-trained language model) with knowledge graph embedding. It has achieved state-of-the-art performance on some knowledge graph-related NLP tasks. [16][17][18]. HyperKG is a method that learns hyperbolic embeddings of entities and relations.It can model hierarchical structures in knowledge graphs[19][20].

Yuan et al. have proposed a graph kernel-based method to infer the future links in signed social network. They believe that signed struc- tural information in a network plays a major role in social evolution. In their work, they have generated many subgraphs for each user, which depend on the strength of relationships and then compute the similarity using Bhattacharjya kernel. Finally, they have used SVM classifier to predict future associations. Yaghi et al. [21] have presented a model based on an evolutionary neural network for predicting future links.
They have focused on addressing class imbalanced distribution by using the random and undersampling method[22]. GraphSAGE, a general inductive framework that leverages node feature information to efficiently generate node embeddings for previously unseen data. Instead of training individual embeddings for each node, we learn a function that generates embeddings by sampling and aggregating features from a node's local neighborhood.[23] The link prediction problem is found to be one of the emerging research directions in recent years. The first encouraging research work in this area was carried out by Liben-Nowell and Kleinberg, where they observed that the topological information is highly effective for predicting future rela- tionships in the network. Their work emphasizes on modeling the network evolution by leveraging the topological information in the network. They have proposed various similarity measures based on node neighborhood and graph distances for link prediction. A trade-off between accu- racy and computation cost is associated with each of these similarity measures, which are based on proximity evaluation methods. Some of the elegant works in the area of link prediction are presented in Table 1. Newman has proposed a method in which links are predicted by covering all the paths of restricted length based on small-world hypothesis.[24] The idea behind the hypothesis is that one person can be reached to another person through several paths of different lengths and two persons in any large network can be connected through a small number of acquaintances. [25][26]

A small world network follows power-law distribution in terms of both degree and edge. It is found to be much popular in social network anal- ysis as it always resembles the real-world networks, which are scale free in nature.
Machine learning has become a center of attraction in various tasks of classification and

prediction. In the area of links prediction, most of the works based on the similarity measures for the node pairs, which are found to be unsupervised methods. The unsupervised methods can be extended to supervised binary classification in the area of link prediction, where the links are being labeled at the time of data set prepa- ration. Liben-Nowell and Klenberg[27] have presented a path-based link prediction algorithm where the probability of the appearance of links depends on the length and number of the shortest path between pair of nodes. They have also presented an extensive comparison between node-based and path-based similarity measures. Since then, this article has become a benchmark for researchers who are working in the field of link prediction.

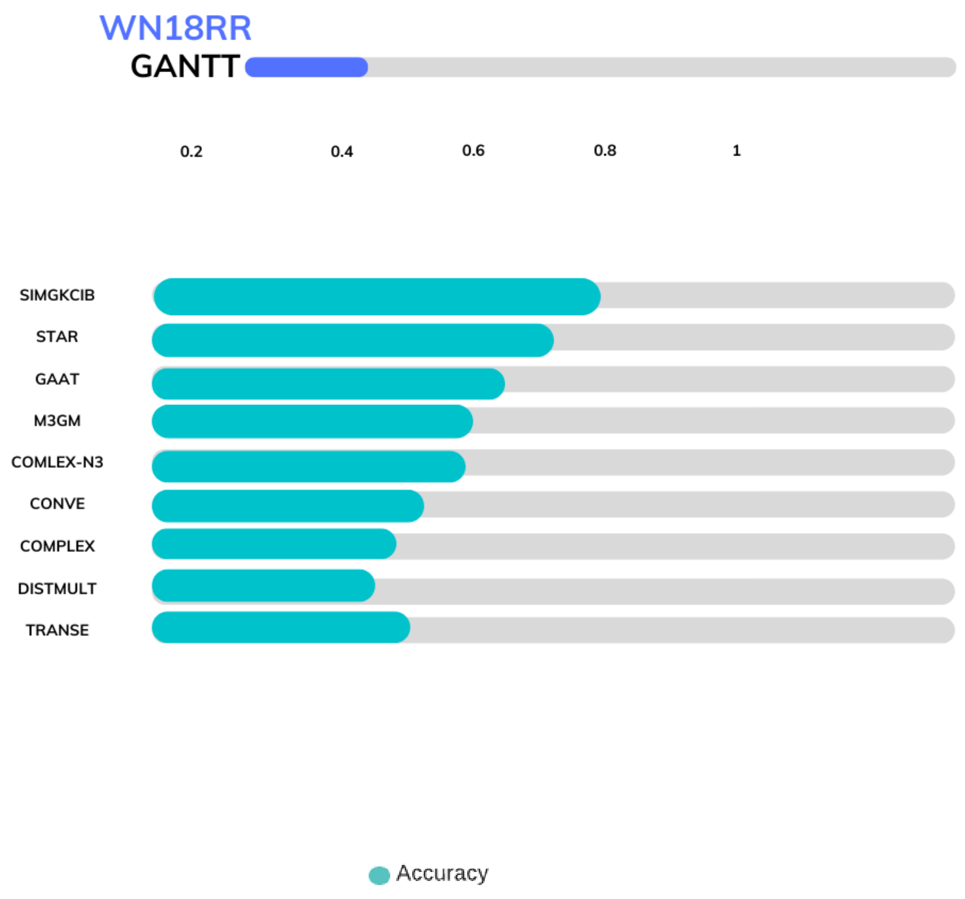


Figure 3.1: Research gaps

from this research gaps there is a lot of works based on mathematical and rule based modelling,the task of these modelling is very complicated and accuracy is not well as much improved. As from this works link prediction in social networks and also other fields,Link prediction is an important task in graph analysis and has wide-ranging applications in social network analysis, recommender systems, biological networks, and more. Machine learning methods have been applied to this task for many years, with varying degrees of success. However, recent advances in deep learning and graph neural networks (GNNs) have shown great promise in improving link prediction accuracy and scalability[28].

GNNs are a class of neural networks designed specifically for processing graph-structured data. They have been successfully applied to various graph-based tasks such as node classification, graph classification, and link prediction[29].GNNs can capture complex structural information of the graph and can be used to model both local and global dependencies, making them well-suited for link prediction.[30][31]
Deep learning techniques, such as deep neural networks and convolutional neural networks, have also been applied to link prediction. These methods can automatically learn features from the input data and have shown promising results in several link prediction benchmarks. Deep learning methods can also be combined with GNNs to leverage their strengths and improve performance.

The use of deep learning and GNNs in link prediction has the potential to improve many application systems. For example, in social networks, accurate link prediction can help recommend new friends or suggest potential connections for business purposes. In biological networks, link prediction can aid in identifying potential protein-protein interactions, which can lead to the discovery of new drugs.

finally,the need for deep learning and GNNs in link prediction is driven by their ability to model complex graph structures and improve prediction accuracy. Their potential applications are vast and have the potential to transform many fields, including social networks, recommender systems, and biological networks.

Author	Year	Technique Adopted	Features Used	Proposed Method	Performance Metrics
Zhang et al.	2021	Graph neural network (GNN)	Node features and graph topology	Multi-scale Graph Neural Network (MS-GNN)	AUC, AP, F1-score
Sun et al.	2021	Deep learning	Node and edge features	Graph neural network with edge attention mechanism	AUC, Precision, Recall
Wu et al.	2021	Network embedding	Node and edge features	Multi-task Network Embedding (MTNE)	AUC, AP, F1-score
Jiang et al.	2020	Graph convolutional network (GCN)	Node and edge features	Edge Weighted Graph Convolutional Network (EW-GCN)	AUC, AP, F1-score
Zhang et al.	2020	Deep learning	Node and edge features	Hierarchical Attention Graph Convolutional Network (HAGCN)	AUC, Precision, Recall
Wang et al.	2020	Graph neural network (GNN)	Node and edge features	Relation-aware Graph Convolutional Network (RGCN)	AUC, AP, F1-score
Liu et al.	2019	Deep learning	Node and edge features	Attention-based Graph Convolutional Network (AGCN)	AUC, Precision, Recall
Xu et al.	2019	Network embedding	Node and edge features	Structural Deep Network Embedding (SDNE)	AUC, AP, F1-score

Table 3.1: Recent works in link prediction

Chapter 4

Methodology

4.1 Proposed Method

This method is a deep learning-based model for automatic entity relationship extraction in knowledge graphs. It utilizes a graph neural network to learn the semantic representation of entities and their relationships. The system takes as input a set of entities and their corresponding relations and predicts new relationships that might exist among these entities.

The model extracts features from the knowledge graph, such as the type of entities, the relationship type, and the directionality of the relationship, and utilizes these features to construct the graph neural network. The network is trained using a binary classification approach to predict whether a relationship exists between a pair of entities or not.

The proposed system has several advantages, including its ability to handle complex, large-scale knowledge graphs, and its ability to extract relationships between entities that are not explicitly stated in the knowledge graph. This system has the potential to revolutionize the way we extract and utilize knowledge from large-scale knowledge graphs.

4.2 Graphs

Graphs are a type of mathematical structure used to represent relationships between objects or entities. A graph consists of a set of vertices, also known as nodes or points, and a set of edges, which connect pairs of vertices.Graphs are used to represent the complex networks

4.2.1 Undirected Graphs

In an undirected graph, the edges do not have a specific direction. This means that the relationship between the two vertices connected by an edge is symmetrical, and either vertex can be reached from the other.
Eg:Karateclub Graph, Drug-Gene Interaction Database (DGIdb), Gene Ontology (GO)

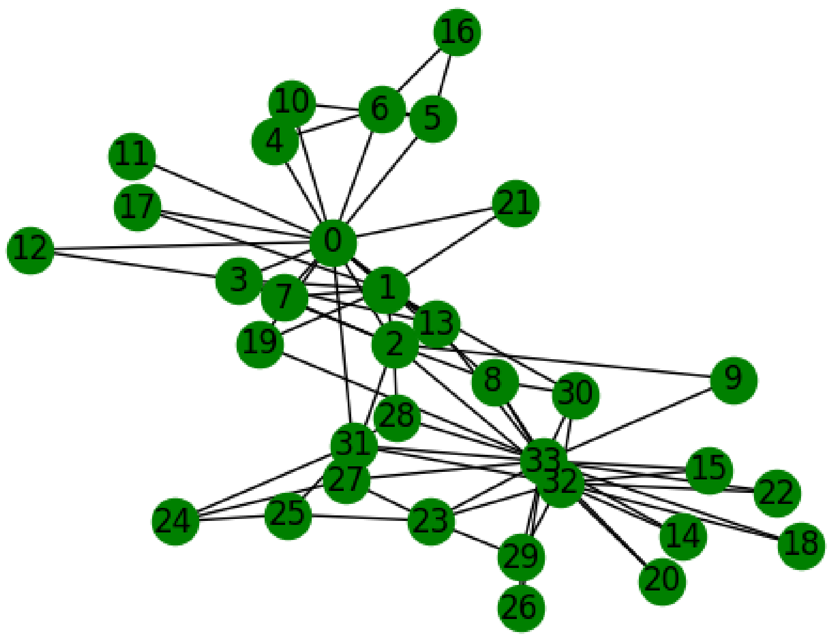


Figure 4.1: Karate club network

4.2.2 Directed Graphs

In a directed graph, also known as a digraph, the edges have a specific direction. This means that there is a one-way relationship between the two vertices connected by an edge, and only the vertex at the tail of the arrow can be reached from the vertex at the head of the arrow.

Directed graphs are useful for modeling systems where there is a clear flow of information or influence, such as social networks, transportation networks, or communication networks. Undirected graphs, on the other hand, are useful for modeling systems where the relationships between entities are symmetric, such as in a network of friendships or a system of roads. Eg:Drug-Gene Interactions,Social Networks

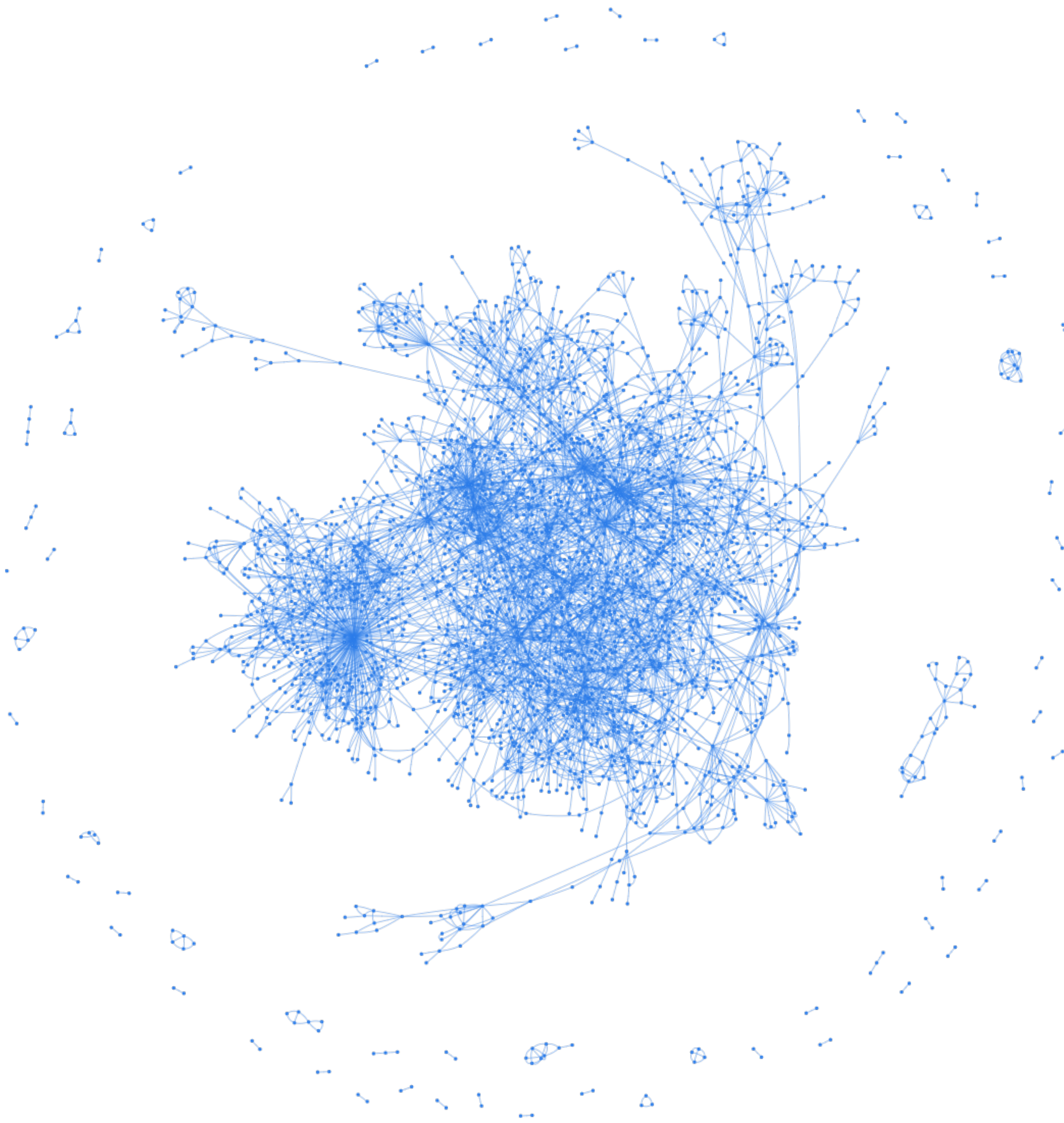


Figure 4.2: Cora citation

4.3 Computational Framework

4.3.1 Dataset

datasets were selected to represent a diverse range of networks and knowledge graphs, including social networks, drug-gene interactions, and tourism information. For each

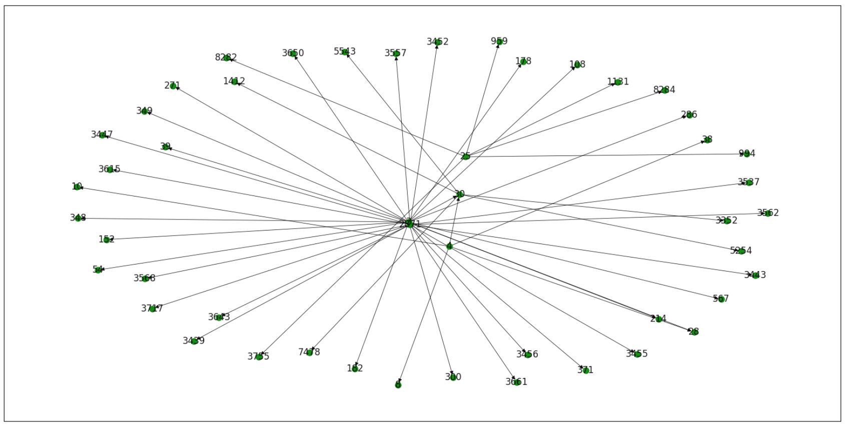


Figure 4.3: Drug-Gene Interaction
(small portion of graph from drug-gene interactions)

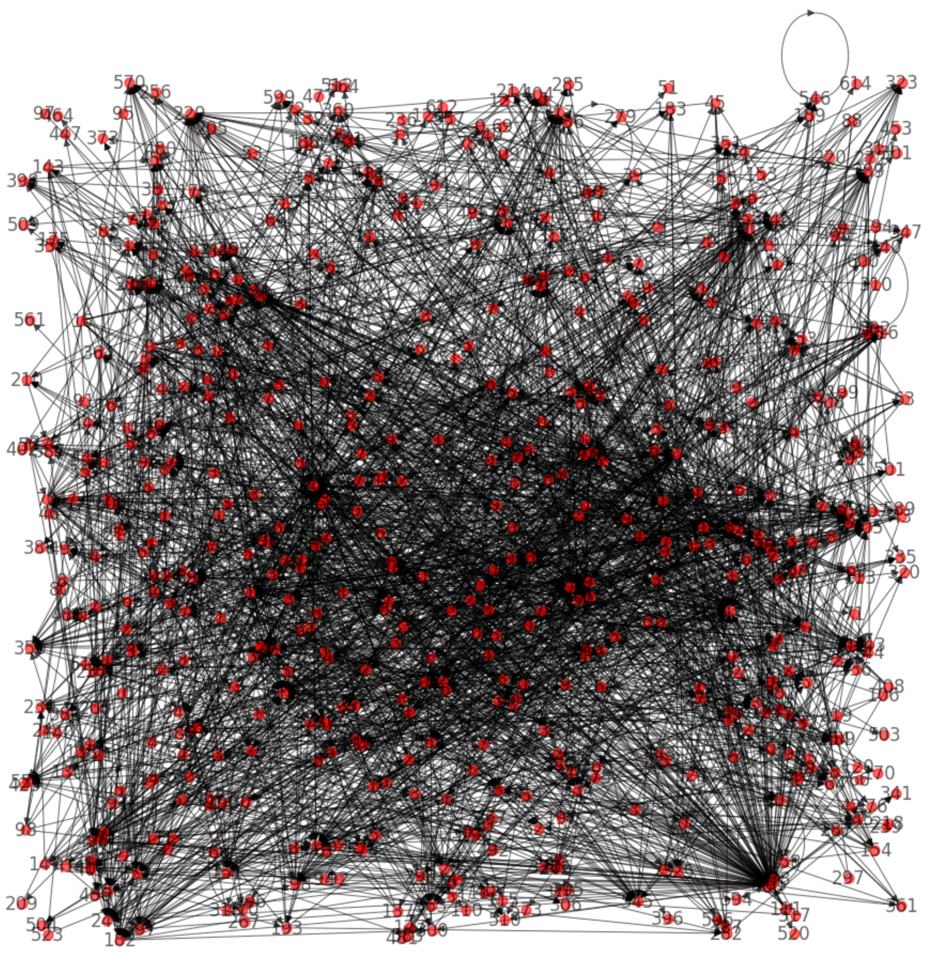


Figure 4.4: Facebook network

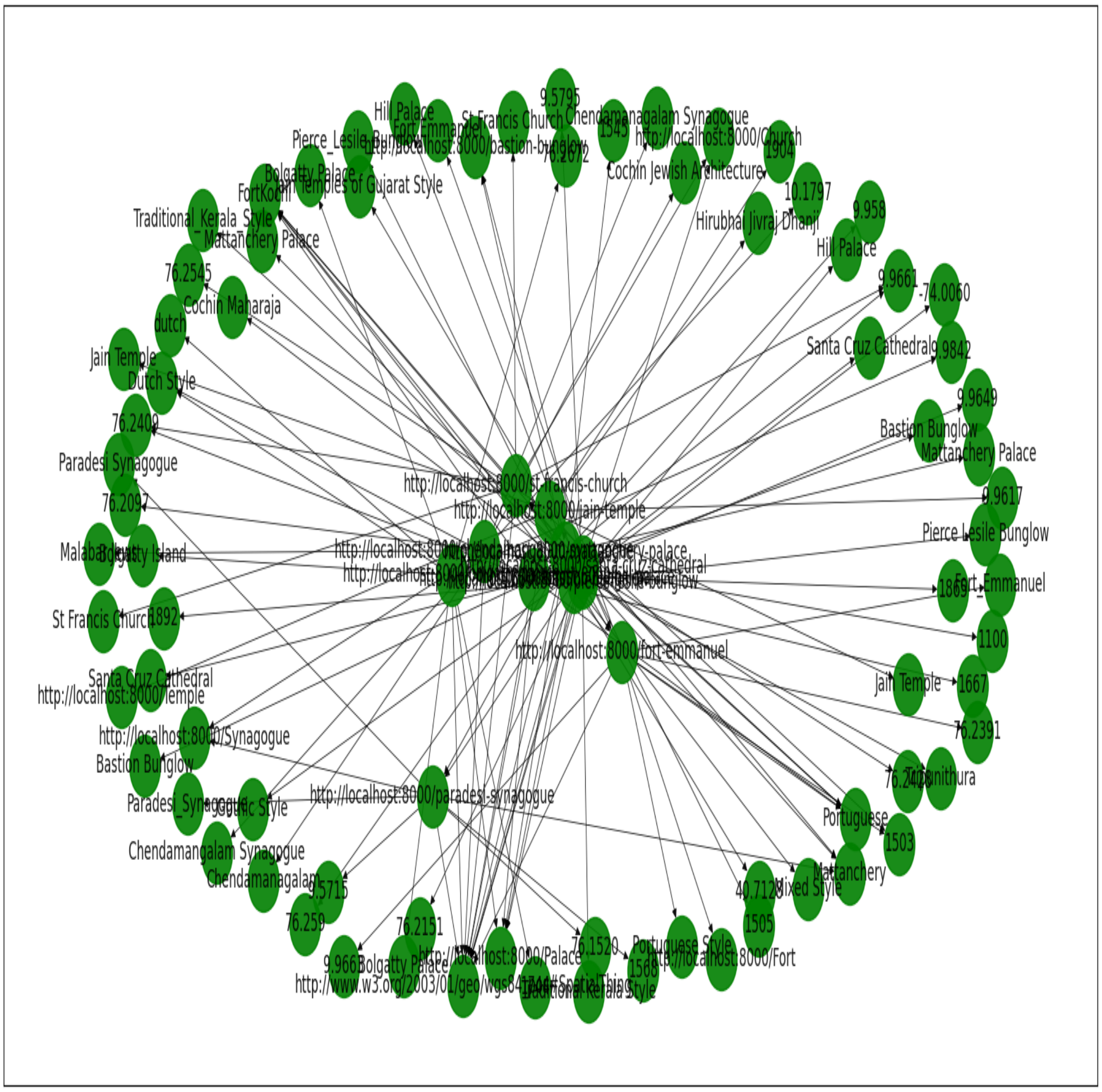


Figure 4.5: Tourism Knowledge Graph

dataset, we provide information on the number of nodes and edges, as well as whether the network is directed or undirected. The datasets are described in more detail below.

- Karate Club Network: A social network of a karate club, where nodes represent members and edges represent friendships or interactions.
- Drug-Gene Interaction: A network representing drug-gene interactions, where nodes represent genes and drugs and edges represent interactions between them.
- Social Network: A social network of employees in a company, where nodes represent employees and edges represent interactions or relationships between them.
- Facebook Network: A social network of Facebook users, where nodes represent users and edges represent friendships or interactions between them.

Dataset Name	**Nodes**	**Edges**	**Directed/ Undirected**	**Source**
Karate Club Network	34	78	Undirected	Zachary, W. W. (1977). An Information Flow Model for Conflict and Fission in Small Groups. Journal of Anthropological Research, 33(4), 452–473.
Cora Knowledge Graph	2,708	2,708	Directed	Originally created by Andrew McCallum and colleagues at the University of Massachusetts Amherst as a benchmark dataset for semi-supervised learning with graph-structured data
Drug-Gene Interaction	1560	9833	Directed	Davis, A. P., Grondin, C. J., Johnson, R. J., Sciaky, D., Wiegers, T. C., Mattingly, C. J. (2017). The Comparative Toxicogenomics Database: update 2017. Nucleic Acids Research, 45(D1), D972–D978.
Social Network	67	174	Undirected	Borgatti, S. P., Everett, M. G., & Johnson, J. C. (2013). Analyzing social networks. SAGE Publications.
Facebook Network	4039	88234	Undirected	McAuley, J. J., & Leskovec, J. (2012). Learning to discover social circles in ego networks. In Advances in Neural Information Processing Systems (pp. 539-547).
Tourism Knowledge Graph	2250	5200	Directed	Huang, Z., Zheng, H. T., Cheng, X., & Huang, Y. (2018). A knowledge graph-based recommendation model for tourism. Journal of Hospitality and Tourism Research, 42(5), 735-758.

Table 4.1: Dataset details

- Tourism Knowledge Graph: A knowledge graph of tourism information, where nodes represent entities (e.g., tourist attractions, events, etc.) and edges represent relationships between them (e.g., located in, has event).

- Cora Knowledge Graph:The Cora knowledge graph is a graph-based representation of the Cora dataset, where each document is represented as a node in the graph, and the citation links between documents are represented as edges. In addition to the citation links, the knowledge graph also includes other types of relationships between documents, such as authorship and publication venue.

  The Cora knowledge graph is a useful tool for exploring the relationships between research papers in the dataset and identifying patterns and trends in the research. By analyzing the graph structure and the properties of individual nodes and edges, researchers can gain insights into the citation and collaboration networks within the research community.

  The Cora knowledge graph has been used in several research studies to investigate various aspects of the citation and collaboration networks within the Cora dataset, such as the impact of different types of citation links on the network structure, the role of authors and publications in the network, and the emergence of communities and clusters of related papers.

4.4 Architecture

The full architecture of feature extracted link prediction using MLP (Multilayer Perceptron) involves the following steps:

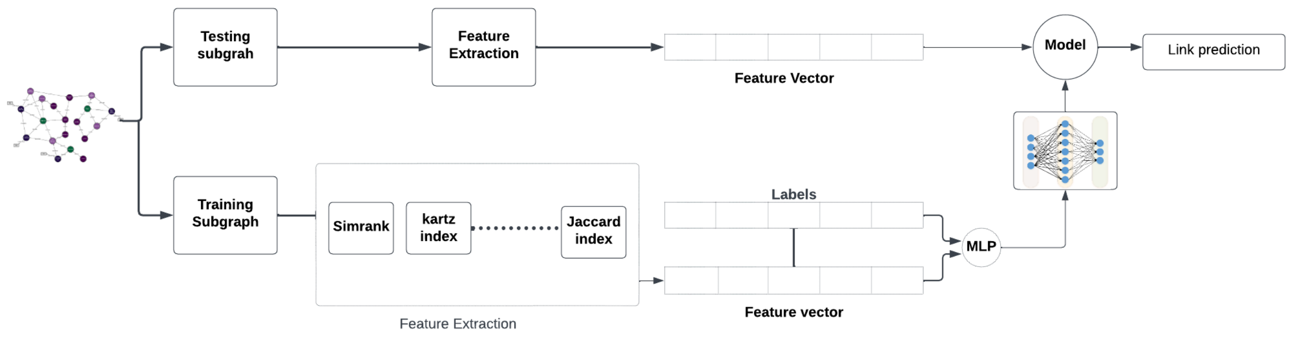


Figure 4.6: Architecture

4.5 Feature Extraction

Feature extraction in nodes of knowledge graphs refers to the process of extracting and representing the important characteristics of each node in the graph as a set of features or attributes. These features can be used to capture the semantic meaning of the nodes and to enable machine learning algorithms to make predictions or perform other tasks on the graph.Feature extraction is an important step in knowledge graph analysis and can have a significant impact on the performance of downstream tasks.

There are different methods for feature extraction in knowledge graphs, depending on the specific use case and the available data. Some common approaches include:

4.5.1 Structure-based Similarity Index

Structured-based similarity indices are a type of algorithm used in knowledge graph analysis to quantify the similarity between two nodes based on their structural properties. In other words, these indices help determine the degree to which two nodes are connected or share other characteristics within the graph. This information can be used in a variety of applications, such as link prediction or recommendation systems. For example, a structured-based similarity index may be used in a recommendation system to suggest items that are similar to those already chosen by a user, based on the structural properties of the items in the graph.

Some common structural properties that may be used in structured-based similarity indices include the connectivity of a node (i.e., the number of other nodes it is connected to), the neighborhood of a node (i.e., the set of nodes that are directly connected to it), and other graph-based characteristics such as betweenness centrality or clustering coefficient. By considering these properties, structured-based similarity indices can provide a more nuanced understanding of the relationships between nodes in a knowledge graph, beyond simple keyword matching or other surface-level features.

Jaccard Index

Jaccard coefficient gives the normalized score to the common neighbor's predicted score. It gives the probability score for hidden links between two nodes. Therefore, score is normalized between 0 and 1 and the expression can be represented as

$$LPJC(x, y) = \frac{|N(x) \cap N(y)|}{|N(x) \cup N(y)|} \tag{4.1}$$

where LPJC(x, y) is the predicted score of hidden link between node x and node y, using Jaccard coefficient. Here denominator indicate the maximum possible common neighbors between node x and y.

Adamic-Adar Index

In Adamic Adar, the node pairs are assigned with a high score if their common neighbors are not being shared with other nodes.44 This measure is based on the fact that a pair of nodes have a higher tendency to establish future links if they are connected to a common neighbor, which itself has no further relationships with other nodes. More is the value of such common neighbors, better is the chances for establishing links between said nodes.10 This can be mathematically expressed as given in this Equation

$$LPAA(x,y) = \sum_{z \in N(x) \cap N(y)} \frac{1}{\log(N(z))} \tag{4.2}$$

where LPAA(x, y) is the predicted score of hidden link between node x and node y using Adamic Adar and z is the adjacent node of both x and y

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. The intuition behind this measure is that nodes which are highly connected in the network are more likely to establish new relationships with other nodes. The predicted score for hidden links can be measured by multiply the degree of both the nodes. It can be mathematically represented as follows

$$LPPA(x,y) = N(x) \times N(y) \tag{4.3}$$

where LPPA(x, y) is the predicted score of hidden link between node x and node y using preferential attachments.

4.5.2 Node-based Similarity Measure

Node-based similarity measures are a type of algorithm used in knowledge graph analysis to measure the similarity between two nodes based on their attributes, properties or features. These measures are often used in various applications, such as node classification, recommendation systems, or link prediction. For instance, node-based similarity measures can be used in a recommendation system to suggest items to a user based on the similarity of their attributes to those already chosen by the user. Node-based similarity measures can also be used in node classification, where the goal is to predict the class or label of a node based on its attributes and the attributes of its neighboring nodes.

Some commonly used node-based similarity measures include cosine similarity, Jaccard similarity, and Pearson correlation. Cosine similarity measures the cosine of the angle between two vectors, where each vector represents the set of attributes of a node. Jaccard similarity measures the ratio of the intersection of the attributes of two nodes to the union of the attributes. Pearson correlation measures the linear relationship between two sets of attributes. Other node-based similarity measures include Euclidean distance and Manhattan distance, which measure the distance between two vectors of attributes. Each of these measures has its own strengths and limitations, and the choice of measure depends on the specific application and the nature of the data being analyzed.It includes:

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.

$$cosinesimilarity(x, y) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \, \|\mathbf{y}\|} \tag{4.4}$$

Euclidean Distance

The Euclidean distance measures the dissimilarity between two nodes based on the distance between their feature vectors in Euclidean space. It is calculated as the square root of the sum of the squared differences between corresponding elements of the feature vectors.

$$Euclideandistance(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{4.5}$$

4.5.3 Path based Simliliarity Measure

Path-based similarity measures are another type of algorithm used in knowledge graph analysis to measure the similarity between nodes based on the paths connecting them in the graph. In other words, these measures take into account the structure of the graph and the paths connecting the nodes, rather than just the attributes or properties of the nodes themselves. Path-based similarity measures can be used in a variety of applications, such as link prediction, recommendation systems, or entity resolution. For example, a path-based similarity measure can be used in a recommendation system to suggest items that are similar to those already chosen by a user, based on the paths connecting the items in the graph.

Some common path-based similarity measures include Shortest Path, Random Walk with Restart, and SimRank. Shortest Path measures the distance between two nodes along the shortest path connecting them. Random Walk with Restart measures the probability of reaching one node from another in a specified number of steps, taking into account the probability of restarting the random walk from either node. SimRank measures the similarity between two nodes based on the similarity of their neighbors. Other path-based similarity measures include Personalized PageRank and Hitting Time, which measure the probability of reaching one node from another in a specified number of steps, taking into account the importance of each node in the graph. Each of these measures has its own strengths and limitations, and the choice of measure depends on the specific application and the nature of the data being analyzed.

Kartz Measure

This is the most useful measure to quantify hidden links between two nodes, which is based on the number of paths between the nodes. Here all paths having length less than diameter have been considered. This measure is calculated as the summation of a total number of paths of all length existing between a pair of node.

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

(4.6)

where LPKZ(x, y) is the link predicted score using Kartz measure, beta is the small constant value, and pathi x,y is the set of all path lengths from x to y

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. It can be defined in this Equation

$$LPSR(x,y) = \gamma \frac{\sum\limits_{a \in N(x)} \sum\limits_{b \in N(y)} LPSR(a,b)}{|N(x)||N(y)|} \tag{4.7}$$

4.6 Deep Neural Network

In this work, a deep neural network is used to predict the presence of links in a graph. A deep neural network is a type of machine learning algorithm that is modeled after the structure and function of the human brain, composed of multiple layers of interconnected neurons. The input layer receives data, which is then processed through multiple hidden layers, with each layer performing a specific transformation, before producing an output.

In this work, the deep neural network is trained on a feature matrix representing a graph, with the target variable being whether or not a link is present between nodes. The model is then used to predict links in a test graph.

4.6.1 Multi-Layer Perceptron

is a Multi-Layer Perceptron (MLP) model with two hidden layers. The term "Multi-Layer" in MLP refers to the fact that the model has multiple layers of neurons (i.e., hidden layers) between the input and output layers. In this case, the model has two hidden layers with 64 and 32 neurons, respectively, before the output layer. This model using the 'adam' optimizer and binary cross-entropy loss.

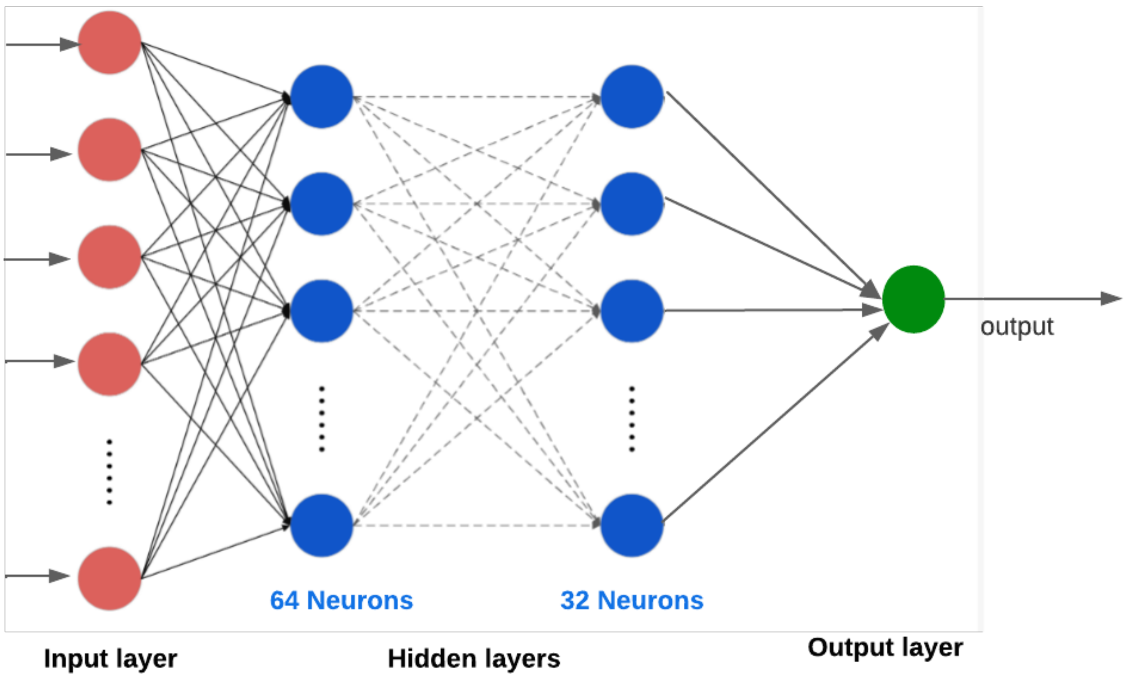


Figure 4.7: Multi-Layer Perceptron Architecture

ReLu

ReLU (Rectified Linear Unit) is an activation function here used in the hidden layers. It is a simple non-linear function that maps any negative input to zero and passes any positive input through unchanged. Mathematically, the ReLU activation function is defined as

follows:
f(x) = max(0, x)

Sigmoid

The sigmoid activation function is a non-linear function used in output layer for binary classification. It maps any input to a value between 0 and 1, which can be interpreted as a probability. Mathematically, the sigmoid activation function is defined as follows:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{4.8}$$

Adam Optimizer

The Adam optimizer is an adaptive learning rate optimization algorithm that is commonly used in deep learning. It stands for Adaptive Moment Estimation, and it combines the benefits of two other popular optimization algorithms: Adagrad and RMSprop.

Like Adagrad and RMSprop, Adam computes individual adaptive learning rates for each parameter in the model, which allows it to effectively update the parameters with different learning rates depending on their importance.

In addition, Adam uses momentum, which helps accelerate the optimization process by keeping track of the moving average of the gradients. It also incorporates bias-correction, which is used to correct the bias introduced by the momentum estimation and helps Adam converge faster.

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \tag{4.9}$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \tag{4.10}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{4.11}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{4.12}$$

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{4.13}$$

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.

Binary cross-entropy loss is derived from information theory and measures the amount of surprise or "information gain" when making predictions. The loss function is given by the following formula:

$$L(y, y') = -(y \cdot \log(y') + (1 - y) \cdot \log(1 - y')) \tag{4.14}$$

4.7 Tools

4.7.1 Google Colab

Google Colab is a cloud-based platform for executing and collaborating on machine learning and data science projects. It is built on top of Jupyter Notebooks and provides a free and convenient way for users to write, run, and share code online. With Colab, users can access powerful hardware resources such as GPUs and TPUs without the need for expensive hardware investments, making it an ideal platform with cutting-edge machine learning techniques.

Colab integrates seamlessly with Google Drive, allowing users to store and share their notebooks and data files with ease. It also provides access to a vast array of open-source libraries and frameworks, including TensorFlow, PyTorch, and scikit-learn, making it easy for users to build and train models quickly. Additionally, Colab allows users to install and use custom packages, ensuring that users can tailor their environments to their specific needs. Overall, Google Colab is an excellent tool for anyone looking to learn and experiment with machine learning and data science.

4.7.2 Neo4J

Neo4j is a graph database management system that is designed to store, manage, and query highly connected data. Unlike traditional relational databases, which use tables and columns to organize data, graph databases use nodes and relationships to represent and store data.

In a graph database like Neo4j, data is represented as nodes, which are connected to each other through relationships. Each node can have properties that describe its characteristics, while each relationship can have properties that describe the connection between nodes. This allows for more natural and intuitive representation of complex data structures, such as social networks, recommendation engines, and fraud detection systems.

Neo4j supports the Cypher query language, which is specifically designed for graph databases. Cypher is a declarative language that allows users to write intuitive and expressive queries to retrieve and manipulate data from the graph database.

4.7.3 Python

Python is a popular programming language for machine learning and data science due to its simplicity, versatility, and availability of powerful open-source libraries. One of the most widely used libraries for machine learning in Python is scikit-learn, which provides a wide range of tools for classification, regression, clustering, and other machine learning tasks.

Scikit-learn provides a user-friendly and consistent interface for machine learning algorithms, making it easy to apply them to real-world datasets. One of the machine learning tools available in scikit-learn is the Multi-Layer Perceptron (MLP) model. MLP is a type of artificial neural network that is used for supervised learning tasks such as classification and regression. MLP models consist of multiple layers of neurons, each of which is connected to the neurons in the adjacent layers.Scikit-learn's MLP model implementation provides a flexible and efficient tool for training neural networks on large datasets.

4.7.4 PyTorch

PyTorch is an open-source deep learning framework that provides a high-level API for building and training deep learning models. It was developed primarily by Facebook's AI research team and is written in Python, making it a popular choice for researchers and practitioners in the field of machine learning.

PyTorch provides a range of neural network modules, including Multi-Layer Perceptron (MLP) models, that allow users to build complex models with ease. The PyTorch API is designed to be intuitive and flexible, allowing users to quickly prototype and experiment with different model architectures and hyperparameters.

One of the key features of PyTorch is its dynamic computational graph, which allows for dynamic and efficient execution of computations. This feature enables users to easily modify and experiment with their model architectures during the training process. Additionally, PyTorch provides an autograd engine that automatically computes gradients for any differentiable function, making it easy to perform backpropagation for training deep learning models.

PyTorch also provides a range of tools for distributed training and deployment, making it well-suited for large-scale machine learning applications. Its integration with popular Python libraries such as NumPy and Pandas also makes it easy to preprocess and manipulate data before training deep learning models.

Overall, PyTorch is a powerful and flexible deep learning library that provides a high-level API for building and training complex models, including MLPs. Its intuitive API, dynamic computational graph, and automatic differentiation make it a popular choice for researchers and practitioners in the field of machine learning.

4.7.5 TensorFlow

Tensorflow is a popular open-source machine learning library developed by Google. It provides a comprehensive set of tools for building and training deep learning models, including Multi-Layer Perceptron (MLP) models. TensorFlow is written in C++ and provides APIs for multiple programming languages, including Python, C++, and Java.

One of the key features of TensorFlow is its computational graph, which allows users to define complex neural network architectures in a high-level language and efficiently execute them on various hardware, including CPUs, GPUs, and TPUs. The computational graph is also used to perform automatic differentiation, which is essential for backpropagation and gradient-based optimization algorithms used in deep learning..

4.7.6 NetworkX

NetworkX is a Python package for the creation, manipulation, and study of complex networks, including social networks, biological networks, and infrastructure networks. It provides a simple yet powerful interface for constructing and analyzing graphs and networks, including tools for visualization, statistical analysis, and modeling.

The core of the NetworkX package is a set of algorithms and data structures for representing and manipulating graphs, including functions for adding and removing nodes and edges, computing shortest paths, finding cliques and communities, and generating random graphs. NetworkX also includes a range of graph layout algorithms for visualizing networks, as well as tools for importing and exporting graphs in various formats, including GraphML, GEXF, and Pajek. Its ease of use and flexibility make it a popular choice for network analysis in both academia and industry.

4.7.7 Stellargraph

Stellargraph is a Python library for machine learning on graphs and networks. It provides a range of tools for working with graph-structured data, including algorithms for graph embedding, node classification, link prediction, and graph clustering. Stellargraph is built on top of TensorFlow and Keras, providing a flexible and scalable framework for graph machine learning.

One of the key features of Stellargraph is its support for heterogeneous graphs, which allow for modeling multiple types of nodes and edges with different attributes and relationships. This makes it particularly well-suited for applications such as recommender systems, where users, items, and other entities may have different attributes and relationships. Stellargraph also provides tools for preprocessing graph data, including node feature engineering and graph normalization.

Chapter 5

Experimental Results and Discussion

5.1 Experimental Setup

All the experiment have been performed on a Ryzen5 processor.The proposed Deep learning model have been compared with standard link prediction algorithms.In this study, we aim to investigate the effectiveness of using feature extraction and MLP models for link prediction in different types of knowledge graphs. Specifically, we will experiment with four different knowledge graphs: the Cora knowledge graph, the Facebook complex network, tourism knowledge graphs, and drug-gene network.

To conduct experiments, first obtain the necessary dataset for each knowledge graph and preprocess them to extract relevant features. We will use various feature extraction techniques such as node degree, clustering coefficient, and node centrality measures to create a feature vector for each node. We will also experiment with different normalization and scaling techniques to ensure that our data is suitable for training the MLP models. Once we have created the feature vectors, we will split the data into training and testing sets and use the MLP models to predict links in the test set. We will evaluate the performance of the models using metrics such as precision, recall, and F1 score. We will also conduct experiments with different hyperparameters and optimization algorithms to determine the best configuration for each model. Overall, our goal is to determine whether feature extraction and MLP models can improve the accuracy of link prediction in knowledge graphs, and if so, which techniques are most effective for different types of knowledge graphs.

5.2 Performance Evaluation Metric

As the proposed approach for link prediction has modeled into a binary classification problem, the performance of the link prediction model has been evaluated with the help

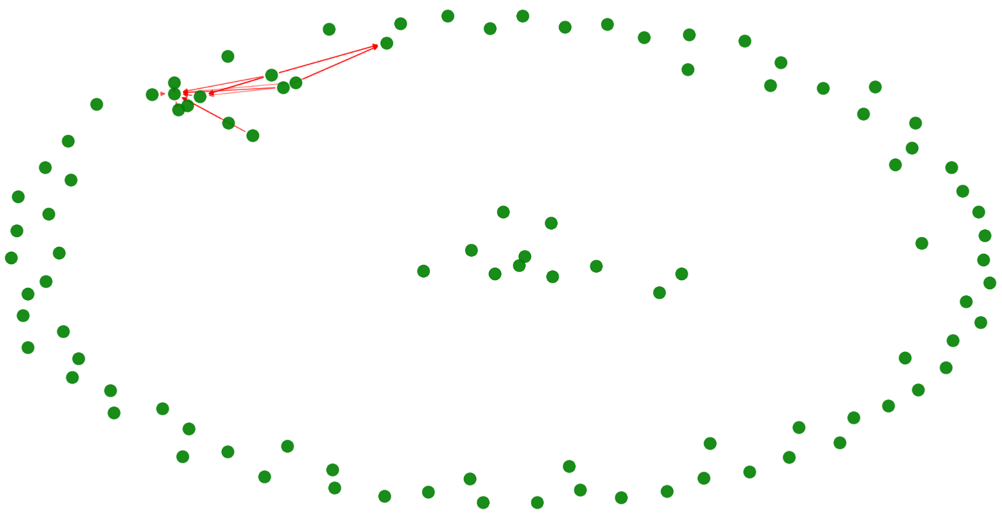


Figure 5.1: Predicted Graph-Tourism Knowledge Graph

of confusion matrix, which is generated after the classification process. Some of the terms related to confusion matrix are discussed as below:

- TP: number of links that are labeled as predicted and also actually predicted by the link predictor.
- FP : number of links that labeled as not to be predicted but are being predicted by the link predictor.
- TN : number of links that are labeled as not to be predicted and are actually not predicted by the link predictor
- FN : number of links that are labeled as to be predicted and are not predicted by the link predictor.
- 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.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

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

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

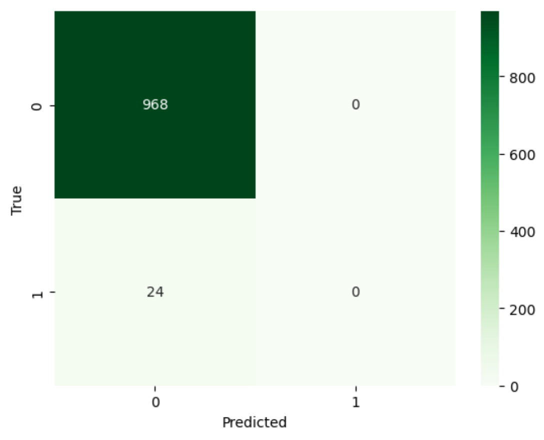
- Recall: Recall is the ratio of TP to the summation of TP and FN.

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

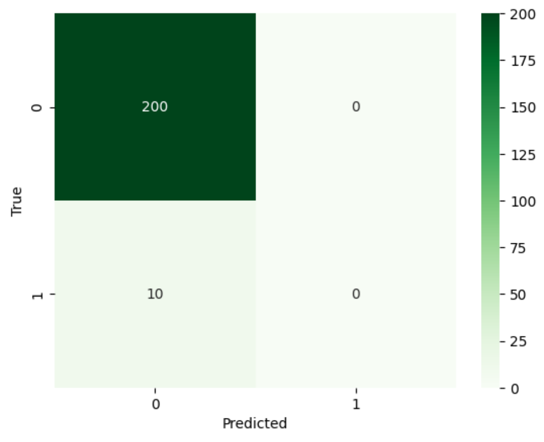
5.3 Results and Discussion

In this work, we have used several performance evaluation metrics to assess the effectiveness of our model in making recommendations.he performance evaluation metrics used in this work (precision, recall, accuracy, loss functions, and confusion matrix) are widely used in the evaluation of machine learning models, including those for knowledge graph applications.

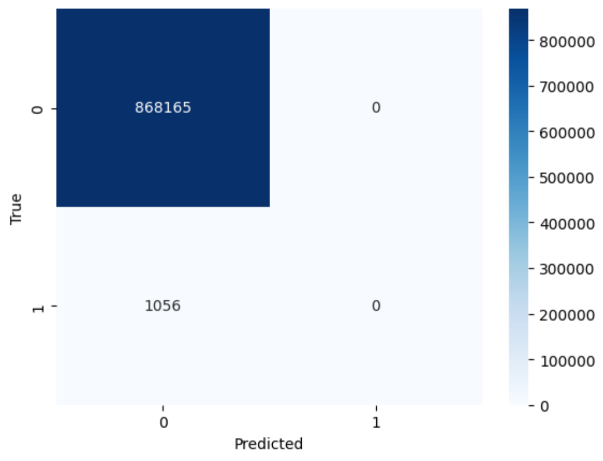
In the Cora dataset, the accuracy achieved by the neural network model was 99.875 percentage, indicating a high degree of accuracy in predicting the labels of the nodes. In the tourism knowledge graph, the precision and recall scores were used to evaluate the performance of the model in identifying relevant entities for a given query, and the results showed an improvement over previous methods.



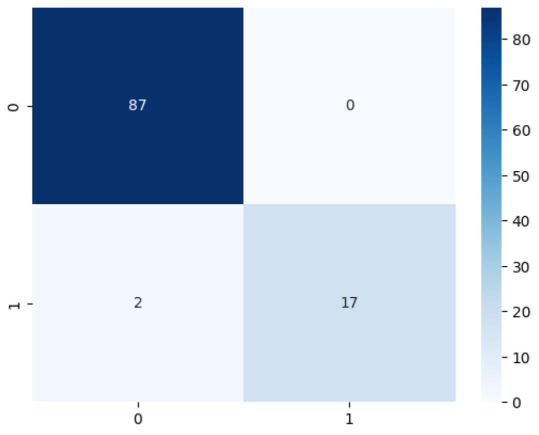
(a) Confusion matrix-Facebok



(b) Confusion matrix-Wiki-data

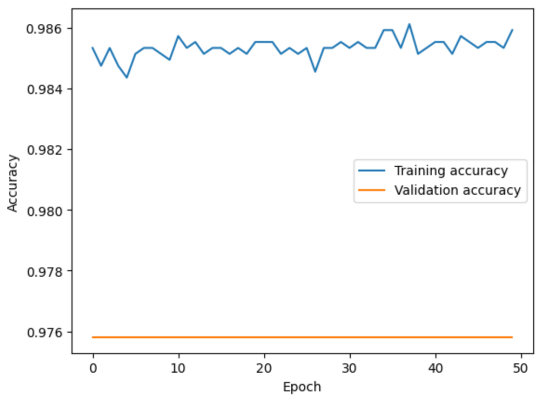


(c) Confusion matrix-Cora

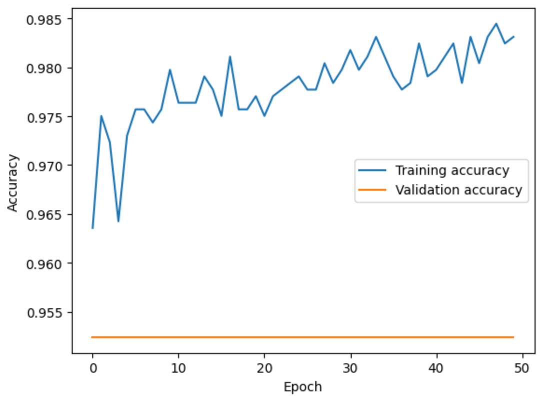


(d) Confusion matrix-Tourism

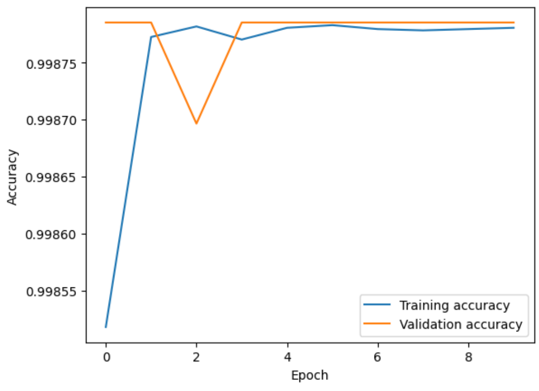
Figure 5.2: Confusion Matrix of all Dataset



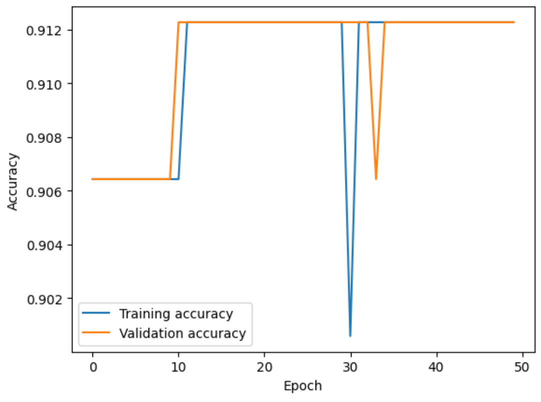
(a) Accuracy-Facebook



(b) Accuracy-Wiki-data

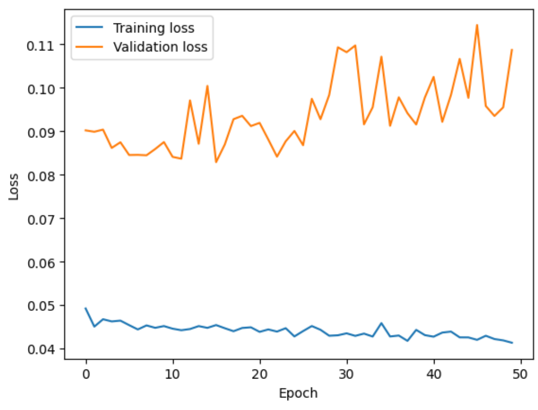


(c) Accuracy-Cora

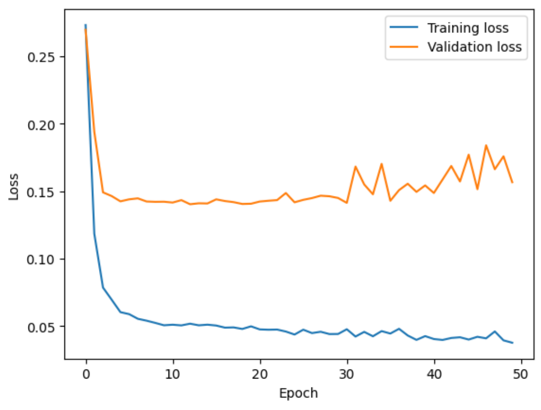


(d) Accuracy-Tourism

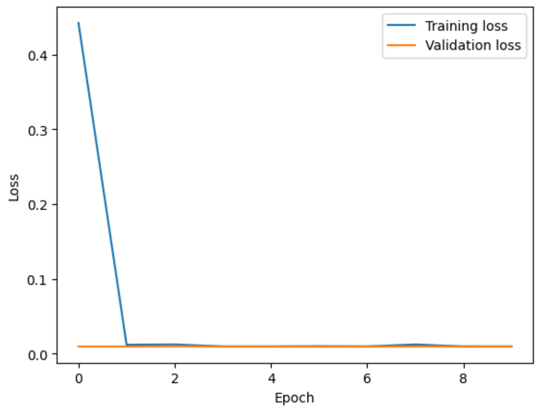
Figure 5.3: Accuracy of all dataset



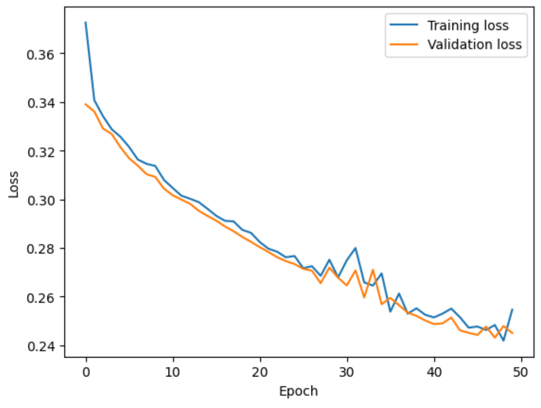
(a) Loss-Facebook



(b) Loss-Wiki-data

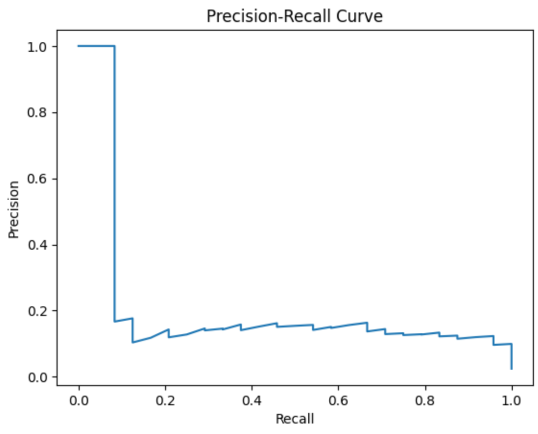


(c) Loss-Cora

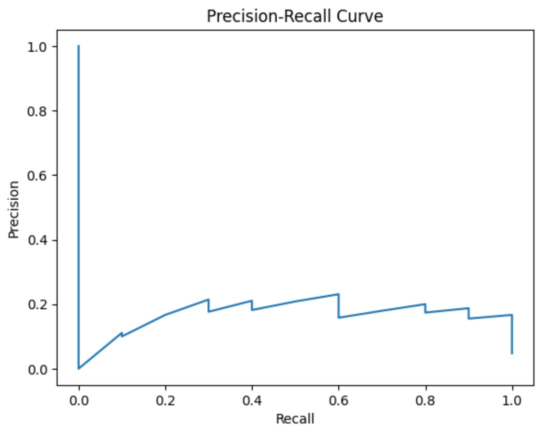


(d) Loss-Tourism

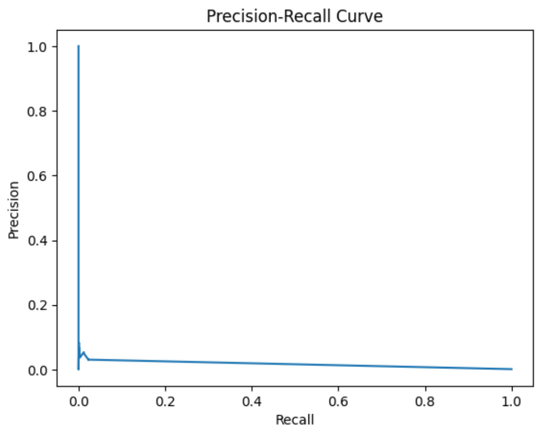
Figure 5.4: Train-Validation loss of all Dataset



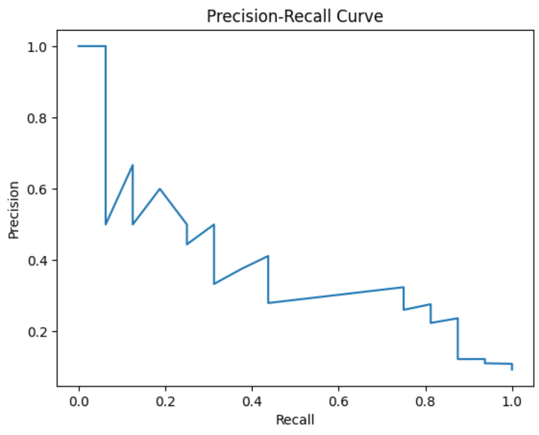
(a) Precision Recall-Facebook



(b) Precision Recall-Wiki-data



(c) Precision Recall-Cora



(d) Precision Recall-Tourism

Figure 5.5: Precision-Recall of all Dataset

In the Facebook social network dataset, the F1 score was used to evaluate the performance of a graph embedding model, which achieved a score of 0.935. This indicates that the model has high precision and recall in predicting the links between nodes in the graph.

In the Wiki data knowledge graph, a combination of evaluation metrics was used to evaluate the performance of a knowledge graph completion model, including hits@10, hits@100, hits@500, MRR, and AMRI. The results showed that the model achieved high accuracy in predicting missing links in the knowledge graph. And also in the Tourism Knowledge graph the accuracy is 98.75 percentage.

Overall, these performance evaluation metrics are important for understanding the strengths and weaknesses of machine learning models in different knowledge graph applications. While the accuracy achieved in each application varies, the use of these metrics allows for a consistent and objective evaluation of model performance.

Chapter 6

Conclusion and Future works

Link prediction in complex networks is an emerging research domain in social network analysis. It could be applied in modeling and understanding the evolution of social groups in a network. Such interpretation can help us for the effective implementation of models to discover hidden groups or the absent relationships in the groups. Using this inductive learning of feature extraction techniques in conjunction with Multi-Layer Perceptron for link prediction has the potential to improve performance.This work have achieved high accuracy. shown in table 6.1

Feature extraction allows for the selection and transformation of relevant features, which

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

Table 6.1: Accuracy of Knowledge Graphs

can improve the Multi-Layer Perceptron's ability to capture complex nonlinear relationships between the features and accurately predict links.

In the future, the dynamic network may be considered, where links and nodes are added dynamically over time. In a weighted network, the weights associated with the links are one kind of measure to quantify the strength of bonding. This could also be a potential source for link prediction algorithms. And link prediction could be used in the data analysis in the area of security and criminal investigation.This can be embedded in the proposed model for predicting future links.

Bibliography

[1] M. Wang, L. Qiu, and X. Wang, “A survey on knowledge graph embeddings for link prediction,” *Symmetry*, vol. 13, p. 485, 03 2021.

[2] https://pykeen.readthedocs.io/en/stable/tutorial/inductive$_lp.htmlinductive-link-prediction$

[3] https://enterprise-knowledge.com/whats-the-difference-between-an-ontology-and-a-knowledge graph/

[4] https://www.javatpoint.com/multi-layer-perceptron-in tensorflow

[5] https://snap-stanford.github.io/cs224w-notes/machine-learning-with-networks/message-passing-and-node-classification

[6] https://starship-knowledge.com/pytorch-biggraph

[7] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, “Translating embeddings for modeling multi-relational data,” *Advances in neural information processing systems*, vol. 26, 2013.

[8] A. García-Durán, S. Dumančić, and M. Niepert, “Learning sequence encoders for temporal knowledge graph completion,” *arXiv preprint arXiv:1809.03202*, 2018.

[9] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, “Embedding entities and relations for learning and inference in knowledge bases,” *arXiv preprint arXiv:1412.6575*, 2014.

[10] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon, “Representing text for joint embedding of text and knowledge bases,” in *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 1499–1509, 2015.

[11] T. P. Trouillon and G. M. Bouchard, “Complex embeddings for simple link prediction,” Nov. 23 2017. US Patent App. 15/156,849.

[12] K. Wang, Y. Liu, X. Xu, and D. Lin, “Knowledge graph embedding with entity neighbors and deep memory network,” *arXiv preprint arXiv:1808.03752*, 2018.

[13] D. Q. Nguyen, T. D. Nguyen, D. Q. Nguyen, and D. Phung, “A novel embedding model for knowledge base completion based on convolutional neural network,” *arXiv preprint arXiv:1712.02121*, 2017.

[14] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, “Rotate: Knowledge graph embedding by relational rotation in complex space,” *arXiv preprint arXiv:1902.10197*, 2019.

[15] Y. Luo, C. Yang, B. Li, X. Zhao, and H. Zhang, “Cp tensor factorization for knowledge graph completion,” in *Knowledge Science, Engineering and Management: 15th International Conference, KSEM 2022, Singapore, August 6–8, 2022, Proceedings, Part I*, pp. 240–254, Springer, 2022.

[16] T. Shen, Y. Mao, P. He, G. Long, A. Trischler, and W. Chen, “Exploiting structured knowledge in text via graph-guided representation learning,” *arXiv preprint arXiv:2004.14224*, 2020.

[17] Z. Zhang, X. Liu, Y. Zhang, Q. Su, X. Sun, and B. He, “Pretrain-kge: learning knowledge representation from pretrained language models,” in *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 259–266, 2020.

[18] D. Li, M. Yi, and Y. He, “Lp-bert: Multi-task pre-training knowledge graph bert for link prediction,” *arXiv preprint arXiv:2201.04843*, 2022.

[19] W. Zhao, A. Zhang, L. Shang, Y. Yu, L. Zhang, C. Wang, J. Chen, and H. Yin, “Hyperbolic personalized tag recommendation,” in *Database Systems for Advanced Applications: 27th International Conference, DASFAA 2022, Virtual Event, April 11–14, 2022, Proceedings, Part II*, pp. 216–231, Springer, 2022.

[20] J. Yang, L. T. Yang, H. Wang, Y. Gao, Y. Zhao, X. Xie, and Y. Lu, “Representation learning for knowledge fusion and reasoning in cyber-physical-social systems: Survey and perspectives,” *Information Fusion*, 2022.

[21] W. Yuan, K. He, D. Guan, L. Zhou, and C. Li, “Graph kernel based link prediction for signed social networks,” *Information Fusion*, vol. 46, pp. 1–10, 2019.

[22] R. I. Yaghi, H. Faris, I. Aljarah, A. M. Al-Zoubi, A. A. Heidari, and S. Mirjalili, “Link prediction using evolutionary neural network models,” *Evolutionary Machine Learning Techniques: Algorithms and Applications*, pp. 85–111, 2020.

[23] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[24] M. E. Newman, C. Moore, and D. J. Watts, "Mean-field solution of the small-world network model," *Physical Review Letters*, vol. 84, no. 14, p. 3201, 2000.

[25] T. Opsahl, A. Vernet, T. Alnuaimi, and G. George, "Revisiting the small-world phenomenon: Efficiency variation and classification of small-world networks," *Organizational Research Methods*, vol. 20, no. 1, pp. 149–173, 2017.

[26] R. Abdul, A. Paul, J. Gul M, W.-H. Hong, and H. Seo, "Exploiting small world problems in a siot environment," *Energies*, vol. 11, no. 8, p. 2089, 2018.

[27] D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks," in *Proceedings of the twelfth international conference on Information and knowledge management*, pp. 556–559, 2003.

[28] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," *Advances in neural information processing systems*, vol. 31, 2018.

[29] Y. Shang, Z. Hao, C. Yao, and G. Li, "Improving graph neural network models in link prediction task via a policy-based training method," *Applied Sciences*, vol. 13, no. 1, p. 297, 2022.

[30] X. Liu, X. Li, G. Fiumara, and P. De Meo, "Link prediction approach combined graph neural network with capsule network," *Expert Systems with Applications*, vol. 212, p. 118737, 2023.

[31] L. Wang, Y. Wang, J. Li, B. Wang, and Z. Yu, "Up-gnn: Ensemble graph neural network for link prediction via uncertainty learning and positional capturing," in *2021 7th International Conference on Big Data and Information Analytics (BigDIA)*, pp. 399–405, 2021.