Explainable link prediction in large complex graphs - application to drug repurposing

Kamrul Islam

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Explainable link prediction in large complex graphs - application to drug repurposing

THÈSE

présentée et soutenue publiquement le 16 Décembre 2022

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Md Kamrul ISLAM

Composition du jury

Président :	Fatiha Saïs, Professeure, Université Paris Saclay, France
Rapporteurs :	Luc Brun, Professeur, ENSICAEN, France Paolo Merialdo, Professeur, Université Rome III (Roma Tre University), Italie
Examinateurs :	Miguel Couceiro, Professeur, Université de Lorraine, France Fatiha Saïs, Professeure, Université Paris Saclay, France
Invité :	Marie-Dominique Devignes, Chargée de Recherches, CNRS, France
Encadrants :	Malika Smaïl-Tabbone, Maître de conférences, HDR, Université de Lorraine, France Sabeur Aridhi, Maître de conférences, Université de Lorraine, France

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Acknowledgments

I would like to take this opportunity to show my sincere gratitude to all who have directly or indirectly helped me sail through the tides.

I must start with my supervisors, Malika Smail-Tabbone and Sabeur Aridhi, for their emerging ideas, proper guidance, and continuous support in the form of encouragement to carry on this research on the right track. They have always impressed me with their outstanding professional conduct, their strong conviction for science, and their belief that a doctoral program is only the start of a life-long learning experience. I appreciate their consistent support from the first day I applied to the doctoral program to these concluding moments. I am truly grateful for their progressive vision of my training in science, their tolerance of my naive mistakes, and their commitment to my future career. I also want to thank them from the bottom of my heart for taking the time to proofread and fix my many mistakes. Without hesitation, I would say that having superiors like them makes me very happy.

Marie-Dominique Devignes, the team leader of CAPSID, deserves a very special thank from me for all that she does to make my learning experience comfortable. I can very clearly remember being driven to the doctorate school by her for my admittance. She gave me advises and did the findings analysis for me during the molecular evaluation in my drug repurposing study, which was a big help to me. In addition to studying, she frequently asked about my family. She is truly a great-hearted person.

I would like to thank the reviewers, Luc Brun and Paolo Merialdo, and the examiners, Miguel Couceiro and Fatiha Saïs, for accepting to be part of the jury board. Their insightful comments improve the overall presentation quality of my thesis.

I also would like to express my gratitude to Bernard Maigret, Emiritus Researcher, CNRS-Nancy for his enormous help in my drug repurposing experiment. Honestly speaking, it could be impossible to complete the molecular evaluation part of the study without his proper guidance.

I thank Mohammed Khatbane, who worked with me as an M2 intern on the development of distributed knowledge graph embedding. I enjoyed working with him. I wish him a very successful life.

I owe a great debt of gratitude to my former colleague Bishnu Sarker, who completed his PhD last year. In the first two years of my PhD, he made my life here easy and comfortable. We had discussions on many disciplinary topics, and those discussions enriched my knowledge a lot. I thank Diego Amaya Ramirez, a PhD researcher at CAPSID, for his help during the drug repurposing study. I also thank Hrishikesh Dhondge, PhD researcher, CAPSID team for helping me in learning biological concepts. We had great discussions nearly every working day, and honestly speaking, I have learned a lot from him. I would like to thank my colleagues Isaure Chauvot de Beauchêne, Hamed Khakzad, Yasaman Karami, Anna Kravchenko, Athénaïs Vaginay, and Antoine Moniot from the CAPSID team for making this journey fun and being there with me. They will always remain in my heart. My list will remain incomplete without thanking Isabelle Herlich and Antoinette Courrier, who were very kind and helpful for all types of administrative work throughout these years.

In the end, I would like to thank my entire family for being with me through my PhD journey. My daughter, Ruhana Jannat Parisha, is my greatest strength. I would like to remember my parents today. I thank my mother, Rejeha Begum, for being the greatest inspiration of my life. I thank her for teaching me how to stay calm and stable during hard times. I thank my father, Abdul Karim Gazi, for his unconditional love and trust in me at every step of life. Lastly, I would like to remember my two uncles, two aunties, and one cousin who died in the last three years. May God rest them in peace.

I am dedicating this thesis to my beloved daughter and mother.

Contents

List of Figures

List of Tables

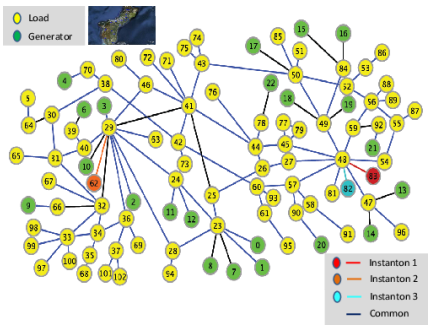
General Introduction

1 Context

Today, uncountable complex systems surround us such as power grids, logistics and transportation, communication networks, and biological systems. Studying these systems is inherently challenging due to the dependencies, relationships, or other types of interactions between their components. We need structured representations of these systems to analyze them. Graphs are diagrammatic and pervasive data structures that can be used to describe any complex system. Although the graph concept was first used by the famous Swiss mathematician Leonhard Euler in 1735 to solve the Königsberg bridge problem, the term "graph" was first used by the English mathematician Sylvester in 1878 [BLW86]. In its most general form, a graph consists of a set of nodes representing objects or entities and a set of links representing relationships between pairs of entities. For an illustration, a power-grid complex system can be represented by a graph that considers power transmission devices (e.g. generators, transformers, relays) as nodes and transmission lines as links (Figure 1a). Graphs offer more than simply beautiful representations. They also provide a mathematical foundation from which we can analyze, comprehend, and extract knowledge about the systems. Considering the potentialities of graphs, researchers studied numerous real-world graphs that range from simple social graphs to complex biological graphs. Figure 1 illustrates a few examples of graphs. Graphs may encode rich information about nodes and their complex relationships. The links in a graph could be directed, where it is only possible to travel from first node to second one, or undirected, where it is also possible to travel from second node to first one. Graphs may contain loops where nodes link themselves and multi-links where two nodes are connected by multiple distinguishable directed links. Graphs without loops, undirected links and multi-links are known as simple graphs. For example, the friendship graph (Figure 1b), where the nodes are people, the links are friendships between couples of people; the PPI graph (Figure 1c), where the nodes are proteins, and the links are interactions between pairs of proteins. Graphs may contain more descriptive information about nodes, such as types, attributes and also about links, such as link names. These types of graphs are generally known as knowledge graphs (Figure 1d). Directions and names are two mandatory requirements of links in knowledge graphs (KGs). These graphs support multiple links between a pair of nodes as well as loops. Therefore, knowledge graphs encode more semantic information than simple graphs.

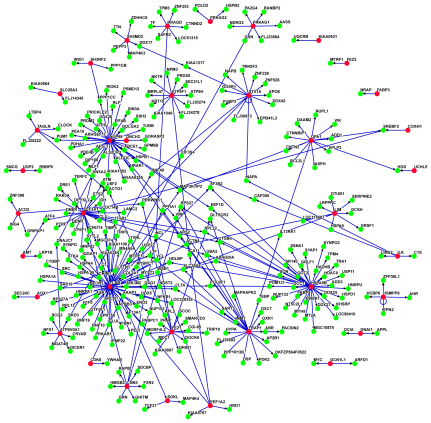
Most real-world graphs are large in size. For example, the HI-III PPI graph [KLS+19] consists of 18,000 human proteins and 35,500 protein-protein interactions; the Freebase knowledge graph [Goo13] contains nearly 44 million nodes representing entities from various domains (e.g. persons, places, religions, companies) and 2.4 billion links representing named relations between entity pairs. However, these graphs are far from their completeness. The birth-place and nationality are missing for about 78.5% and 93.8% of people, respectively, in the Freebase knowledge

[3]Source: https://medium.com/analytics-vidhya/social-network-analytics-f082f4e21b16

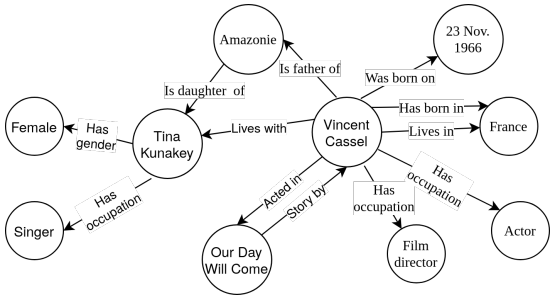


(a) A power graph of the island of Guam [CPS10]

(b) A social graph[3]



(c) A PPI graph [BHC+09]



(d) A toy example of knowledge graph

Figure 1: Examples of graphs. (a) In the power graph of the Island of Guam, the nodes are different types of power devices, and the links are physical connections between pairs of devices; (b) in the friendship graph, a node represents a person, and a link represents the friendship between a couple of persons; (c) in the PPI graph, the nodes are human proteins, and the links represent physical interactions between pairs of proteins; (d) in the toy example of knowledge graph, the nodes are different types entities (e.g. person, place, movie), and the links represent named relationships between pairs of entities.

graph [MGW+13]. The problem of predicting missing links or inferring new links in a graph is commonly known as the link prediction problem [LZ11]. Researchers have developed several approaches for link prediction in graphs, which compute the plausibility of a link between two unconnected nodes in a graph. Formally, link prediction is the task of predicting the likelihood of a link between two nodes based on available topological and/or attribute information of a graph [XPY16]. Link prediction approaches uncover complex relationships in a graph, which helps us make better decisions. For example, the drug repurposing task can be formulated as a link prediction task in a biological graph where unobserved links between drug (approved/ clinically tested) and disease nodes are predicted [SLGA21, FCFP21, KS21]. Experts can subsequently do experimental validation of these predicted drugs-disease pairs toward finding effective drugs for the disease of interest.

Developing a link prediction approach for graphs raises two major challenges. The first challenge is to provide high prediction performance across graphs from various domains. The second one is to provide sufficient explanation of results to bring reliability of the approach.

There exist many approaches in the literature for link prediction in simple graphs, ranging

from node similarity-based heuristics to more recent representation learning-based approaches, aka embedding-based approaches. Embedding-based approaches learn low-dimensional vector representations, also known as embeddings, of graph nodes and then use the vector representations to compute the plausibility of unobserved links in the graph [CWPZ18, ZCH+20, KSSB20]. Studies show that embedding-based approaches offer better link prediction performance across graphs from different domains than similarity-based approaches. However, a major and serious concern with these approaches is their lack of explainability. Classical supervised learning-based approaches could be a potential solution here as they are explainable. However, they require defining a good feature set by hand and often depend on external attributes of nodes that are not available in many graphs, such as social graphs, e-commerce graphs, due to the data privacy concern [MBC16]. In this context, the state-of-the-art raises two questions: (1) How to explain embedding-based link prediction?, and (2) How to define a good feature set without using node attributes for supervised learning-based link prediction?

Link prediction in knowledge graphs is more challenging than in simple graphs as knowledge graphs contain complex relationships between nodes. Link prediction approaches for knowledge graphs are either logical rule-based or embedding-based. Studies show that embedding-based link prediction approaches offer better prediction performance and scalability than traditional rule-based approaches. However, embedding-based approaches are not explainable. Actually embedding-based link prediction approaches in knowledge graphs raise two new challenges in addition to high-prediction performance and explainability. The first one is the generation of high-quality negative examples, as they are not readily available. The second one is the scalability of the link prediction approaches, as real-world knowledge graphs are huge in size. Although there exists numerous approaches for link prediction in graphs, it is remarkable that existing approaches are not good enough to offer satisfying solutions. In this context, the state-of-the-art for link prediction in knowledge graphs calls for three research directions: (1) generating high-quality negative examples for embedding-based approaches; (2) explaining embedding-based link prediction; and (3) implementing embedding-based link prediction approaches in a distributed environment.

2 Contributions

The thesis studies explainable link prediction methods in graphs and their application to biomolecule graphs. Concretely, the major contributions of this research work are:

1. **Explaining embedding-based link prediction in simple graphs**

   Similarity-based and embedding-based approaches are two popular categories for link prediction in simple graphs. Approaches from the first one are explainable but good performers only in graphs with specific characteristics, whereas approaches from the second category offer high prediction performance in most of the graphs but lack explainability. Firstly, we describe the difficulty of computing precision scores of similarity-based approaches and present a strategy to overcome the difficulty. We compare similarity-based approaches to embedding-based approaches on several benchmark graphs with different properties from various domains. Beyond the intra and inter-category comparison of the performances of the approaches, we come up with a new method to uncover interesting connections between embedding-based and similarity-based approaches as a means to alleviate the well-known black-box problem of embedding-based. The results show interesting connections that give ideas about the learned heuristics in embedding-based approaches.

- Islam, M. K., Aridhi, S., Smaïl-Tabbone, M. (2020). A comparative study of similarity-based and GNN-based link prediction approaches. Graph Embedding (GEM) Workshop, ECML-PKDD. available in https://gem-ecmlpkdd.github.io/archive/2020/papers/

-Islam, M. K., Aridhi, S., Smaïl-Tabbone, M. (2021, July). Appraisal study of similarity-based and embedding-based link prediction methods on graphs. In Proceedings of the 10th International Conference on Data Mining & Knowledge Management Process (pp. 81-92).

- Islam, M. K., Aridhi, S., Smaïl-Tabbone, M. (2021). An experimental evaluation of similarity-based and embedding-based link prediction methods on graphs. International Journal of Data Mining & Knowledge Management Process, 11, 1-18.

2. **Supervised link prediction in simple graphs**

   Link prediction based on classical supervised machine learning methods is less explored due to the unavailability of node attribute information and difficulties in defining a good feature set. However, supervised link prediction could be a good solution when a good feature set is available as they are explainable like similarity-based approaches and highly performant like embedding-based approaches. Considering the fact that different similarity-based heuristics focus on different structural aspects of a graph, we define a rich structural feature set where each feature represents a similarity metric. We designed a supervised learning-based link prediction approach based on this feature set. The results show that our supervised link prediction approach provides comparable prediction performance to recent embedding-based approaches. For the purpose of explanation, we compute the importance score of each similarity-metric/feature to identify dominant metrics those contribute in generating links in a graph.

   - Islam, M.K., Aridhi, S., Smail-Tabbone, M. (2022). From Competition to Collaboration: Ensembling Similarity-Based Heuristics for Supervised Link Prediction in Biological Graphs. In: Bangabandhu and Digital Bangladesh. ICBBDB 2021. Communications in Computer and Information Science, vol 1550, pp. 121-135, Springer, Cham.

3. **Simple negative sampling for link prediction in knowledge graphs**

   During the learning of efficient embeddings, the sampling of negative triples was highlighted as an important step, as KGs only contain positive triples. We present a new efficient simple negative sampling (SNS) method to sample high-quality negative triples in KGs. SNS is general and injectable to any geometric KG embedding method to sample negative triples. The results show that SNS improves the prediction performance of KG embedding methods in many knowledge graph datasets and outperforms the existing sampling methods.

   - Islam, M.K., Aridhi, S., Smail-Tabbone, M. (2022). Simple Negative Sampling for Link Prediction in Knowledge Graphs. In: Complex Networks & Their Applications X. COMPLEX NETWORKS 2021. Studies in Computational Intelligence, vol 1016, pp. 549-562. Springer, Cham.

4. **Explaining link prediction in knowledge graphs**

   The applicability of embedding-based link prediction approaches to knowledge graphs is often limited due to their failure in explaining the predictions. We developed a new method to learn high-quality rules based on a knowledge graph and its learned embeddings. Then, we use an abductive strategy to explain predictions by the embedding method based on the instantiation of the learned rules to find explaining paths for the predictions.

   - Islam, M. K., Aridhi, S., Smaïl-Tabbone, M. (2022). Negative sampling and rule mining for explainable link prediction in knowledge graphs. Knowledge-Based Systems (*Elsevier*), 109083.

5. **Explainable and molecular-validated drug repurposing for COVID-19**

   As each knowledge graph embedding method has its own strong and weak points in learning embeddings, we generate high-quality ensemble embeddings of knowledge graphs based on the embeddings learned by complementary embedding methods. We present a new drug repurposing approach based on the ensemble embeddings of a biological knowledge graph. The approach formulates the link prediction problem as predicting repurposable drugs for a disease. We apply our drug repurposing approach to a COVID-19 centric knowledge graph. The results show that our approach offers better prediction performance. We also apply our explainable link prediction approach to provide explanations of our predictions. To increase the reliability of our predictions, we also integrate molecular evaluation to further validate our predictions, which is the first attempt to combine a knowledge graph embedding-based approach with the established virtual screening approach for drug repurposing.

   - Islam, M. K., Amaya-Ramirez, D., Maigret, B., Devignes, M.D., Aridhi, S., & Smaïl-Tabbone, M. Molecular-evaluated and explainable drug repurposing for COVID-19 using ensemble knowledge graph embedding. (Submitted)

6. **A distributed framework for learning knowledge graph embedding with Ray**

   Traditional knowledge graph embedding methods learn embedding in a centralized way (single machine). But most of the real-world KGs are huge in size, and a single machine is not enough to store such knowledge graphs and learn embeddings of them. We come up with a new distributed framework for learning knowledge graph embeddings in a fully distributed and parallel manner. We use the Ray [MNW$^+$18] environment for implementing our framework. The results show that our distributed framework for training embedding methods is scalable to very large-scale knowledge graphs and to the size of the used cluster while preserving link prediction performance.

   - Khatbane, M., Islam, M. K., Aridhi, S., Smaïl-Tabbone, M. A distributed framework for learning knowledge graph embedding with Ray (Article under preparation)

3 Thesis scope

This thesis focuses on studying explainable link prediction in simple graphs and knowledge graphs. The studied link prediction approaches consider only structural features of graphs, and hence the external attribute information of nodes and links (except name) in KGs is not considered. We evaluate our link prediction approaches based on several public benchmark datasets. As for the application of our explainable link prediction approach for knowledge graphs, we study the drug repurposing task using a biological knowledge graph. Although our drug repurposing framework is generic, we apply the framework to drug repurposing for COVID-19 disease only. We use the open source Ray framework [MNW+18] for developing a distributed and parallel link prediction approach for knowledge graphs.

4 Thesis organisation

The rest of this thesis report is organized as follows. **Chapter 1** provides preliminaries about graphs. The chapter starts by providing a detailed description of graphs, their structures, and properties. Then, it provides the formulation and existing approaches for link prediction in simple and knowledge graphs. The chapter ends with a description of a few real-world applications of link prediction. **Chapter 2** presents a comparative study on link prediction approaches and their linkages for providing explanations of embedding-based link prediction in simple graphs. This chapter also describes our supervised learning-based approach for link prediction in simple biological graphs and provides explanation of the predictions. **Chapter 3** is comprised of two parts. The first part describes our SNS method to sample high-quality negative examples for knowledge graph embedding methods with comprehensive results. The second part describes our rule mining method based on the KG and its embeddings. This chapter then presents an abduction strategy to instantiate the learned rules for explaining embedding-based link predictions. **Chapter 4** describes our drug repurposing framework based on knowledge graph embeddings. The chapter also presents explanations of our predictions using our prediction explanation strategy. It ends with providing molecular evaluations of predictions to increase reliability of our approach. **Chapter 5** presents our distributed framework for training knowledge graph embedding methods on large scale knowledge graphs. The chapter presents a few preliminary results. Finally, **Chapter 6** summarizes the whole work with a brief self-assessment and discusses some perspectives. Some additional information is provided in the appendices.

Chapter 1

Background

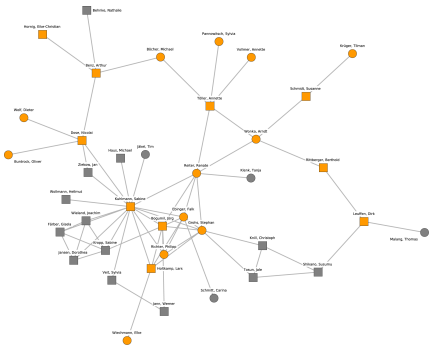
Contents

1.1 Preliminaries about graphs

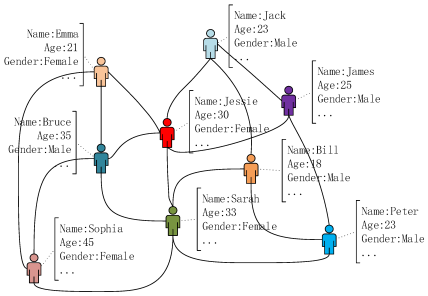
1.1.1 Definitions

A **graph**, $G = (V, E, X, W)$ is a representation of a network, where V is the set of vertices (or nodes), X is a set of node attributes, $E \subseteq V \times V$ is a set edges (or links) that connects two vertices and W is a set of link attributes. Nodes represent entities in networks, such as persons in a social network, proteins in a protein-protein interaction (PPI) network, authors and papers in a co-authorship network, and so on. Nodes may contain descriptive information, such as names, types, sources. Links represent connections between pairs of entities, such as co-authorship in authorship networks, friendship in social networks, interaction in PPI networks. Links may contain descriptive information, such as names, sources, weights. $|V|$ and $|E|$ denote number of nodes and number of links respectively. The link between two nodes $x \in V$ and $y \in V$

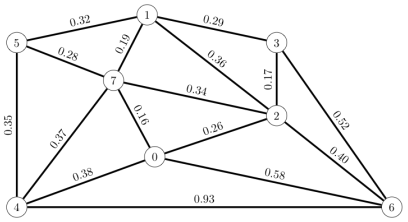
(a) A homogeneous graph



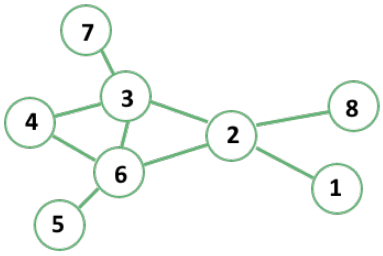
(b) A heterogeneous network



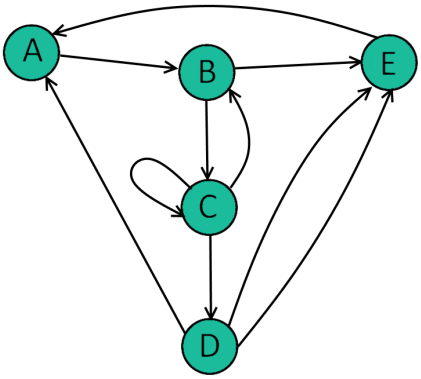
(c) An attributed graph



(d) A weighted graph



(e) An undirected graph



(f) A directed graph

Figure 1.1: Homogeneous vs heterogeneous graph: Different types of nodes are indicated with different colors.

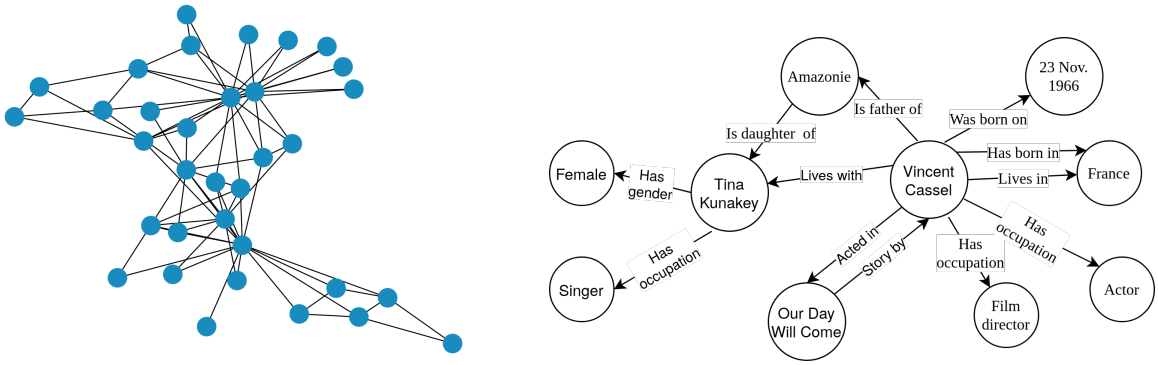
is noted as $e_{xy} \in E$. e_{xy} is called a directed link if the direction of the link goes from the node x to node y and undirected link otherwise. For a directed link, $e_{xy} \in E$, x and y are termed as source and destination respectively. If the node x connects itself then the formed link $e_{x,x} \in E$ is called a loop. Two nodes $x, y \in V$ are **neighbour** to each other if they are linked, *i.e.*, $e_{xy} \in E$. The neighbours of the node x collectively form a neighbour set Γ_x of the node. The number of neighbours for a node, x is termed as **degree** of the node and denoted by $|\Gamma_x|$. An **adjacency matrix**, A is a square matrix representation of the graph, G where an element A_{xy} indicates whether the nodes x and y are adjacent or not in the graph.

$$A_{xy} = \begin{cases} 1 & \text{if } e_{xy} \in E \\ 0 & \text{if } e_{xy} \notin E \end{cases}$$

A **path**, $P_{x,z}$ is a sequence of links $e_{xy} - e_{ya} - .... - e_{tz}$ between nodes $x, z \in V$ where $x, y, a, ..., t, z \in V$ and $e_{xy}, e_{ya}, ..., e_{tz} \in E$. The number of links in $P_{x,z}$ is called length of the path or simply **path length**. A path with length l between two nodes x and y is denoted as $P^{l}_{x,y}$. There may exist multiple paths in G between a pair of nodes x, z. Among the paths, the path with smallest path length is called the shortest path for the pair and denoted as $P^{shortest}_{x,z}$. There could be multiple shortest paths of smallest length between pair of nodes. A graph is connected if there exist one or more path between each pair of nodes. The **clustering coefficient** is defined as the ratio of number of triangles and expected number of triangle passing through a node. If t_x is the number of triangles passing through a node $x \in V$ then the clustering coefficient, $CC(x)$ of x is defined as Equation 1.1.

$$CC_x = \frac{2 \times t_x}{|\Gamma x| \left(|\Gamma x| - 1\right)} \tag{1.1}$$

Here, t_x is computed based on links among neighbours of the node, x. The graph G is called **homogeneous graph** if all of its nodes have the same type (Figure 1.1a). A **heterogeneous graph** contains two or more types of nodes (Figure 1.1b).



(a) A simple graph

(b) A knowledge graph

Figure 1.2: Simple graph and knowledge graph

If E contains only directed links then G is called a **directed graph** (Figure 1.1f). On the other hand, all links in E are undirected in an **undirected graph** (Figure 1.1e). A **mixed graph** contains both directed and undirected links. Based on link weights, a graph can be a **weighted graph**, where a weight (simply a value) is assigned to each link (Figure 1.1d). Otherwise, the graph is called an **unweighted graph**. The weight of a link may represent a cost, length or capacity. A graph may allow multiple links between a pair of nodes and the occurrence is called multi-link.

1.1.2 Simple *versus* knowledge graph

We consider a graph without directed links, loops, multi-links, link attributes as a simple graph (Figure 1.2a). Formally, we define a simple graph $G = (V, E)$ where $\forall_{(x,y)}(x, y) \in E \implies (x \neq y)$.

A knowledge graph (KG) allows to encode more semantic information (Figure 1.2b). It has been introduced in the Semantic Web domain to build a machine readable "Web of data". KGs are usually represented using the Resource Description Framework (RDF) data model [Kly04]. It allows the occurrence of multi-links, loops, directed links, mandatory link names and auxiliary

node and link attributes [Pau17]. Few examples of real-world KGs are mentioned in Table 1.1. Formally, a knowledge graph is defined as $KG = (V, R, Q)$ tuple [GDX22], where

— V is a set of nodes representing entities;

— R is a set of relation names;

— $Q \subseteq V \times R \times V$ is a set of facts (positive triples);

The facts are formatted in triple format, $(h, r, t) \in Q$ where $h \in V$ and $t \in V$ are called the head and tail entities respectively and $r \in R$ is a directed relation from h to t.

Based on structural properties, a relation can be symmetric or anti-symmetric, inverse of another relation, and composition of multiple relations [SDNT18a]. Formally, if $\{(h, t)\}$ is the set of entity-pairs for a relation r, then the relation $r \in R$ is

— anti-symmetric, if $\forall_{(h,t)}(h, r, t) \in Q \implies (t, r, h) \notin Q$.

— symmetric, if $\forall_{h,t}(h, r, t) \in Q \implies (t, r, h) \in Q$.

— composition of two relations, $r_1,\ r_2 \in R$, if $\forall_{h,t}(h, r_1, z) \in Q \wedge (z, r_2, t) \in Q \implies (h, r, t) \in Q$.

— inverse of another relation, $r_1 \in R$, if $\forall_{h,t}\{(h, r, t) \in Q \implies (t, r_1, h) \in Q\}$;

— one-to-many, if $(h, r, t_1) \in Q \wedge (h, r, t_2) \in Q \wedge .... \wedge (h, r, t_n) \in Q$;
many-to-one, if $(h_1, r, t) \in Q \wedge (h_2, r, t) \in Q \wedge .... \wedge (h_n, r, t) \in Q$.

The definition of KG comes with two important terms: closed-world assumption (CWA), and open-world assumption (OWA). Under CWA, an observed triple in a KG is necessary false while the triple may be true or false under OWA [NMTG15, HBC+21]. The unobserved triples are simply unknown under OWA. To illustrate, the unobserved triple (Tina Kunakey, Was_born_in, France) is false under CWA in the toy example of KG (Figure 1.2b). In contrast, the unobserved triple is neither true nor false under OWA.

Table 1.1: Few examples of real-world knowledge graphs. The second column gives the year of release of each KG. The fourth column describes the domain of entities and relations.

KG	Year	Purpose	Domain
FreeBase [Goo13]	2007	Academic	Generic
DBpedia [LIJ+15]	2015	Academic	Generic
YAGO [SKW07]	2008	Academic	Generic
KBpedia [Cog16]	2016	Academic	Generic
Google [Sin12]	2012	Commercial	Generic
Yahoo [Tor12]	2014	Commercial	Generic
Amazon [BCG+18]	2018	Commercial	E-commerce
Facebook [NGJ+19]	2013	Commercial	Social network
Linkedin [He16]	2016	Commercial	Social network

1.2 Link prediction in simple graphs

1.2.1 Problem formulation

Given a simple graph at a particular time t, the link prediction problem is defined as discovering or inferring a set of links in the graph at time $t + \Delta t$ [IAST20a]. The problem can be illustrated with a simple graph in Figure 1.3, where circles represent nodes and lines represent links. Black solid lines represent observed links and red dashed lines represent missing links in the current graph. Figure 1.3a shows the snapshot of the graph at time t. The link prediction problem aiming to predict the appearance of the missing links as observed links in the graph in $t + \Delta t$, as illustrated in Figure 1.3b.

(a) Graph at time t

(b) Graph at time $t + \Delta t$

Figure 1.3: Illustration of link prediction problem in a simple graph.

To evaluate the performance of a link prediction approach, the set of observed links in a graph dataset is split into a training set and a test set. These observed links in the test set form a positive test set representing missing links. Some unobserved links are also selected to form a negative test set and added to the test set. Hence, a test set consists of a positive test set and a negative test set. A training graph is then built based on the training set. The prediction performance of the approach is evaluated with two metrics: precision and area under the ROC Curve (AUC). Precision describes the fraction of missing links which are accurately predicted [PZLH16]. To compute the precision, the link prediction scores of all links from the test set are computed based on the training graph and the test links are ranked in decreasing order of their prediction scores. If L_r is the number of observed links from the positive test set among the L-top ranked predicted (test) links then the precision is defined as Equation 1.2.

$$Precision = \frac{L_r}{L} \tag{1.2}$$

An ideal prediction approach has a precision of 1.0 that means all the missing links are accurately predicted. On the other hand, the metric AUC is defined as the probability that a randomly chosen positive test link has a higher prediction score than a randomly chosen negative test link [PZLH16]. Suppose, n positive and an equal number of negative test links are chosen from positive and negative test sets. If n_1 is the number of positive test links having a higher score than negative test links and n_2 is the number of positive test links having equal prediction score as negative test links then AUC is defined as Equation 1.3.

$$AUC = \frac{n_1 + 0.5n_2}{n} \tag{1.3}$$

The link prediction problem can be also be considered as a binary classification problem [KSSB20] where an unobserved link is classified as existent or non-existent. Based on the true and predicted

classes of links, four different outcomes may happen: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). Based on these outcomes, the following standard evaluation metrics are defined to evaluate a link prediction approach [KSSB20].

$$False\ positive\ rate(FPR) = \frac{FP}{FP + TN} \tag{1.4}$$

$$False\ negative\ rate(FNR) = \frac{FN}{TP + FN} \tag{1.5}$$

$$Accuracy(ACC) = \frac{TP + TN}{TP + FN + TN + FP} \tag{1.6}$$

$$Recall = \frac{TP}{TP + FN} \tag{1.7}$$

$$Precision = \frac{TP}{TP + FP} \tag{1.8}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{1.9}$$

The problem of link prediction in simple graphs is widely studied. Numerous link prediction approaches for simple graphs exist in the literature, ranging from simple similarity-based approaches to more recent graph embedding-based approaches. As illustrated in Figure 1.4, state-of-the-art approaches are broadly categorized in four major categories: similarity-based, probabilistic and statistical-based, classifier-based, and embedding-based approaches [WSGG22]. Few approaches

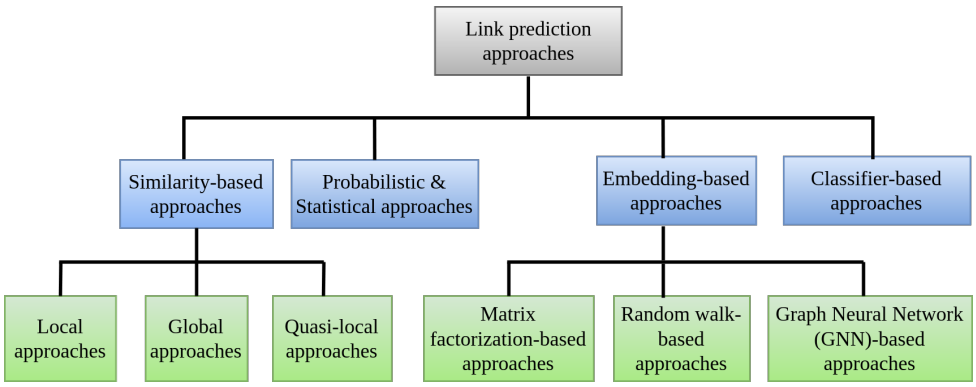


Figure 1.4: Taxonomy of link prediction approaches for simple graphs [IAST21b]

from each category are discussed in Sections 1.2.2-1.2.5.

1.2.2 Similarity-based approaches

For simple graphs, widely studied approaches are similarity-based approaches for link prediction [MBC16, GCL+17]. These approaches are designed based on the common principle that two similar nodes in a graph are most likely to be connected [MBC16, IAST20a, WSGG22]. The similarity-based approaches assume that the similarity of two nodes is positively correlated with the number of paths or amount of resources transferred between them [LJLB17]. The similarity function is defined based on nodes attributes and/or structural information to compute a similarity score between a pair of nodes. Generally, similarity-based link prediction approaches consists of three major steps: (1) computing similarity scores for node pairs of unobserved links; (2) ranking the links in decreasing order of their similarity scores; and (3) selecting few top ranked

links as predicted links. Defining good similarity functions is a non-trivial task due to their heuristic nature and the definition of the similarity function varies domain to domain [MBC16]. Due to this heuristic nature, there are many similarity-based approaches in the literature which are categorized into three categories: local, global and quasi-local similarity-based approaches. **Local approaches** use local neighbourhood information of nodes to compute the similarity of two nodes [LL10, MBC16]. *Common Neighbours (CN)* is one of the primitive similarity-based approaches for link prediction that assumes two nodes more likely to be linked share more neighbours [LW71]. The similarity function is defined based on the number of common neighbours between two nodes (Equation 1.10.

$$S^{CN}(x, y) = |\Gamma x \cap \Gamma y| \tag{1.10}$$

Here, Γx and Γy are the immediate neighbour sets of nodes x and y respectively. An illustration of computing CN metrics is given in Figure 1.5. CN is very simple but it shows low

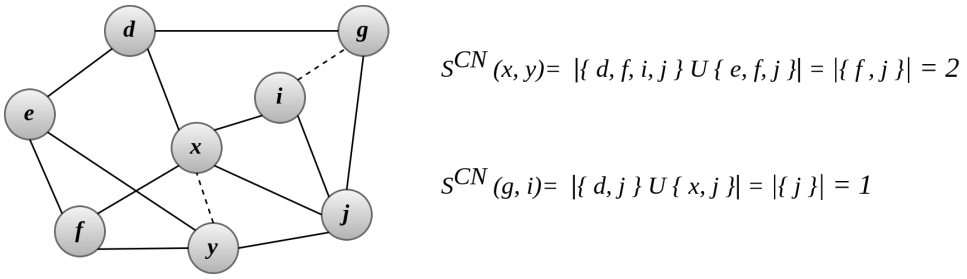


Figure 1.5: Illustration of computing CN similarity metrics. The solid lines represent observed links and dotted lines represent unobserved links to be predicted.

accuracy in many real-world graphs as overlooks individual contribution of neighbours [TXZ14]. Some improvements of CN exist in the literature those consider individual contribution of each neighbour to their similarity functions. The *Adamic-Adar Index (AA)* [AA03] was designed to improve CN by penalizing each common neighbour by its degree and the similarity is defined as Equation 1.11.

$$S^{AA}(x, y) = \sum_{z \in \Gamma_x \cap \Gamma_y} \frac{1}{log\,|\Gamma_z|} \tag{1.11}$$

The *Resource Allocation Index (RA)* was designed based on the resource allocation process that models the transmission of resources between two nodes [ZLZ09]. RA penalizes common neighbours by their degree as in Equation 1.12. The assumption of transferring the amount of resources equally to all common neighbour is not true for many real-world graphs.

$$S^{RA}(x, y) = \sum_{z \in \Gamma_x \cap \Gamma_y} \frac{1}{|\Gamma_z|} \tag{1.12}$$

Resource Allocation Based on Common Neighbour Interactions (RA-CNI) is an improvement of the RA which considers the return of resources in the opposite direction in addition to transferred resources [ZZYY14]. The similarity function of RA-CNI is defined as Equation 1.13.

$$S^{RA-CNI}(x, y) = \sum_{z \in \Gamma x \cap \Gamma y} \frac{1}{|\Gamma z|} + \sum_{e_{i,j} \in E; |\Gamma_i| \leq |\Gamma_j|; i \in \Gamma_x, j \in \Gamma_y} \left(\frac{1}{|\Gamma_i|} - \frac{1}{|\Gamma_j|} \right) \tag{1.13}$$

The *Jaccard Index (JA)* defines the similarity (Equation 1.14) as the ratio of common neighbours in the complete set of neighbours for two nodes [Jac01]. It is a widely used index to compare the similarity and diversity of two data sets in information retrieval systems. It is considered as a normalization of CN where the similarity score is penalized for each non-common neighbour.

$$S^{JA}(x,y) = \frac{|\Gamma_x \cap \Gamma_y|}{|\Gamma_x \cup \Gamma_y|} \tag{1.14}$$

The *Salton Index (SA)* is a variant of JA which considers common and non-common neighbours to compute the end nodes similarity [SM83]. SA gives a similarity score of approximately double as JA in most real-world networks [H$^+$89]. SA is also known as the cosine similarity that is defined based on cosine of the angle between rows of adjacency vector for end nodes as Equation 1.15.

$$S^{SA}(x,y) = \frac{|\Gamma_x \cap \Gamma_y|}{\sqrt{|\Gamma_x| \times |\Gamma_y|}} \tag{1.15}$$

Sørensen Index (SO), another common neighbour based approach, was originally designed to compute the similarity between two ecological community data [Sor48]. It describes the overall proportion of common neighbours from a local perspective. The similarity function is defined as the harmonic mean of the proportion of shared neighbours in the neighbour sets of end nodes (Equation 1.16).

$$S^{SO}(x,y) = \frac{2 \times |\Gamma_x \cap \Gamma_y|}{|\Gamma_x| + |\Gamma_y|} \tag{1.16}$$

Ravasz *et al.* designed *Hub Depressed Index (HDI)* and *Hub Promoted Index (HPI)* during analysis of the topological modularity of a metabolic network [RSM$^+$02]. They described the networks in a hierarchical architecture where two highly internally connected modules are often isolated from each other. The similarity function in HDI(HPI) is defined to promote link formation between low(high)-degree nodes and hubs as Equation 1.17(1.18).

$$S^{HDI}(x,y) = \frac{|\Gamma_x \cap \Gamma_y|}{min(|\Gamma_x|, |\Gamma_y|)} \tag{1.17}$$

$$S^{HPI}(x,y) = \frac{|\Gamma_x \cap \Gamma_y|}{max(|\Gamma_x|, |\Gamma_y|)} \tag{1.18}$$

The *Local Leicht-Holme-Newman Index (LLHN)* [LHN06] defines the similarity function as the ratio of common neighbours of node pairs and the expected number of common neighbours (Equation 1.19).

$$S^{LLHN}(x,y) = \frac{|\Gamma_x \cap \Gamma_y|}{|\Gamma_x| \times |\Gamma_y|} \tag{1.19}$$

The concept of expected number of common neighbours for node pairs is computed using the concept of configuration module [MR95]. The *Local Affinity Structure Index (LAS)* [SHY$^+$17] approach introduces the concept of community structure into its similarity function. Based on the belief that the similarity of two nodes in a graph is correlated with the density of the local structure, LAS defines the similarity function as Equation 1.20.

$$S^{LAS}(x,y) = \frac{|\Gamma_x \cap \Gamma_y|}{|\Gamma_x|} + \frac{|\Gamma_x \cap \Gamma_y|}{|\Gamma_y|} \tag{1.20}$$

The above discussed CN variants consider only the common neighbours of end nodes of a link and their degree information in defining the similarity functions. However, information about interconnections among common neighbours also provide useful information in designing a similarity

function. The *Individual Attraction (IA)* [DKWW11] defines a similarity function considering the relationship among common neighbours along with their individual properties for link prediction. It introduces the concept of inner link that connect two common neighbours. This index assumes that the likelihood of a link between a pair of nodes is high if the number of inner links is high. This similarity is defined as Equation 1.21.

$$S^{IA}(x,y) = \sum_{z \in \Gamma_x \cap \Gamma_y} \frac{\left|e_{z,\Gamma_x \cap \Gamma_y}\right| + 2}{|\Gamma_z|} \tag{1.21}$$

Here, $\left|e_{z,\Gamma_x \cap \Gamma_y}\right|$ is the number of links between a common neighbour, z and all other common neighbours from the set $\Gamma_x \cap \Gamma_y$ Dong *et al.* (2011) [DKWW11] also developed Simple Individual Attraction Indices (SIA), a faster version of IA, leverages the number of all inner links among neighbours of each common neighbour for defining the similarity function as Equation 1.22.

$$S^{SIA}(x,y) = \sum_{z \in \Gamma_x \cap \Gamma_y} \frac{\left|e_{\Gamma_x \cap \Gamma_y}\right| + 2}{|\Gamma_z|\,|\Gamma_x \cap \Gamma_y|} \tag{1.22}$$

Here, $\left|e_{\Gamma_x \cap \Gamma_y}\right|$ is the number of links which connect common neighbors. The *CAR-based Indices (CARs)* [CALR13] define two similarity functions using up to hop-2 neighbourhood or level-2 link information that connects two common neighbours as Equations 1.23-1.24.

$$S^{CAR1}(x,y) = \sum_{z \in \Gamma_x \cap \Gamma_y} 1 + \frac{|\Gamma_x \cap \Gamma_y \cap \Gamma_z|}{2} \tag{1.23}$$

$$S^{CAR2}(x,y) = \sum_{z \in \Gamma_x \cap \Gamma_y} \frac{|\Gamma_x \cap \Gamma_y \cap \Gamma_z|}{|\Gamma_z|} \tag{1.24}$$

Based on the concept of clustering coefficient, two similarity approaches exist in the literature. The first one is *Clustering Coefficient for Link Prediction (CCLP)* [WLWG16] that defines the similarity function based on clustering coefficients of common neighbours between end nodes of a link (Equation 1.25).

$$S^{CCLP}(x,y) = \sum_{z \in \Gamma_x \cap \Gamma_y} CC_z \tag{1.25}$$

Here, CC_z is the clustering coefficient (computed by Equation 1.1) of a common neighbour z. CCLP make link information among the common neighbours, except the seed nodes pair. *Node and Link Clustering (NLC)* [WLWJ16] is the other one that quantifies the contribution of common neighbours based on both their node and link clustering coefficients as Equation 1.26.

$$NLC(x,y) = \sum_{z \in \Gamma x \cap \Gamma y} \Big(\frac{CN(x,z)}{|\Gamma z| - 1} \times CC_z + \frac{CN(x,z)}{|\Gamma z| - 1} \times CC_z \Big) \tag{1.26}$$

All of the above discussed CN variants fail to predict a link between two distant nodes, *i.e.*, end nodes do not have any common neighbour. To overcome this problem, *Preferential Attachment Index (PA)* defines a similarity function based on the power-law distribution where a node is attached preferentially to the node that is already well connected [BA99]. This phenomenon leads to the generation of the concept known as "rich-get-richer". Based on this rich-get-richer concept, the similarity function describes that the likelihood of a link between two high degree nodes is higher than the likelihood of a link between two low-degree nodes as Equation 1.27.

$$S^{PA}(x,y) = |\Gamma_x| \times |\Gamma_y| \tag{1.27}$$

All in one, the local approaches only consider local (up to second hop) neighbourhood information in defining their similarity functions. The local approaches are fast and scalable as they use only local neighbor information. However, many links could connect two nodes which are far from each other in large graphs and the local approaches are less accurate in these cases [LL10]. Researchers studied **global approaches** to include whole graph topological information of graphs in designing similarity functions. These approaches are able to compute similarity score between two nodes which are situated beyond second hop.
Negated Shortest Path (NSP) is the simplest one [LNK07] that defines that similarity function based on the shortest path between a pair of nodes (Equation 1.28). Dijkstra's algorithm can be used to compute the shortest path between a pair of nodes.

$$S^{NSP}(x,y) = -Length(P_{x,y}^{shortest}) \tag{1.28}$$

where $Length()$ denotes the length of a path. NSP fails to provide high accuracy in many real-world graphs as it considers only one (shortest) path and ignores other paths if any.
The influence of all possible paths between two end nodes in a pair is summed by the *Katz Index (KI)*, which incrementally penalizes paths based on their length [Kat53]. If $Ps_{x,y}^{l}$ is the set of length-l paths between end nodes of a pair (x,y), The similarity matrix in KI is defined as Equation 1.29.

$$S_{x,y}^{KATZ} = \sum_{l=1}^{\infty} \beta^l \left| Ps_{x,y}^{l} \right| = \sum_{l=1}^{\infty} \beta^l A_{x,y}^{l} \tag{1.29}$$

Note that each cell in a l^{th} powered adjacency matrix gives the number of length-l paths between the corresponding end nodes. β is a user-defined parameter which is used to increase influences of longer paths to overall similarity score. The value of this parameter ranges from 0 to 1, and a higher number indicates greater influence.
The *Random Walks Index (RW)* [LL10] was developed based on the well-known random walk process [Pea05] on graphs. Given a beginning node in a graph, a random walker moves to a random neighbor of that node and then repeats this process for each node until the walker reaches to target node. Transition probability matrix (TM) is the heart of a random walker which is defined based on the adjacency matrix (A) of the graph. Each cell in TM is defined as $TM_{i,j} = \frac{A_{i,j}}{\sum_k A_{i,k}}$. Given the starting node is $x \in V$ for a random walker and $Prob^x$ is the probability vector of reaching other nodes from x. The probability of reaching each node from x can be iteratively computed as Equation 1.30.

$$Prob^x(t) = TM^T Prob^x(t-1) \tag{1.30}$$

Here, $()^T$ denote the matrix transpose operation. Equation 1.30 is repeated until convergence. The convergence is defined with respect to a stopping criteria, such as $\sum_{i \in V}(Prob_i^x(t) - Prob_i^x(t-1))^2 \leq \epsilon$ where ϵ is a threshold. After convergence, the similarity function for the pair of nodes x and y is defined as $S^{RW}(x,y) = Prob_y^x$ is defined as the similarity score between a pair of nodes x and y. During walking, a random walker may visit far away from its target. To reduce this problem, Tong *et al.* improved RW in *Random Walks with Restart (RWR)* approach where the random walker moves to its neighbours with α or returns to the starting node with a probability $1-\alpha$ [TFP06]. The probability of reaching each node from x can be iteratively computed as Equation 1.31.

$$Prob^x(t) = \alpha TM^T Prob^x(t-1) + (1-\alpha)s^x \tag{1.31}$$

where $s^x \in \mathbb{R}^{|V|}$ is the seed vector for the node x with all elements are 0 except s_x^x=1. The above Equation 1.31 is optimized in the same way as in RW. Considering the anti-symmetry

nature of Equation 1.31, the similarity function for two end nodes x and y of a link is defined as $S^{RWR}(x,y) = Prob_y^x + Prob_x^y$.
Average Commute Time (ACT) is another improvement of the RW [LL10], which is defined as the average number of steps a walker takes to reach the target node from the source node and return back to the source node. The similarity of two nodes in AC is computed based on pseudo-inverse of the Laplacian matrix L^+ as Equation 1.32.

$$S_{x,y}^{ACT} = \frac{1}{L_{x,x}^+ + L_{y,y}^+ - 2L_{x,y}^+} \tag{1.32}$$

The global approaches offer high link prediction performance in many graphs, especially when links are formed at distances greater than two in large scale graphs. However, global approaches are computationally very expensive and are infeasible to be deployed in large scale graphs as they use global topological information contained in graphs. To make a trade-off between prediction accuracy and computational time, **quasi-local approaches** use neighborhood information up to a predefined hop threshold. Quasi-local approaches are almost as efficient to compute as local approaches but also high performer like global approaches. Quasi-local approaches takes into account the similarity between pair of nodes which are beyond neighbours of neighbours, but not too far away. *The Local Path Index (LPI)* is a simple quasi-local approach which is a limited version of the Katz Index indeed [LJZ09]. The similarity matrix is computed as Equation 1.33.

$$S_{x,y}^{LPI} = \sum_{i=2}^{l} \beta^{i-2} A_{xy}^i \tag{1.33}$$

where $l > 2$ is the maximal path length and β is an attenuation factor (same in Katz Index).
Local Random Walk (LRW) [LL10] is a quasi-local version of the global RW approach where a random walker walks for a limited number of steps rather than using the concept of convergence to stop walking. The similarity function in LRW is defined as Equation 1.34.

$$S^{LRW}(x,y) = \frac{|\Gamma x|}{2\,|E|} Prob_y^x(t) + \frac{|\Gamma y|}{2\,|E|} Prob_x^y(t) \tag{1.34}$$

The *Resource Allocation along Local Path (RALP)* [BHT11] is an extension of the LP approach where the resource allocation process is introduced. The similarity function is defined as Equation 1.35.

$$S^{RALP}(x,y) = \sum_{z \in \Gamma_x \cap \Gamma_y} \frac{1}{|\Gamma z|} + \sum_{(i,j) \in l_{x \to y}} \frac{1}{|\Gamma i| \times |\Gamma j|} \tag{1.35}$$

Here, $l_{x \to y} = (x - i - j - y)$ is a simple path between x and y.
The *Third-Order Resource Allocation Based on Common Neighbor Interactions (RA-CNI)* [ZZYY14] is the quasi-local version of the original RA approaches where the similarity function is defined based on the hop-3 common neighbours interactions as Equation 1.36.

$$S^{RA-CNI}(x,y) = \sum_{z \in \Gamma_x \cap \Gamma_y} \frac{1}{|\Gamma z|} + \sum_{(u,v) \in E; u \in \Gamma x, v \in \Gamma y; |\Gamma u| \geq |\Gamma v|} [\frac{1}{|\Gamma v|} - \frac{1}{|\Gamma u|}] \tag{1.36}$$

To sum up, the local approaches are fast and scalable, but they are less accurate when predicting a link between two distant (shortest distance more than 2) nodes in a graph. On the other hand, the global approaches offer high link prediction performance when links are formed at distances greater than two, but they are computationally very expensive and are infeasible to be deployed in large scale graphs. Quasi-local approaches make a trade-off between prediction accuracy and computational time.

1.2.3 Probabilistic and statistical approaches

Probabilistic and statistical approaches usually assume that the graph has a certain structure. A probabilistic model is then developed that best fits with the graph structure and the model parameters are tuned using a statistical method. The model is then used to predict the link existence probability between a pair of nodes in the graph. These approaches work in four steps: (1) training the probabilistic models to estimate their parameters; (2) computing probability values of unobserved links; (3) ranking the unobserved links in decreasing order of their probability values; (4) selecting few top-ranked links as predicted links.
The *Cycle Formation Model* [Hua10] is a cycle structure-based model that formulates link prediction by generalizing the clustering coefficient of nodes. Based on the number of cycles that will be obtained by including this link in the graph, the link probability is computed. The length of the cycle is referred to as the degree of clustering coefficient. A generalized clustering coefficient of degree k is defined in Equation 1.37

$$C(k) = \frac{\#Cycles(k)}{\#Path^k} \tag{1.37}$$

where $\#Cycles(k)$ denotes number of k-length cycles and $\#Path^k$ denotes number of k-length paths in the graph. The model parameters related to k-length cycle formation are estimated based on a series of k generalized clustering coefficients. The expected clustering coefficient of degree k is estimated as Equation 1.38

$$f(c_1, c_2, ..., c_k) = \sum_i |G_i| \, Pr(G_i) Pr(e_{1,k+1} \in E | G_i) \tag{1.38}$$

where c_k is the clustering coefficient of degree k, G_i is the set of possible graph patterns with i nodes, $Pr(e_{1,k+1} \in E$ denotes the conditional probability of a k-length path to become a k-length cycle. The expected clustering coefficient parameters are then used to compute the probability of a link between a pair of nodes as Equation 1.39.

$$Prob^k(x, y) = \frac{c_1 \sum_{i=2}^{k} c_i^{\#Path^i_{x,y}}}{c_1 \sum_{i=2}^{k} c_i^{\#Path^i_{x,y}} + (1 - c_1) \sum_{i=2}^{k} (1 - c_i)^{\#Path^i_{x,y}}} \tag{1.39}$$

Here, $\#Path^i_{x,y}$ denotes the number of k-length paths between nodes x and y in the graph. Studies show that many real-world graphs are hierarchical in nature, such as metabolic graphs, internet domain graphs, actor graphs [RB03, CMN08].
The *Hierarchical Structure Model (HSM)* [CMN08] is another probabilistic approach for link prediction that focuses on the hierarchical structure of graphs. In this approach, a hierarchical graph is represented by a dendrogram with $|V|$ leaves representing nodes and $|V| - 1$ internal nodes correspond to link probability between its descendants. In the dendrogram, the link probability of a pair of nodes is the probability value associated with the lowest common ancestor of the two nodes. The *local probabilistic* approach [WSP07] combines co-occurrence probability features with topology and semantic features for link prediction in graphs. The approach uses a local probabilistic model based on Markov Random Fields (MRF) to compute co-occurrence probability of nodes or simply link probability. The approach identifies the central neighbour set of each end node. The sets are then used to train the local probabilistic model to maximize the co-occurrence probability. The training process is considered as a maximum entropy optimization problem.

The probabilistic approaches offer high link prediction performance in many real-world graphs with certain structures but take very high computational time like global similarity-based approaches [WXWZ15, MBC16, DAHS+20]. As a result, these approaches are not applicable to large scale graphs.

1.2.4 Supervised approaches

The supervised approaches consider link prediction as a binary classification problem where an unobserved link is classified into either existent or non-existent link [SNM11]. Overall, these approaches extract features of nodes and links from a graph and train a supervised machine learning algorithm with the features of positive and negative links to classify unknown links. The success of these approaches depends on the goodness of the handcrafted feature set. There exist few supervised link prediction approaches in the literature. Scellato *et al.* (2011) presents a supervised approach for link prediction in social networks [SNM11]. The approach defines a feature set based on attributes (e.g. user's current and visited locations) and structural (common neighbourhood) information of nodes (users) in social networks. The approach trains a Naive Bayes/ Random forest classifier with these features to classify an unobserved link between two users into friendship and unfriendship link. Wohlfarth & Ichise developed a supervised approach for link prediction in co-authorship networks [WI08]. The approach extracts a set of features based on structural and node attribute information. The structural features include few local and global similarity-based metrics, research keywords and publication venues of researchers. For supervised machine learning algorithm, the approach uses the J48 decision tree based classifier. Pavlov & Ichise [PI07] considered link weight and few local and global similarity-based metrics to define the feature set. Another supervised approach for link prediction in co-authorship graphs was provided by Hasan *et al.* (2006) [AHCSZ06] where authors extract few node attribute-based features and similarity-based matrices to design the feature set. As for the classifier selection, they consider support vector machine (SVM), decision tree (DT), KNN, and Naive Bayes algorithms. The above discussed approaches depend on both node attributes and graph structure for defining feature sets. However, node attributes are not always available due to data privacy concern. Recently, researchers have been studying defining a rich structural feature. Li *et al.* (2021) [LWF+21], Kumari *et al.* (2022) [KBS+22], Cukierski *et al.* (2011) [CHY11] define feature sets by considering local and global similarity-based metrics. Kerkache *et al.* (2022) defines the feature set based on community-based and local similarity-based metrics [KSBBSTA22]. These supervised approaches offer good link prediction performance in many graphs from various domains.

1.2.5 Embedding-based approaches

The success of most categories of link prediction approaches discussed above (Sections 1.2.2 - 1.2.4) is dependent on the carefully engineered similarity function or feature-set and is limited by their inflexibility and high cost [ZCH+20]. In recent years, researchers studied graph embedding methods to automatize feature extraction in graphs in link prediction approaches. An embedding method represents nodes of a graph in a low-dimensional vector space preserving graph structural information. In other words, the embedding method maps the graph to a vector space, while preserving its structural properties. Figure 1.6 illustrates the embeddings of a toy example graph. The learned embeddings of nodes in the example graph aims to keep that neighbour nodes in the graph also close in the embedding or latent space. The vector representations of nodes

[4]Source: https://towardsdatascience.com/embedding-graphs-with-deep-learning-55e0c66d7752

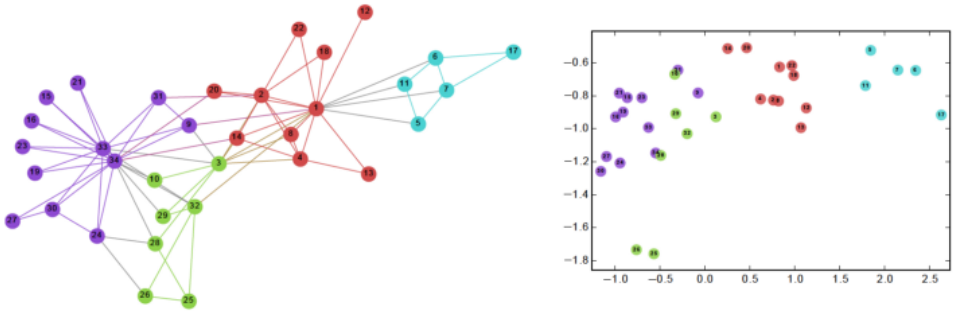


Figure 1.6: Illustration of embedding of a toy graph (left panel) on a 2-dimensional vector space (right panel). Neighbouring nodes are colored with a distinguishable color.[4]

allow to define mathematical operations to extract latent graph features for link predictions. Based on the learned node embeddings, link prediction approaches define various methods to compute link probability between two nodes. The simple binary operator-based methods do binary operations, such as sum, average, Hadamard product, on learned embeddings of end nodes of a link to generate the link embedding and the magnitude of the link embedding is the link score [GL16]. Some link probability methods are designed based on advanced neural network (NN) architectures where NN-based prediction models are used to predict the probability of links based on their end nodes embeddings [CHW22].

Matrix-factorization based approaches

As graphs can be represented with adjacency matrix, matrix factorization based link prediction approaches consider the link prediction problem as a matrix completion problem. The basic idea is to decompose the adjacency matrix of a graph into low-rank matrices, and reconstruct the graph with minimum error [ME11, CXW+19]. These approaches use traditional matrix decomposition techniques to learn latent features or embeddings of nodes and the embeddings are used in the subsequent link prediction task. If a factorization technique factorizes the adjacency matrix, A of a graph, *i.e.*, $A = U\Lambda U^T$ for some $U \in \mathbb{R}^{N\times k}$, $\Lambda \in \mathbb{R}^{k\times k}$ where $N = |V|$, a link prediction approach defines the optimization task as Equation 1.40 [ME11].

$$\min_{U,\Lambda,b} \frac{1}{|E|} \sum_{(x,y)\in E} l(A_{xy}, L(\mathbf{x}^T\Lambda\mathbf{y} + b_x + b_y)) + Reg(U,\Lambda) \tag{1.40}$$

Here, L denotes a link scoring function, l is the loss function, Reg is the regularizer, $\mathbf{x}, \mathbf{y} \in U$ are latent representations or embeddings of nodes x, y extracted from the learned representation matrix U, b_x and b_y are the biases related to the nodes.

Based on this optimization task, there exist few factorization based link prediction approaches. Menon and Elkan [ME11] adopt the pairwise loss concept from SVMRank classification model [Joa02] to design Matrix Factorization (MF) approach where the loss is defined as the difference between positive/observed links scores and negative/ unobserved links scores. Given the negative link set, E', the approach defines the optimization function as Equation 1.41.

$$\min_{U,\Lambda,b} \frac{1}{|E|\,|E'|} \sum_{(x,y)\in E,(x',y')\in E'} l(1, \mathbf{x}^T\Lambda\mathbf{y} - \mathbf{x'}^T\Lambda\mathbf{y'} + b_x + b_y)) + Reg(U,\Lambda) \tag{1.41}$$

As MF factorizes the adjacency matrix of a graph, it considers only local neighbourhood information for link prediction. Cao *et al.* designed **GraRep** [CLX15] to factorize a k-order

transition probability matrix to consider k-hop neighbourhood information for link prediction. Recently, the non-negative matrix factorization has been widely studied in graphs for the link prediction task due to its simplicity. Considering the adjacency matrix as a non-negative matrix, the factorization task is to learn two matrices $U^{N\times k}$ and $V^{k\times N}$ so that their product is very close to A, *i.e.*, $A \equiv UV$ for $U \geq 0, V \geq 0$ [LS00]. The matrix U gives k-dimensional embeddings of nodes in the graph. Non-negative Matrix Factorization (GWNMF) [CXW+19], Fusing Structure and Sparsity-constrained via Deep Non-negative Matrix Factorization (FSSDNMF) [CWFJ22], Manifold regularization and Sparse learning (MS-RNMF) [CXW+20] are few non-negative factorization-based link prediction approaches. Chen *et al.* [CXW+19] proposed GWNMF to integrate local neighbourhood and link weight information to improve link prediction performance. GWNMF presents a weighted non-negative matrix factorization model to learn embeddings of nodes. By utilizing the observed link information, FSSDNMF maps the adjacency matrix of a graph to a multi-layer feature space. The factorization model fully utilizes the observed link information of each layer. To remove noise, FSSDNMF employs the l_2 norm. MS-RNMF is another factorization-based approach that presents a manifold regularization strategy to solve the presence of noise and spurious links in a graph. The approach also presents a k-medoids algorithm to encode global structural information in nodes embeddings.

Studies show that link prediction approaches based on matrix factorization improve prediction performance. However, the results are unstable due to noise, bias, and variance [WC16, DAHS+20]. Another downside of these approaches includes requiring high computational time and memory. Therefore, these approaches are not scalable to large scale graphs.

Random-walk based approaches

Motivated by SkipGram model [MCCD13] model for word embedding methods, random-walk based approaches such as Deepwalk [PARS14], LINE [TQW+15], Node2Vec [GL16] learn embeddings of a graph based on the concept of maximizing co-occurrence probability of neighbour nodes. Given the current node is v, the co-occurrence probability of all its neighbours Γ_v visited by a random walker is estimated as Equation 1.42.

$$Pr(\Gamma_v|\Phi(v)) \tag{1.42}$$

Here, $Pr()$ is the co-occurrence probability, $\Phi : v \in V \rightarrow \mathbb{R}^{|V|\times d}$ is a mapping function that represents the embeddings of size d of nodes. Each random-walk based approach defines an objective function to maximize the co-occurrence probability of neighbours for optimizing node embeddings (Equation 1.43).

$$\max_{\Phi} \sum_{u\in\Gamma_v} Pr(\Gamma_u|\Phi(v)) \tag{1.43}$$

Deepwalk [PARS14] is a popular random walk-based graph embedding approach. Deepwalk uses the simple random walk algorithm to generate a sequence of nodes where the walker walks m steps from the source node. The approach then computes the co-occurrence probability of each neighbour node in the sequence as Equation 1.44.

$$Pr(\Gamma_v|\Phi(v)) = \prod_{u\in\Gamma_v} Pr(u|\Phi(v)) \tag{1.44}$$

Deepwalk models the conditional probability, $Pr(v_i|\Phi(v_j)$ as Equation 1.45.

$$Pr(u|\Phi(v)) = \frac{1}{1+e^{-\Phi(u)\Phi(v)}} \tag{1.45}$$

The optimization function (Equation 1.43) is used to optimize embeddings of nodes. Deepwalk shows high link prediction performance in various graphs. However, the uniform random walk affects the quality of node embeddings and subsequent link prediction performance.
The *Node2Vec* is an improvement of Deepwalk that introduces a guided random walk for node sequence sampling. The guided random walk in Node2Vec is a 2^{nd} order random walk that interpolates between BFS (Breadth First Search) and DFS (Depth First Search)-based sampling strategy where two parameters p and q are used to compute the transition probability, $\pi(v,t)$ during the walk. These parameters control how fast the walk explores and leaves the neighborhood of the starting node. Given, a walker walks to v from u, the transition probability to move to neighbours of v is computed based on the two parameters as Equation 1.46.

$$\pi(v,t) = \begin{cases} \frac{1}{p}, & if\ Length(P_{u,t}^{shortest}) = 0 \\ 1, & if\ Length(P_{u,t}^{shortest}) = 1 \\ \frac{1}{q}, & if\ Length(P_{u,t}^{shortest}) = 2 \end{cases} \tag{1.46}$$

Figure 1.7 illustrates the guided random walk process. The walker travelled from u to v and now decides to visit one of the neighbours of v based on transition probability values. As the link

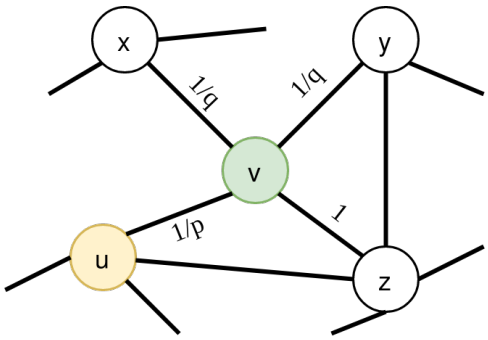


Figure 1.7: Illustration of a guided random walk in Node2Vec. The value associated to each edge is the transaction probability. The walker moved from u (yellow circle) to v (green circle) and now decides to move to node z following the highest transition probability edge.

(v,z) has the highest transition probability, the walker moves to the node z in this case. The co-occurrence probability of each neighbor, sampled by the walker, is estimated by a softmax function as Equation 1.47.

$$Pr(u|\Phi(v)) = \frac{exp(\Phi(u).\Phi(v))}{\sum_{n \in V} exp(\Phi(n).\Phi(v))} \tag{1.47}$$

Node2Vec use the same objective function (Equation 1.43) as Deepwalk to optimize node embedding.

Graph neural network (GNN)-based approaches

In recent years, the Neural Network (NN) has shown great success in a variety of machine learning applications, from image classification [RDGF16, RHGS15] to natural language processing [LPM15]. The objects in these applications are usually represented in regular Euclidean space, for example images are 2D grids and texts are 1D sequences. However, there are an increasing number of applications in which data is generated in non-Euclidean space and represented as graphs to include complex interactions between items [ZCH+20]. Due to the irregular

nature of graphs, NN faces two major challenges [WPC+20]. The first challenge is to define a good function for information propagation among nodes in graphs. The second challenge is that the assumption of independence of objects in classical machine learning methods are not true in graphs. The nodes in a graph are connected to their neighbours. Many recent studies in the literature are tackling the challenges to apply NN to graphs. These methods are called graph neural networks (GNNs) [SGT+08]. GNNs extend the convolution concept to graph data to learn embeddings of nodes and edges of a graph. The embeddings are then utilized in a neural network based model for the link prediction task. Since the development of the first GNN by Scarselli *et al.* in 2008 [SGT+08], researchers have developed a number of GNNs.

Weisfeiler-Lehman Neural Machine (WLNM) [ZC17] is a sub-graph sampling GNN to learn the structural features from a graph and use them in the link prediction task. As illustrated in Figure 1.8, WLNM is a three steps link prediction approach that starts with extracting a sub-graph for a target link that contains a predefined number of neighbour nodes around the target link. At second step, the nodes in the sub-graph are labelled with numbers using the well-known Weisfeiler-Lehman(WL) canonical labelling algorithm [WL68]. At the final step, the adjacency matrix of the sub-graph is computed where row and column indices correspond to node labels (1st row and column for node-1, 2nd row and column for node-2, and so on). The adjacency matrix is column-wise scanned to get the embedding of the sub-graph, and a neural network (NN)-based prediction model is fed with the embedding to predict the target link. WLNM is a

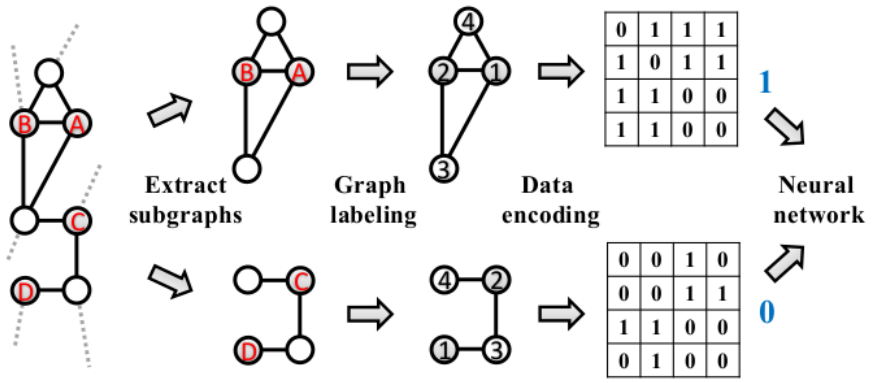


Figure 1.8: Illustration of WLNM with observed (A,B) and unobserved link(C,D) [ZC17]. The enclosing sub-graph of a link is extracted and the nodes in the sub-graphs are numbered (or labelled) using WL labelling algorithm. Based on the increasing numbering of nodes, the adjacency matrix of the sub-graph is passed to a NN-based prediction model.

simple GNN-based link prediction approach which is able to learn the link prediction heuristics from a graph. The downside of WLNM is that it truncates some neighbours to limit the sub-graph size to a user-defined size which may be informative for the prediction task.

Learning from Sub-graphs, Embeddings and Attributes (SEAL) [ZC18] is an improvement of WLNM that tackles variable neighbourhood size of a link. The approach integrates latent and explicit features of nodes with the structural information of graph to generate embedding of a link. The overall architecture of the approach is shown in Figure 1.9. Like WLNM, SEAL also consists of three major steps: (1) sub-graph extraction and node labelling, (2) node information matrix construction, and (3) training and evaluation of the Deep Graph Convolutional Neural Network (DGCNN) [ZCNC18] to handle neighbours of variable size. SEAL utilizes the available

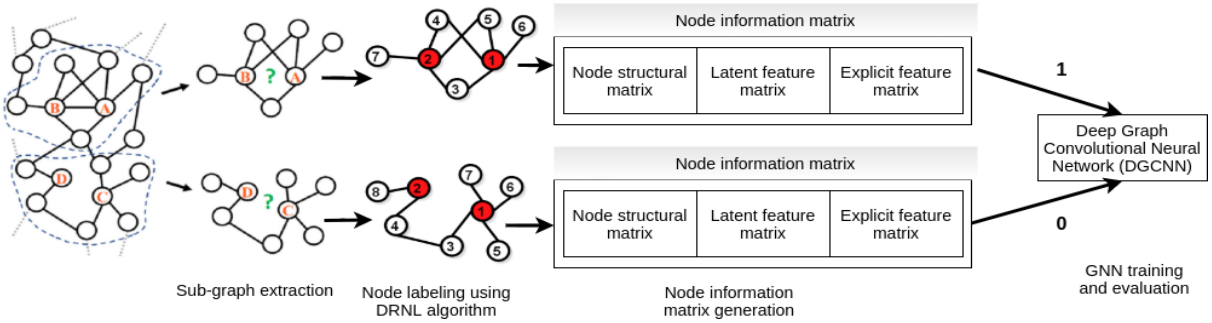


Figure 1.9: Architecture of the SEAL approach [ZC18]

information in the graph to improve the prediction performance.

The two discussed GNNs, WLNM and SEAL, are designed based on sub-graph sampling around a link. However, the consideration of distant neighbours may help to extract useful features and therefore improves the embeddings. The Graph Convolutional Network (GCN) [KW16a] is such a method where the embedding learning is defined by a multi-layer architecture. The propagation rule in GCN is a convolution operation. The first layer in GCN convolutes the immediate neighbourhood information whereas the last layer convolutes the most distant neighbourhood information during embedding learning. The basic Graph Convolutional Networks (GCN) [KW16a] method takes the adjacency matrix $A^{|V|\times|V|}$ of a graph, $G = (V, E)$, a feature matrix $F^{|V|\times D}$ as inputs, where D is the number of distinct external features of nodes. GCN defines an embedding learning layer, analogous to hidden layer in CNN, based on the feature and adjacency matrices as Equation 1.48.

$$H^i = f(H^{i-1}, A) \quad H^0 = F \tag{1.48}$$

Here, $H^i \in \mathbb{R}^{|V|\times D}$ is the embedding matrix of nodes at layer i, and f is the convolution function. Each row in H gives embedding of a node in the graph. The embeddings in the current layer are aggregated to learn embeddings in the next layer. If $W^{i-1} \in \mathbb{R}^{D\times D}$ is the trainable weight matrix at the layer $i-1$, the convolution function is defined as Equation 1.49.

$$f(H^{i-1}, A) = \sigma(AH^{i-1}W^{i-1}) \tag{1.49}$$

Here, σ is the ReLU non-linear activation function. Link prediction based on GCN method shows superior prediction performance in many real-world networks [KW16a, WV21]. After the introduction of GCN, many researchers aimed to improve it. *Graph Attention Networks (GAT)* [VCC+18] comes in front among them. GAT addresses a downside of GCN that the convolution function is defined based on immediate neighbours where all neighbors contribute equally which affects the prediction performance in many graphs. GAT overcomes this shortcoming by leveraging an attention mechanism for learning different weights (or coefficients) to different nodes in a neighborhood. GAT defines an attention layer to compute attention coefficient of each neighbour, y of a node, x. If $\mathbf{x}$ and $\mathbf{y}$ are feature vectors (embeddings) for nodes x and y respectively, the layer computes an attention score, c_{xy} for the neighbour y of node x as Equation 1.50.

$$c_{xy} = f_a(W^{i-1}\mathbf{x^{i-1}}, W^{i-1}\mathbf{y^{i-1}}) \tag{1.50}$$

Here, the attention function f_a is a LeakyReLU nonlinear function. The attention coefficient of

y is then computed using the following softmax function (Equation 1.51).

$$\alpha_{xy} = softmax(c_{xy}) = \frac{exp(c_{xy})}{\sum_{z \in \Gamma_x} exp(c_{xz})} \tag{1.51}$$

where Γ_x is the set of neighbours for node x. The attention mechanism is illustrated in Figure 1.10. The coefficient of the neighbour y indicates its importance to node x. The attention

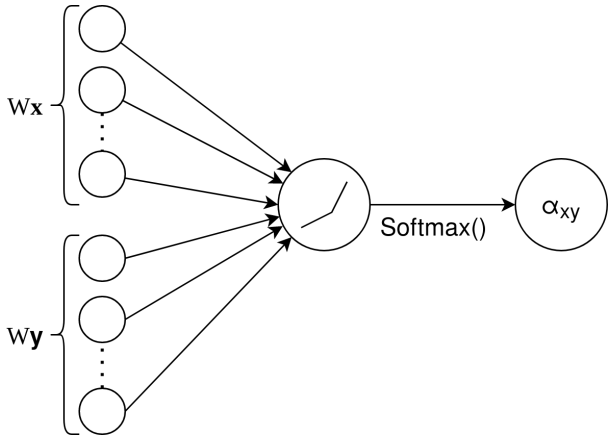


Figure 1.10: The attention mechanism.

coefficients of neighbours is used to update the embedding of x in the current layer, i as in Equation 1.52.

$$\mathbf{x}^i = \sigma\Big(\sum_{y \in \Gamma_x} \alpha_{xz} W^{i-1} \mathbf{y}^{i-1} \Big) \tag{1.52}$$

Link prediction based on GAT shows excellent prediction performance in many graphs. Recent years, researchers provided many improvements of the basic GCN architecture. AGCN [LWZH18], DualGCN [ZM18], FastGCN [CMX18], Cluster-GCN [CLS+19] are few examples. These improvements include various information in the convolution function such as node clustering, node position, neighbourhood selection.

Auto-encoder (AE)-based methods are another kind of embedding methods that work in an unsupervised manner [WPC+20]. An AE consists of two components: an encoder and a decoder [HZ94]. The encoder employs graph convolutional layers to learn node embeddings in a graph and the decoder reconstructs the graph structure from embeddings. Structural Deep Network Embedding (SDNE) [WCZ16] is considered a simple AE method to learn graph embedding that preserve first-order and second-order proximities in the embedding space. The encoder and the decoder are multi-layer perceptron (MLP) architectures. The loss function (Equation 1.53) for the encoder is defined based on the distance between end nodes embeddings of links to preserve first-order proximity.

$$L^{Enc} = \sum_{(x,y) \in E} A_{xy} \, ||\mathbf{x} - \mathbf{y}| \, |^2 \tag{1.53}$$

Here, $\mathbf{x}$ and $\mathbf{y}$ are embeddings of the nodes x and y respectively and A is the adjacency matrix. The decoder loss function is defined based on the distance between embeddings of a node embedding (x) and its reconstructed embedding (x') by the decoder as Equation 1.54.

$$L^{Dec} = \sum_{x \in V} \big| |(\mathbf{x}' - \mathbf{x}) \odot A_x \big| \, |^2 \tag{1.54}$$

Here, A_x is the adjacency vector for the node x. The *Variational Graph Auto-encoder (VGAE)* [KW16b] is another AE based graph embedding method that was designed based on the well-known variational auto-encoder (VAE) framework [KW14]. The main idea of VAE is that it represents an input to a multivariate Gaussian distribution rather than a point in low dimensional vector space and then the decoder randomly sample a low-dimensional embedding from the distribution to reconstruct the input from this embedding [KW14]. Like SDNE, VGAE consists of two components, an encoder and a decoder. The encoder of VGAE is a 2-layer GCN which takes the adjacency matrix (A), degree matrix(D), and a feature matrix(F) of graph as inputs. The first layer generates a low-dimensional feature matrix as in Equation 1.55.

$$X = GCN(F, A) = ReLU(A'F'W_0) \;\; with \; A' = D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \tag{1.55}$$

where W_0 is the weight matrix of first layer. The second layer generates the mean(μ) and standard deviation(σ) of the Gaussian distribution as in Equation 1.56.

$$\mu = GCN_\mu(X, A) = A'XW_\mu \quad log\sigma^2 = GCN_\sigma(X, A) = A'XW_\sigma \tag{1.56}$$

where W_μ, W_σ are weight matrices of second layer. The embeddings(Z) of nodes are computed using parameterization trick as in Equation 1.57.

$$Z = \mu + \sigma * \epsilon \;\; with \; \epsilon = \mathcal{N}(0, 1) \tag{1.57}$$

The decoder is a generative model which is defined as the inner product between latent variables to reconstructed adjacency matrix $\hat{A}$, which is defined as Equation 1.58.

$$\hat{A} = \sigma(Z^T Z) \tag{1.58}$$

where $\sigma(.)$ is the logistic sigmoid function. The weight matrices are learned by minimizing the reconstruction loss capturing the similarity of A and $\hat{A}$. To reduce the computational complexity of VGAE, *Local Variational Graph Auto-encoder(LGVAE)* [SHV19] replaces the GCN in VGAE with a straightforward linear model w.r.t. the normalized adjacency matrix. In contrast to GCN encoders in VGAE, LGVAE aggregates information from immediate neighbors of a node to learn its embeddings.

$$\mu = A'FW_\mu, \quad log\sigma = A'FW_\sigma \tag{1.59}$$

The node embeddings(Z) are computed using the same way (Equation 1.56) as in VGAE.

$$Z = \mu + \sigma * \epsilon \;\; with \; \epsilon = \mathcal{N}(0, 1) \tag{1.60}$$

The decoder remains the same as the decoder in VGAE (1.58). The weight matrices W_μ and W_σ are learned by minimizing the reconstruction loss as in VGAE. The node embeddings from these autoencoder are utilized in a prediction model for the link prediction task.

Overall, similarity-based approaches for link prediction in simple graphs are simple and explainable. They offer excellent link prediction performance in graphs with certain properties. Local similarity-based approaches are scalable to large graphs while global ones are not. The probability and statistical based approaches show better link prediction performance when graph structures are known, but take very high computational time and are not scalable to large graphs. Both similarity-based and probability and statistical approaches lack of the universal applicability. Supervised approaches show good prediction performance in graphs from various domains. They are also explainable. When features are defined based on local information, supervised approaches are scalable to large graphs. The recent embedding-based approaches automatically learn appropriate features in a graph for the link prediction task and outperform other approaches in most graphs. The major downside of these approaches is that they suffer from the 'black-box' problem, *i.e.*, they are unable to explain their predictions.

1.3 Link prediction in knowledge graphs

1.3.1 Problem formulation

Given a knowledge graph $KG = (V, R, Q)$, where V is the set of entities, R is the set of relation names, Q is the set of facts, the link prediction problem is defined as predicting the missing entity in a $(h, r, ?)/(?, r, t)$ triple, where ? represents the missing entity [RBF+21, LYL+22]. Figure 1.11 illustrates the link prediction task in the toy example KG (Figure 1d), where the unobserved triple (Vincent Cassel, Spouse of, Tina Kunakey) is predicted given (Vincent Cassel, Spouse of, ?) or (?, Spouse of, Tina Kunakey) as a query.

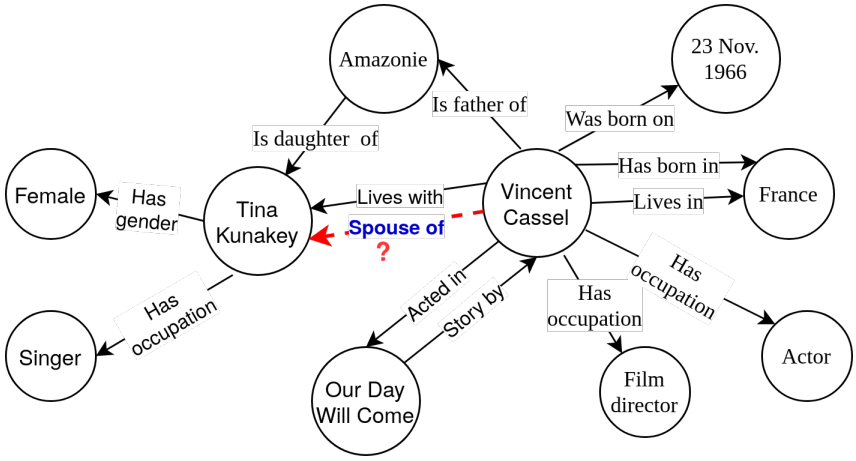


Figure 1.11: Link prediction in the toy KG. The unobserved but potential triple (Vincent Cassel, Spouse of, Tina Kunakey) is marked with a red dashed link. The link prediction task aims to compute plausibility score of this triple.

In this setting, the link prediction performance is defined with two widely used metrics: Hit@z, and mean reciprocal rank (MRR) [WQW21]. The metrics are defined based on the rank of the positive test triple. Hit@z is defined as the average number of times a positive test triple is among the z highest ranked triples; whereas MRR is the average reciprocal rank of the positive test triple [BUGD+13, YYH+15a]. The range of both scores is 0 to 1. The higher value of MRR demonstrates the better ranking of positive test triples and better ranking provides better prediction performance. Also, higher Hit@z score indicates better performance. To illustrate, consider the positive test triple $q = (h, r, t) \in \mathbb{D}$. A set of negative triples $q'_t = \{(h, r, t') \notin \mathbb{Q} | t' \in \mathbb{E}\}$ is generated by simple triple perturbation (replacing tail with other entities) [BUGD+13, JHX+15] confirming that no positive triple exists in q'_t. The triples in $q \cup q'_t$ are then ranked in decreasing order of their scores (computed by the triple scoring function of a prediction approach). The rank of the positive test triple q in $q \cup q'_t$ is defined as $rank^t_q$. Based on the rank of each positive test triple q, the performance metrics are defined in Equations 1.61 and 1.62.

$$Hit^t@z = \frac{1}{|\mathbb{D}|} \sum_{q \in \mathbb{D}} hit^t_q, \quad hit^t_q = \begin{cases} 1, & if\ rank^t_q \leq z \\ 0, & \text{otherwise} \end{cases} \tag{1.61}$$

$$MRR^t = \frac{1}{|\mathbb{D}|} \sum_{q \in \mathbb{D}} \frac{1}{rank^t_q} \tag{1.62}$$

The whole evaluation process is also repeated by corrupting the head entity of each positive triple

as well and $Hit^h@z$, MRR^h are computed. The final $Hit@z$, MRR metrics are the average of head and tails metrics *i.e.*, $Hit@z = (Hit^t@z + Hit^h@z)/2$ and $MRR = (MRR^t + MRR^h)/2$. As suggested by the most of the literature for link prediction in KGs, we consider $z \in \{1, 3, 10\}$.

For link prediction in KGs, researchers studied two types of approaches (Figure 1.12): rule-based and embedding-based approaches. State-of-the-art rule-based approaches are designed based on the symbolic representation of KGs while the embedding-based ones are designed based on the vector representations of the KGs.

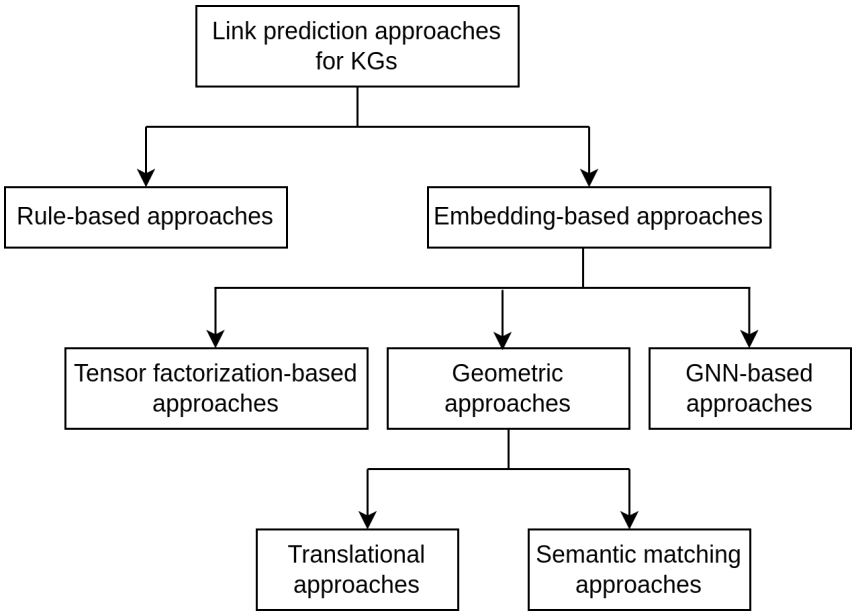


Figure 1.12: Taxonomy of link prediction approaches for knowledge graphs

1.3.2 The rule-based approaches

KGs contain facts describing real-world entities and their relations. Based on the RDF triple representation, it is possible to extract rules from KGs characterizing interesting patterns among relations. For example, we can extract the rule $IsfatherOf(h, x) \wedge IsdaughterOf(x, t) \rightarrow spouseOf(h, t)$ from toy example of KG (Figure 1.12) describing the fact that, very often, the father of person is spouse of another person if the first person is daughter of the second person. In this rule, $spouseOf$ is the target/head relation and $IsfatherOf$, $IsdaughterOf$ are the body relations. The rule-based approaches aim to extract rules based on the observed triples or facts in KGs and the extracted rules are used to infer new triples. Each rule in these methods can be exploited in prediction (or classification) mode using a deductive reasoning schema (*i.e.*, *If* the left (body) part of a rule is true *Then infer That* the right (head) part of the rule is true) [SKBH22]. For example, a rule-based approach can use the extracted rule to infer a new triple *(Vincent Cassel, spouseOf, Tina Kunakey)* or *spouseOf(Vincent Cassel, Tina Kunakey)* based on two facts *(Vincent Cassel, IsfatherOf, Amazonie)* and *(Amazonie, IsdaughterOf, Tina Kunakey)* in the KG (in Figure 1.12).

Rule mining has been widely studied in the context of inductive logic programming (ILP) [Qui90, Mug95, SDEW10]. ILP methods deduce logical rules from ground examples (positive and negative). These methods, however, are not suitable for KGs for two reasons. Firstly, they require negative examples which are not readily available in KGs and unobserved triples cannot serve as

negative examples because of the OWA. Secondly, the ILP methods are too slow and not scalable to large KGs. For example, Meilicke *et al.* [MCRS19] reported that state-of-the-art ILP methods were unable to generate any rule on YAGO2 in a couple of days. In this context, researchers study the transitive properties of relations in KGs to extract rules.

Based on the RDF representations of KGs, several rule mining methods exist in the literature which aim to mine rules implying target relations based on path information in the KGs. AMIE (Association Rule Mining under Incomplete Evidence) [GTHS13] is a popular rule mining approach to learn rules under OWA. As negative triples are not readily available in KGs, AMIE develops a strategy to imitate negative triples based on the assumption of partial completeness of a KG. For a certain relation, AMIE replaces the tail entity in a positive triple by a locally connected entity and checks if the generated negative triple does not exist in the current KG. This reduced search space for negative triples makes the approach faster and the use of negative triples makes it efficient. AMIE iteratively extends bodies of rules. At each iteration, it add new relations to bodies of rules, computes qualities of newly generated rules, and prunes low-quality rules. AMIE uses the *positive-only learning evaluation score* metric based on positive and sampled negative triples to measure the quality of a rule. **AMIE+** [GTHS15] is an improvement of AMIE where rule refinement operation is speed up by providing a faster rule quality score computation. It approximates the quality score of a rule based on its statistics, such as the functionalities of the atoms, or the size of the joins between two relations. Although AMIE+ is faster than AMIE, the rule pruning operation based on the approximate quality score leads to prune few good rules. Recently, Meilicke *et al.* [MCRS19] developed Anytime Bottom Up Rule Learning(AnyBURL) to learn rules in an incremental fashion. AnyBURL learns length-1 (one body relation) rules for a certain relation based on length-1 paths, computes their quality scores and retains only good rules with a quality score threshold. AnyBURL then increases the rule length by 1, samples paths, generates rules and filter the rules to keep only good rules. This process is repeated until a user-defined rule length is reached.

The aforementioned rule mining methods extract rules based on the symbolic representations of KGs. Recently, two methods, EmbedRule [YYH$^+$15a] and RLvLR (Rule Learning via Learning Representations) [OWW21] were developed to extract rules based on numerical representations (embeddings) of KGs. Overall, these methods learn embeddings of entities and relations in the KGs using KG embedding methods (will be discussed in Section 1.3.3) and use the learned embeddings and structure of the KGs for extracting rules. EmbedRule learns Horn rules based on the closed-path pattern where a path starts from the head and ends at the tail entity of a triple. EmbedRule models a relation as the composition of several relations and the composition operation is defined based on the multiplication or addition of two relation embeddings. It enumerates all possible pairs of relations for generating the bodies of rules for a relation. Like EmbedRule, RLvLR learns rules for a relation based on the closed-path pattern. It computes average embeddings of all heads and average embeddings of all tails of relations. RLvLR computes co-occurring relations based on the assumption that the average tail embedding should be very close to the average embedding of another relation. The co-occurring relations form the rule bodies for a given relation.

1.3.3 Embedding-based approaches

In recent years, researchers have studied approaches to represent a KG elements (entities and relations) in a low-dimensional vector space while preserving the inherent structure of the KG. These low-dimensional representations (embeddings) are used in the downstream link prediction task. These approaches are broadly categorized into three major categories: semantic matching,

geometric approaches, GNN-based approaches.

Tensor factorization-based approaches:

Extending the concept of adjacency matrix representation in simple graphs, a tensor is a 3D representation of a KG where each frontal slice is a adjacency matrix with respect to one relation. Figure 1.13 illustrates the tensor $A \in \mathbb{R}^{n \times n \times m}$ representation of a knowledge graph, $KG = (V, R, Q)$ for $n = |V|$, $m = |R|$. In the representation, A_k gives the adjacency matrix representation of the KG with respect to the relation R_k; $A_{ijk} = 1$ represents the triple (v_i, R_k, v_j). Like 2D matrix factorization, researchers study tensor factorization algorithms for

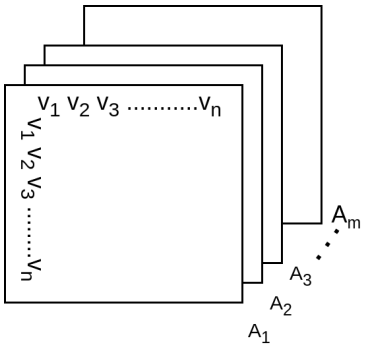


Figure 1.13: The 3D binary tensor representation $A \in \mathbb{R}^{n \times n \times m}$ of a KG

learning representations of entities and relations. RESCAL [NTK11] is considered as the first factorization-based approach for link prediction in KGs which was designed based on the well-established CANDECOMP/PARAFAC (CP) decomposition algorithm [HL94]. The approach employs a rank-d factorization of the tensor representation as Equation 1.63.

$$A_k = XW_kX^T \quad for \quad k = 1, 2, ..., m \tag{1.63}$$

Here, $X \in \mathbb{R}^{n \times d}$ is the embedding matrix where each row gives the embedding of an entity, $W_k \in \mathbb{R}^{d \times d}$ is the interaction matrix modelling interaction between pairs of embeddings with respect to the relation R_k. RESCAL defines an embedding optimization task by minimizing the loss between the original tensor and reconstructed tensor as Equation 1.64.

$$\min_{X, W_k} \frac{1}{2}\Big(||A_k - XW_kX^T||^2\Big) + g(X, W_k) \tag{1.64}$$

Here, $g(X, W_k)$ is a regularization factor to prevent over-fitting of the approach and defined as Equation 1.65.

$$g(X, W_k) = \frac{1}{2}\lambda\Big(||X||^2 + ||W_k||^2\Big) \tag{1.65}$$

The plausibility score of an unobserved triple $(v_i, r_k, v_j) \notin Q$ is computed as $X_i^T W_k X_j$. The number of parameters in RESCAL is very high and rises quadratically with increasing embedding dimension size and the number of relations. Balazevic *et al.* developed TuckER [BAH19] approach based on the Tucker decomposition [Tuc66]. The Tucker algorithm decomposes the original tensor, $A \in \mathbb{R}^{n \times n \times m}$ into three matrices $X \in \mathbb{R}^{n \times d}$,$W \in \mathbb{R}^{m \times d}$ and a small tensor $Z \in \mathbb{R}^{d \times d \times d}$ as Equation 1.66.

$$A = ZXWX^T \tag{1.66}$$

Here, X is the entity embedding matrix, W is the relation embedding matrix, Z is a core tensor modelling interactions between the different components. The plausibility score of an unobserved triple $(v_i, r_k, v_j) \notin Q$ is computed as $ZX_i^T W_k X_j$.

GNN-based approaches:

The success of GNN for link prediction in simple graphs motivated researchers to study GNN in knowledge graphs. GNN-based approaches extend the convolution operation to include relation information during embedding learning. **KG2Vec** [SRM$^+$18] is the skip-gram model based KG embedding method that considers one triple $T = (v_1, v_2, v_3)$ as a sequence of three KG elements. The method maximizes the co-occurrence probability of elements in the sequence as Equation 1.67.

$$\max \frac{1}{3} \sum_{v_i \in T} \sum_{v_j \in T} logp(v_i|v_j) \tag{1.67}$$

The probability is defined as

$$p(v_i|v_j) = \frac{exp(\mathbf{v_i v_j})}{\sum_{v_k \in E \cup R} exp(\mathbf{v_k v_j})} \tag{1.68}$$

Here, E represents the set of entities, R represents the set of relations, the bold notations $\mathbf{v_i}, \mathbf{v_j}, \mathbf{v_k}$ represent embeddings of the elements v_i, v_j, v_k respectively. A neural network (NN)-based prediction model is trained with the positive and negative triple embeddings and the probability of each unobserved triple is computed using the trained prediction model.
Schlichtkrull *et al.* (2018) extended the GCN architecture from simple graphs to KGs as **Relational GCN (RGCN)** [SKB$^+$18]. RGCN consists of two components: a encoder that learns embeddings of entities and a decoder that computes plausibility scores of unobserved triples based on learned embeddings. The encoder defines a convolution layer based on relation-wise propagation of neighbours of a node, h as Equation 1.69.

$$\mathbf{h}^l = ReLU\Big(\sum_{r \in \mathbf{R}} \sum_{t \in N_h^r} \frac{1}{|N_h^r|} W_r^{l-1} \mathbf{t}^{l-1} + W_0^{l-1} \mathbf{h}^{l-1} \Big) \tag{1.69}$$

Here, $ReLU(.)$ is a non-linear activation function for the current layer l, bold notations $\mathbf{h}, \mathbf{t} \in \mathbb{R}^d$ represent embeddings of the nodes h, t, N_h^r is the set of neighbours for the node h which are connected by the relation r, $W_r \in \mathbb{R}^{d \times d}$ is a diagonal learnable weight matrix representing embedding of the relation r. RGCN adds a self loop to each node to include its embedding from the previous $(l-1)$ layer in defining the convolution operation in the current layer. Figure 1.14 illustrates the convolution layer. Multiple convolution layers are stacked to design a multi-layer RGCN model. For the decoder part, RGCN uses the scoring function concept of RESCAL. The scoring function for a triple (h, r, t) is defined as Equation 1.70.

$$f(h, r, t) = \sigma(\mathbf{h}^T W_r \mathbf{t}) \tag{1.70}$$

RGCN corrupts the positive triples to generate negative triples and uses a cross-entropy loss function (Equation 1.71) to optimize embeddings of the entities and relations of a KG.

$$L = \frac{1}{\omega} \sum_{(h,r,t,y) \in \omega} ylog\Big(f(h, r, t)\Big) + \Big(1 - ylog(f(h, r, t))\Big) \tag{1.71}$$

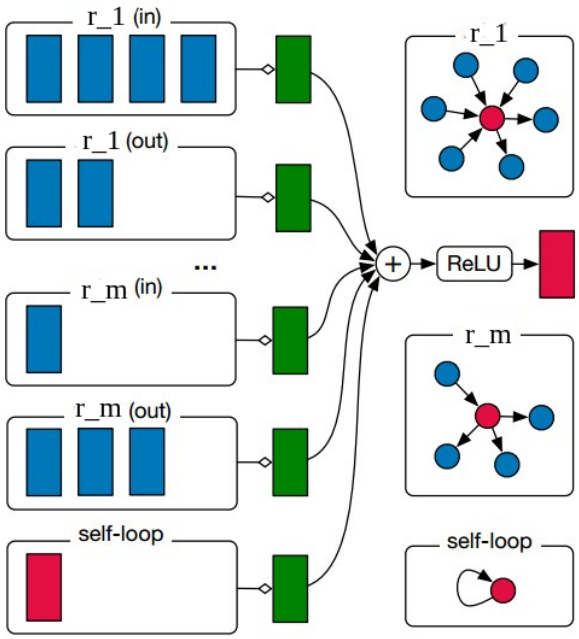


Figure 1.14: Illustration of a single RGCN layer. The current node/entity and its embedding are marked in red color, (in) and (out) denote the in-edges and out-edges respectively.

Here, ω is the set of all triples including positive (label $y = 1$) and negative (label $y = 0$) triples. Relational GAT (RGAT) [BSCH18] is the extension of the GAT architecture that learns entities and relations embeddings of KGs. Like RGCN, RGAT also consists of an encoder and a decoder. The encoder computes an attention coefficient for each neighbour of a node and utilizes this coefficient in the convolution operation. Given $\mathbf{h}$ and $\mathbf{t_i}$, the embeddings of the entities h and t_i respectively, W_r is the representation matrix for the relation r, RGAT computes an attention score c_{hrt_i} as Equation 1.72.

$$c_{hrt_i} = f_a(\mathbf{h}^T W r \odot \mathbf{t_i}^T W r) \tag{1.72}$$

Here, f_a is an attention function, $\odot$ denotes a multiplication between two vectors. RGAT uses the non-linear LeakyReLu as the attention function. The attention scores for all neighbours are normalized by a softmax function to compute attention coefficients of all neighbours of a node as Equation 1.73.

$$\alpha_{hrt_i} = \frac{c_{hrt_i}}{\sum_{t_j \in \Gamma_i^r} c_{hrt_j}} \tag{1.73}$$

The convolution operation is defined based on the learned attention coefficient and neighbours embeddings as Equation 1.74

$$\mathbf{h}^l = ReLU\Big(\sum_{r \in \mathbf{R}} \sum_{t \in \Gamma_h^r} \alpha_{hrt} \mathbf{t}\Big) \tag{1.74}$$

Here, Γ_h^r is the neighbour set of the node h which are connected by the relation r. The encoder in RGAT uses the same loss function as RGCN (Equation 1.70) to optimize embeddings. For link prediction task, the decoder uses the same triple scoring function (Equation 1.71) as RGCN to define the decoder.

There are few other GNN-based approaches for link prediction in KGS. RGHAT [ZZZ+20] is another GAT-based architecture which introduces entity and relation attentions in convolution operations. Liu *et al.* developed Relation Aware GAT (RAGAT) [LTCL21] that treats

different relations differently and defines different convolution functions for different relations. Rather than learning embedding in a single vector space, Mixed-Curvature Multi-Relational GNN (M^2GNN) [WWNdS$^+$21] learns embeddings in a Riemannian product space to address heterogeneity structures in a KG. The Riemannian product space is a manifold space that mixes the Euclidean, hyperbolic, and spherical spaces.

Geometric approaches:

The geometric approaches for learning KG embeddings work based on a basic assumption that the observed triples in a KG should have high plausibility scores and the unobserved triples should have low plausibility scores in the latent space [WMWG17]. Each approach defines a triple scoring function, $f(\mathbf{h}, \mathbf{r}, \mathbf{t})$ based on the embeddings of entities and relation of the triple. The embeddings are denoted with bold letters. The architecture of a geometric KG embedding model is given in Figure 1.15 which starts with initializing the embeddings of entities and relations randomly from uniform/Gaussian distributions [WMWG17]. For the training of the model, a batch of positive training triples S_m is fetched and a negative sampling method is then used to generate a batch of negative triples, S'_m. The batches of positive and negative triples are then used to compute loss. The loss is propagated to an optimizer to update the embeddings. Geometric approaches allow to define two types of loss functions: squared and pairwise ranking

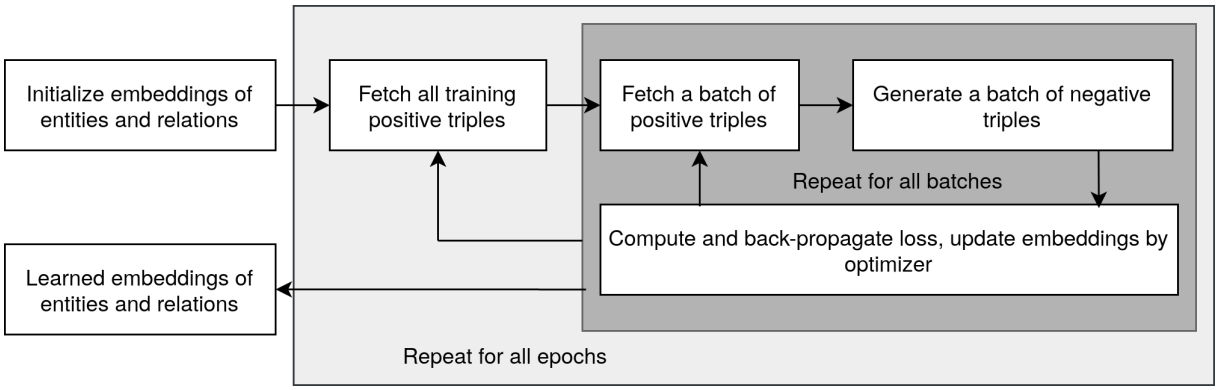


Figure 1.15: Architecture of a geometric KG embedding model

losses [NMTG15, WMWG17, RBF$^+$21]. The first squared loss function is defined under CWA (discussed in Section 1.3.1) as Equation 1.75.

$$L = \sum_{(h,r,t) \in S_m \cup S'_m} \Big(y_{hrt} - f(h,r,t) \Big)^2 \tag{1.75}$$

Here, $y = 1/0$ for a positive/negative triple. On the other hand, the pairwise ranking loss is defined under OWA (discussed in Section 1.3.1). The loss is computed as the score difference between a pair of positive and negative triple with a margin λ as Equation 1.76.

$$L = \sum_{(h,r,t) \in S_m} \sum_{(h',r',t') \in S'_m} \Big[\lambda - f(h,r,t) + f(h',r,t') \Big]_+ \tag{1.76}$$

Here, $[.]_+ = max(0, .)$ is the hinge function. The training objective is to optimize the embeddings of entities and relations for minimizing the total loss as designed in Equation 1.77.

$$\min_{\Theta} \sum_{\forall_{(h,r,t) \in S_m, (h',r,t') \in S'_m}} L(f(h,r,t), f(h',r,t')) + \lambda reg(\Theta) \tag{1.77}$$

The embedding updating process is repeated for all batches of positive triples, and the whole training process is repeated for T times (or epochs).

The state-of-the-art geometric approaches for link prediction in KGs are categorized into two categories: translational and semantic matching approaches. The **translational** approaches represent a relation in a triple as the translation from the head entity to tail entity in the embedding space. They measure the plausibility of a triple as the distance between the two entities, usually after a translation carried out by the relation [WMWG17]. These approaches are also called latent distance approaches. *TransE* [BUGD+13] is considered to be the first translational approach for link prediction in KGs. Given $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^{\mathbf{d}}$ are the embeddings of h,r,t of a triple (h, r, t), TransE defines the scoring function as Equation 1.78.

$$f(h, r, t) = -||\mathbf{h} + \mathbf{r} - \mathbf{t}|| \tag{1.78}$$

Figure 1.16a illustrates the TransE approach. TransE is simple but efficient in KG completion task. However, it has limitations to dealing with one-to-many and many-to-one relations [WZFC14]. Several improvements of TransE exist including TransH [WZFC14], TransR [LLS+15], TransD [JHX+15]. *TransH* improves TransE by introducing relation-specific hyperplane to define different roles for different relations of an entity. TransH defines $w_r \in \mathbb{R}^d$ as the normal vector on a relation, r specific hyperplane and projects h and t on this hyperplane as follows.

$$\mathbf{h_r} = \mathbf{h} - w_r^T \mathbf{h} w_r$$
$$\mathbf{t_r} = \mathbf{t} - w_r^T \mathbf{t} w_r$$

As illustrated in Figure 1.16b, the projected embeddings of entities are then used to define the scoring function as Equation 1.79.

$$f(h, r, t) = -||\mathbf{h_r} + \mathbf{r} - \mathbf{t_r}|| \tag{1.79}$$

TransH shows better prediction performance when dealing with one-to-many relations. However, TransH still assumes that entities and relations are in the same vector space. *TransR* introduces relation vector space and projects the entity embedding into the relation specific vector space based on the relation specific matrix $M_r \in \mathbb{R}^{k \times d}$ to project the entity embedding into the relation vector space as follows.

$$\mathbf{h_r} = M_r \mathbf{h}$$
$$\mathbf{t_r} = M_r \mathbf{t}$$

The projected embeddings of entities are then used to define the scoring function as Equation 1.80 (illustrated in Figure 1.16d).

$$f(h, r, t) = -||\mathbf{h_r} + \mathbf{r} - \mathbf{t_r}|| \tag{1.80}$$

TransR improves the prediction performance comparing to previous state-of-the-art approaches in many KG datasets. However, it is not an efficient mechanism for defining projection matrices only based on relation names as different entities may interact with the same relation [WMWG17]. *TransD* improves TransR by decomposing the projection matrix, M_r into a product of two vectors $w_h \in \mathbb{R}^d$ and $w_t \in \mathbb{R}^d$ which are used for projecting head and tail entities in the relation space (Figure 1.16a). For each triple (h,r,t), the projection matrices M_r^h and M_r^t of head and tails entities are defined based on the mapping vectors w_h, w_t and w_r as following.

$$M_r^h = w_r w_h^T + I$$
$$M_r^t = w_r w_t^T + I$$

Here, I is an identity matrix for relations. The projected embeddings of head and tail entities on relation space are computed based on the these two projection matrices in the same way as in TransR.

$$\mathbf{h_r} = M_r^h \mathbf{h}$$
$$\mathbf{t_r} = M_r^t \mathbf{t}$$

The projected embeddings are utilized to compute the score of a triple using the same scoring function as in TransR (Equation 1.80). Xiao *et al.* designed *TransG* [XHZ16] to model the multiple semantics (e.g. (Hand, partOf, Human body), (Handle, partOf, Bicycle), (Word, partOf, Sentence)) of a relation by leveraging the Gaussian mixture model. The approach produces multiple translation components for a relation. TransG uses the Chinese Restaurant Process (CRP) to define the semantic components of a relation by defining a weight $w_{r,i}$ and translation vector $\mathbf{r_i}$ for the i^{th} semantic of the relation r. TransG defines its scoring function based on h and t embeddings and S_r semantics of the relation r in a (h,r,t) triple as Equation 1.81.

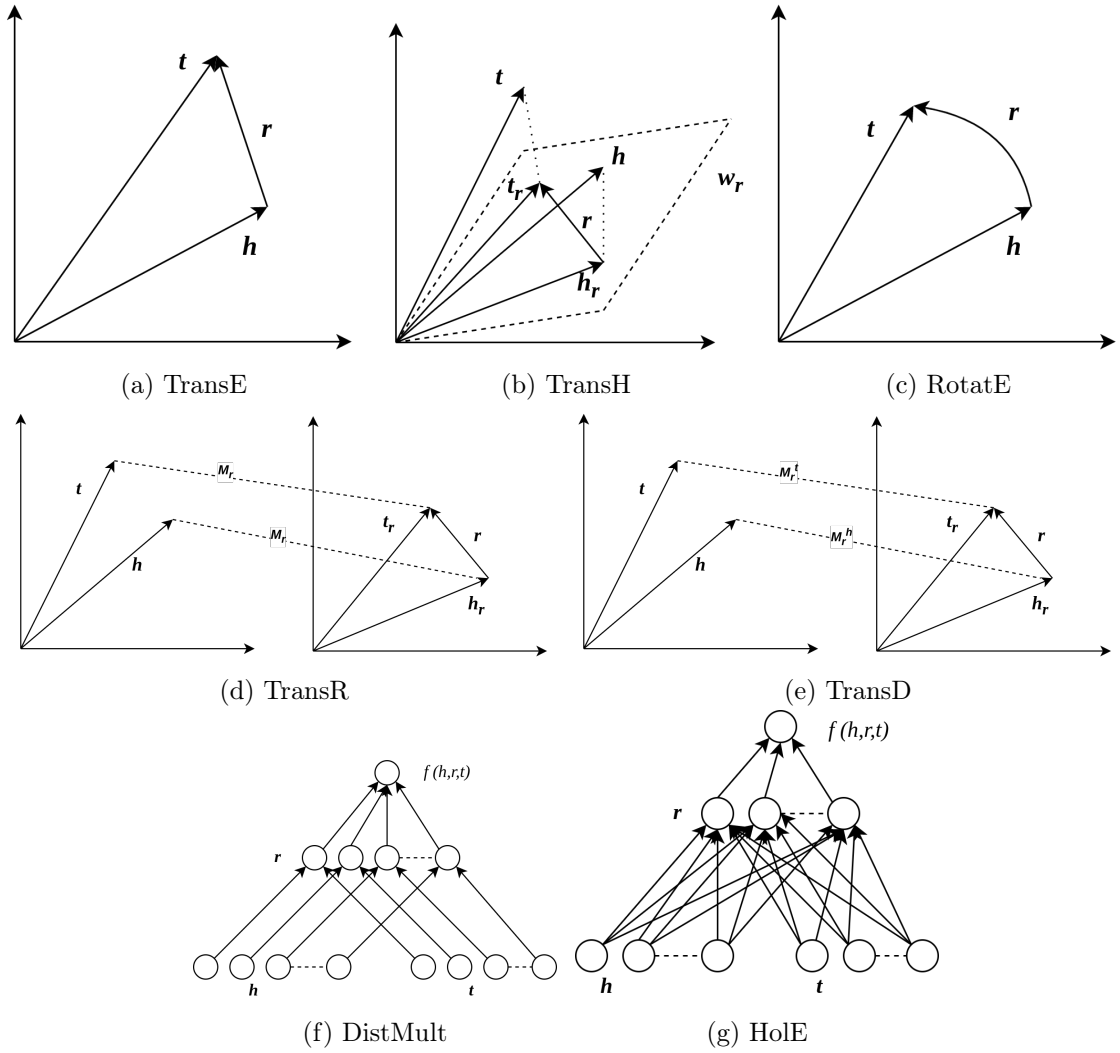
$$f(h,r,t) = \sum_{i=1}^{S_r} w_{r,i} e^{\frac{-||\mathbf{h}+\mathbf{r_i}-\mathbf{t}||_2^2}{k}} \tag{1.81}$$

Here, k is a normalizing constant that is computed as $k = \sigma_h^2 + \sigma_t^2$ for the variances σ_h and σ_t of head and tail entities respectively. TransG offers good prediction performance in many KG datasets. However, the problem of modelling one-to-many or many-to-one relations still persists in TransG.
RotatE [SDNT18a] is another translational approach that models the relation in a triple as the rotation from the head to the tail in the complex space as illustrated in Figure 1.16c. The embedding of a KG element consists of a real and an imaginary components, *i.e.*, $\mathbf{o} = Re(\mathbf{o}) + Im(\mathbf{o})$. RotatE defines the scoring function as Equation 1.82.

$$f(h,r,t) = -||\mathbf{h} \odot \mathbf{r} - \mathbf{t}|| \tag{1.82}$$

Here, $\odot$ is the Hadamard product of two embeddings. RotatE is able to handle symmetric as well as anti-symmetric relations. There exist a few other translational approaches in the literature to improve link prediction performance in KGs such as TorusE [EI18] to learn embeddings in Torus space, TransA [JWL+16] to learn the margin in the pairwise loss function, TransMS [YTZ+19] to use the concept of multiple semantics of a relation, StructurE [ZWYX22] to define different scoring functions for one-to-one, one-to-many, many-to-one and many-to-many relations.

Apart from translational approaches, researchers studied **semantic matching** approaches for link prediction in KGs. These approaches define the triple scoring function based on the concept of matching semantics of entities and relations in the embedding space [WMWG17]. Generally, the plausibility of a triple in a semantic matching approach is computed based on the product of head and tail embedding, and their relation specific interaction matrix [NMTG15, WMWG17]. These approaches are also called bilinear approaches as they use product of embeddings in their scoring function. These approaches are close to tensor factorization-based link prediction approaches. However, we distinguish this category from the factorization-based category for two reasons. Firstly, the semantic matching approaches are more about optimisation process rather than the tensor factorization. Secondly, tensor factorization-based approaches require triple labels (A_k in Equation 1.64) while semantic matching approaches do not require triple labels as we use pairwise ranking loss function in the embedding optimization process. *DistMult* [YYH+15a]



(a) TransE (b) TransH (c) RotatE

(d) TransR (e) TransD

(f) DistMult (g) HolE

Figure 1.16: Illustrations of a few geometric KG embedding methods

is considered as the primitive semantic matching approach. This approach simplifies the relational interaction matrices to diagonal matrices to significantly reduce the number of learnable parameters. DistMult defines the scoring function as the following Equation 1.83 (illustrated in Figure 1.16f).

$$f(h,r,t) = \mathbf{h}^T diag(r)\mathbf{t} = \sum_{i=0}^{d-1} [\mathbf{r}]_i.[\mathbf{h}]_i.[\mathbf{t}]_i \tag{1.83}$$

As the scoring function is a bilinear function, DistMult offer good prediction performance over symmetric relations in a KG. However, the approach can not model anti-symmetric and inverse relations. *Holographic Embeddings (HolE)*[NRP16] improves DistMult by employing the circular correlation [Pla95] between the head and tail of a triple to produce compositional embedding of the entity pairs. As illustrated in Figure 1.16g, the scoring function is then defined as the product of the compositional and relation embeddings as Equation 1.84.

$$f(h,r,t) = \mathbf{r^T}[\mathbf{h} \star \mathbf{t}] = \sum_{k}^{d-1} [h \star t]_k \tag{1.84}$$

Here, $[h \star t]_k$ represents the k^{th} component of the compositional embedding and it is computed as following.

$$[h \star t]_k = \sum_{i=0}^{d-1} [\mathbf{h}]_k . [\mathbf{t}]_{(k+i) \ mod \ d} \tag{1.85}$$

Here, (.) denotes the dot product between two elements. As the circular correlation operation is not symmetric, *i.e.*, $(h \star t) \neq (t \star h)$, HolE is able to model anti-symmetric relations. Instead of learning embeddings in the Euclidean space, *ComplEx* [TWR+16] learns embeddings in the Complex space to better model symmetric as well as anti-symmetric relations. The embedding of a KG element consists of a real and an imaginary component, *i.e.*, $\mathbf{o} = Re(\mathbf{o}) + Im(\mathbf{o})$. CompleX uses the Hermitian dot product of embeddings instead of the traditional dot product based on the conjugate-transpose of one of two entity embeddings. This type of dot product operation enables Complex to capture anti-symmetric relations. The scoring function is defined as follows.

$$\begin{aligned} f(h,r,t) &= Re(h^T diag(r) \bar{t}) \\ &=< Re(\mathbf{r}), Re(\mathbf{h}), Re(\mathbf{t}) > + < Re(\mathbf{r}), Im(\mathbf{h}), Im(\mathbf{t}) > + \\ &\quad < Im(\mathbf{r}), Re(\mathbf{h}), Im(\mathbf{t}) > - < Im(\mathbf{r}), Im(\mathbf{h}), Re(\mathbf{t}) > \end{aligned}$$

where $\bar{t}$ is the conjugate of t and $< ., ., >$ denotes the Hamilton dot product. As the scoring function contains both symmetric (the real product) and anti-symmetric (the imaginary product) components, ComplEx is able to model both symmetric and anti-symmetric relations. Quaternion Embedding (QuatE) [ZTYL19], *Quaternion Relational Embedding (QuatRE)* [NVNP22] are two extensions of ComplEX that use one real and four imaginary components to define embedding of each entity and relation.

To summarize, rule-based approaches for link prediction in KGs are fully explainable while they have two major drawbacks. Firstly, they mostly work under the closed-world assumption (CWA). However, this assumption is not entirely true as real-world KGs are generally incomplete. Secondly, these approaches take excessive computational time which make them infeasible to study in large scale KGs. GNN-based approaches are faster than rule-based approaches, but they also work under CWA. Tensor-factorization based approaches offer high prediction performance in many KG datasets, but they need high-computation time and memory to store and factorize tensors and are scalable to large scale KGs. The recent geometric approaches are simple, but provide high link prediction performance requiring low computational time and memory. They are also able to work under OWA but lack explainability.

1.3.4 Sampling of negative triples

Knowledge graph embedding methods require positive and negative triples for learning entity and relation embeddings. However, KGs only provide positive triples. Within this context, negative triple generation or sampling is an important aspect for knowledge graph embedding methods. There exist four main negative triple sampling methods in the literature including random-uniform [BUGD+13], self-adversarial [SDNT18b], KBGAN [CW18], and NSCaching [ZYSC19].

Random-uniform sampling

Random-uniform sampling was firstly used by Bordes et al. [BUGD+13] for sampling negative triples. The method generates negative triple(s) for a positive triple by perturbing the positive triple. For a given positive triple $q = (h, r, t) \in Q$, the method generates two sets of candidate negative triples $q'_h = \{(h', r, t) \notin Q | h' \in V\}$ and $q'_t = \{(h, r, t') \notin Q | t' \in V\}$ by replacing h and

t with other entities in the entity set (V). A required number of negative triples are uniformly sampled from the union set $q'_h \cup q'_t$ of the candidate sets, where the sampling probability of each negative triple is equal. Random-uniform sampling method is very simple and fast. However, it has a serious issue. There is a high probability of sampling very easy and less informative negative triple which leads to the "zero-loss" problem [WLP18]. To illustrate the problem, consider the scores of a positive and its corresponding easy negative triples are 5.0 and 1.0 respectively. With a margin of 3.0, the loss for this pair of triple is $[3.0 - 5.0 + 1.0]_+ = 0.0$. This "zero-loss" leads to "zero/vanishing gradient" problem because the easy negative triples make no contribution to improving the embeddings.

Self-adversarial sampling

Sun *et al.* (2019) developed a self-adversarial method [SDNT18a] to sample high-quality negative triples for KGs. Like random-uniform sampling, self-adversarial sampling generates a set of candidate negative triples, $q' = \{(h', r, t')\} = q'_h \cup q'_t$, by corrupting h and t of a positive triple (h,r,t). Rather than considering uniform sampling probability, the sampling method uses a softmax function (Equation 1.86) to compute sampling probability of each negative triple in the candidate set.

$$p(h', r', t') = \frac{exp(f(h', r, t'))}{\sum_i exp(f(h'_i, r, t'_i))} \tag{1.86}$$

A negative sample with high score gets high sampling probability. The sampling method samples the negative triple with the highest sampling probability. This negative is also called a hard negative due to the hardness of identifying it as a negative based on its score. The sampling method is able to minimize the "vanishing gradient" by sampling hard negatives.

GAN-based sampling

Generative adversarial networks (GAN) [GPAM$^+$14] is an approach to generative modelling that has been successfully applied to computer vision for generating very realistic photos, videos. The generative property of GAN motivated researchers to study it in KGs to generate negative triples. **KBGAN** [CW18] is considered to be the first GAN-based sampling method for KGs. KBGAN consists of two components: a generator and a discriminator. The generator is trained to generate high-quality negative triples where the discriminator is trained based on a pair of positive and generated negative triples to learn the embeddings of entities and relations. Unlike other negative sampling methods, KBGAN generates negative triples while learning embeddings. The generator first uses the “Bern” sampling strategy [WZFC14] when replacing the head/tail in a positive triple for generating a candidate set of negative triples. The generator utilizes a semantic matching scoring function (e.g. Equation 1.83) to compute scores of all candidates. Based on the scores of all candidates, the generator uses the same softmax probabilistic function (Equation 1.86) as in self-adversarial method to compute the sampling probability distribution over all candidates. The selector in generator then samples the negative with the highest sampling probability. The discriminator is a translational model (e.g. TransE [BUGD$^+$13], TransH [WZFC14]). The discriminator receives a positive and its corresponding negative triples and computes their scores. Based on the scores of the positive and negative triples, it computes the pairwise loss and updates embeddings of entities and relations. The score of the sampled negative triple is passed as the reward to the generator. The generator is trained to maximize the reward using the simple one-step policy gradient-based reinforcement learning strategy called REINFORCE [Wil92]. The loss in the discriminator is forwarded as the reward for the generator. There are other GAN-

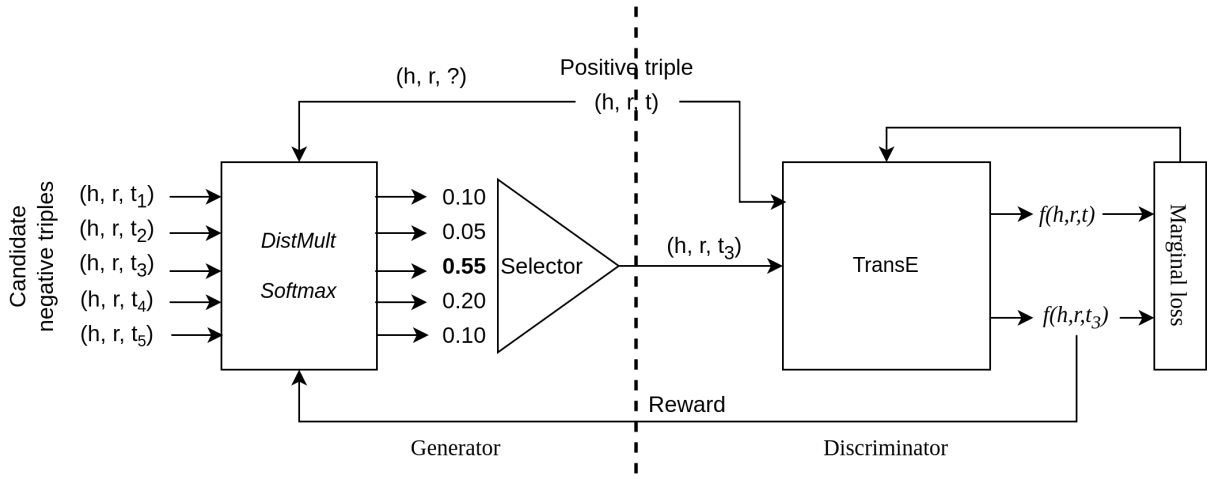


Figure 1.17: Illustration of the KBGAN training. The generator generates a set of candidate negative triples for a positive, computes a probability distribution over candidate negative triples and samples the negative triple with the highest sampling probability. The discriminator collects the positive and the sampled negative, computes the pairwise loss and update embeddings. The score of the negative triple is passed as a reward to the generator to train the generator.

based negative triple sampling methods for KGs, such as *IGAN* [WLP18], *KSGAN* [HLH19], *NKSGAN* [LHWH20]. The generator computes score of each candidate negative triple by applying a non-linear ReLU function to the concatenated embeddings of its entities and relation. The generator then uses a softmax function like KBGAN to compute sampling probability of each candidate negative triple. The discriminator, reward definition and maximization steps are same as KBGAN. The *KSGAN* [HLH19] method uses a knowledge selector in the generator part to build a set of negative triples with relatively high probabilities from the candidate set and identifies the negative triple which has the closest distance to its corresponding positive triple. *NKSGAN* [LHWH20] leverages the attention mechanism to neighbours of the corrupted entities to further improve the quality of negative triples. Empirical results show that GAN-based sampling methods are able to generate high-quality (hard) negative triples to reduce "vanishing gradient problem".

Caching-based sampling

Zhang *et al.* [ZYSC19] claim that the number of high-quality negative triples for a positive triple is low in a KG. NScaching stores these high-quality negative triples in a special cache and samples required negative triples from the cache during embedding method training. The basic idea is to generate a set of candidate negative triples for a positive triple, compare them with the set of negative triples already in the cache memory to build a set of high-quality negatives and keep them in the cache memory. The quality of negatives are quantified by the same softmax function (Equation 1.86) as in self-adversarial method. The required number of negative triples are sampled from the cache memory.

To summarize, the random-uniform sampling is the simplest and fastest one to generate negative triples for KGs. However, this method has the serious "vanishing gradient" issue. Self-adversarial, GAN-based and NSCaching methods bring strategies to reduce this issue. Self-adversarial computes sampling probability of candidate negative triples to sample candidates with high probability values. This method is also simple, but it does not involve the use of

already known high-quality negative triples. GAN-based methods uses GAN architecture to generate high-quality negatives. The downsides of these methods are that they suffer from the instability and degeneracy problems to prediction performance which is a common problem in REINFORCE [ZYSC19]. In addition, the high number of trainable parameters brings extra costs on training to fit the full distribution of negative triples. NSCaching is a distilled version of GAN-based methods that stores negative triples in a cache memory to avoid the training time for additional parameters in GAN. The downside of this method is that it requires high amount of memory and extra time for updating the memory.

1.4 Applications of link prediction

The link prediction task can be applied to numerous applications, ranging from simple social networks to complex biological knowledge graphs. Friend recommendation in social networks [Xie10, KMR18, Nar20], product recommendation in e-commerce [KBV09, ZDC+21, XRK+20, ZYY+21], knowledge graph completion, question answering [BCW14, BWU14, HZLL19, WWL+19], drug repurposing [ZCJ+20, ASGR+21, ZHS+21, YLY+21] are just few examples.

Friend recommendation in social networks

Friend recommendation is a common problem in social networks (e.g. Facebook, Twitter, Instagram, Yelp) where the nodes represent persons and the links represent friendships. The link prediction problem is considered as a friend recommendation problem where the task is to compute the link probability score between two disconnected persons. Many friend recommendation systems are developed based on simple similarity-based link prediction approaches. Xie [Xie10] used the Jaccard Index (JA)-based approach to develop the friend recommendation system. Kumar *et al.* [KMR18] and Cai *et al* [CBK+10] used the common neighbour (CN)-based link prediction approach in their recommendation systems. Huang *et al.* [HTL+15] adopt the CCLP similarity-based approach for the friend recommendation task. Based on the matrix factorization-based approaches, He *et al.* [HLF+15] and Xu [Xu18] presented their friend recommendation systems. He *et al.* used users attribute matrix while Xu used the adjacency matrix. Recently, Narasanna [Nar20], Yu *et al.* [YQZG22] and Zhao *et al.* [ZQW+19], Wei *et al.* [WLS+22] developed friend recommendation systems based on GNN architectures. Narasanna and Yu *et al.* used the basic GCN while Zhao *et al.* used the VGAE method to learn embeddings of users. They define a binary operator to compute the friendship probability for a pair of disconnected persons based on their embeddings. Wei *et al.* extract two neighbourhood sub-graphs of end users of an unobserved friendship, learns embedding of each sub-graph and defines the friend prediction based on these two embeddings [WLS+22]. Based on supervised link prediction approaches, Krakan *et al.* [KHS18] and Chen *et al.* [CLYY16] presented their recommendation systems where they extracted a set of features based on nodes structural and attribute information and then train a supervised learning algorithm, such as SVM, Naive Bayes, kNN, for the link prediction task.

Drug repurposing

Drug repurposing, also known as drug repositioning, is a strategy that seeks to identify new indications and targets for approved or investigational drugs that are beyond the scope of its original medical indication [NSC21]. In very recent years, the drug repurposing task based on biological KGs has attracted much attention of researchers. There exist few drug-repurposing frameworks

in the literature which compute rank of each drug in a candidate drug set for a disease and recommend few top-ranked drugs for the disease [ZCJ+20, ASGR+21, ZHS+21]. Few of them rely on simple statistical methods. Al *et al.* [ASGR+21] built a COVID-19 centric biological KG and studied a simple statistical approach based on the number and length of paths between a drug and a disease entity in the KG. They applied their framework for drug repurposing for the COVID-19 disease. Malas *et al.* [MKS+21] propose another statistical framework that uses drug and disease neighbourhood information for the drug ranking task. A majority of the frameworks are based on embedding methods. Some of the embedding-based approaches are specific to few diseases and some are generic. Zhang *et al.* [ZHS+21], Yan *et al.* [YLY+21] presented a drug repurposing framework for the COVID-19 disease which learns embeddings of a COVID-19 related KG using geometric KG embedding methods, such as TransE [BUGD+13], TransH [WZFC14] and RotatE [SDNT18a]. The embeddings of a candidate drug and the COVID-19 disease are utilized in a triple scoring function to compute the link plausibility and the rank of a drug. The framework by Yan *et al.* uses the TransE method to learn embeddings and a drug is ranked by the same mechanism as by Zhang *et al.* Sosa *et al.* [SDG+19] presented a repurposing framework for rare diseases. They used the RotatE embedding method to learn KG embeddings and drug ranking. Zhu *et al.* [ZCJ+20], and Gao *et al.* [GDX22] developed a generic drug repurposing frameworks. Zhu *et al.* used the TransE to learn embeddings of a biological KG and employ a supervised machine learning algorithm to compute the link probability between a drug and a disease. The framework also includes TransH and TransR in experiments to select the best embedding method. The framework by Gao *et al.* uses the RGCN [SKB+18] embedding method to learn embeddings of a KG dedicated to drug repurposing and included a new link scoring function. Ye *et al.* [YHY+21] and Kanatsoulis *et al.* [KS21] learn KG embeddings in their frameworks using tenor factorization based approaches. Ye *et al.* used the RESCAL [NTK11] approach while Kanatsoulis *et al.* developed a new higher order matrix factorization approach called Tex-Graph in their drug repurposing framework.

1.5 Summary

In this chapter, we have introduced the concepts of simple graphs and knowledge graphs with their properties. Then, we have described the formulation of the link prediction problem in both simple and KGs. For simple graphs, we have provided state-of-the-art link prediction approaches. We have seen that the oldest similarity-based approaches are known as explainable, but good performers in only few graphs while the recent embedding-based approaches offer high prediction performance on graphs from various domains, but not explainable. In between, supervised approaches are explainable and perform better than similarity-based ones. For link prediction in KGs, we described different categories of approaches. We briefly discussed few rule-based link prediction approaches for KGs. The symbolic approaches are easily interpretable, but they are often impractical for large scale KGs due to their huge computational time. In addition, the underlying symbolic nature of triples usually makes KGs hard to manipulate. We then discussed various embedding-based approaches for link prediction in KGs. We have seen that these approaches provide good prediction performance in many KGs, but they lack explainability. We also discussed the negative triple sampling aspect of KG embedding methods. Finally, we provided a brief review of published works relative to two applications of link prediction in simple and KGs.

Chapter 2

Explainable link prediction in simple graphs

Contents

2.1 Introduction

Link prediction approaches for simple graphs predict, as we saw in the previous chapter, the probability of a link between two unconnected nodes based on available information in the current graph such as node attributes or graph structure [XPY16]. The prediction of missing or potential links helps us toward the deep understanding of structure and evolution of real-world complex graphs [SWF+14]. A large category of link prediction approaches is based on some heuristics that measure the proximity between nodes to predict whether they are likely to have a link (Section 1.2.2, Chapter 1). Though these heuristics can predict links with high accuracy in many graphs, they lack universal applicability to any kind of graphs. For example, the common neighbor heuristic assumes that two nodes are more likely to connect if they have many common neighbors. This assumption may be correct in social networks, but is shown to fail in protein-protein interaction (PPI) networks where two proteins sharing many common neighbors are actually less likely to interact [KLS+19]. In case of using these heuristics, it is required to manually choose different heuristics for different graphs based on prior beliefs or rich expertise. On the other hand, machine learning methods have shown their impressive performance in many real-world applications like image classification, natural language processing

etc. The built models assume that the input data is represented as independent vectors in a vector space. This assumption is no longer applicable for graph data as graph is a non-Euclidean structure and the nodes in a graph are linked to some other nodes [WPC+20]. To overcome this limitation, a lot of efforts have been devoted to develop novel graph embeddings where the nodes, edges, graphs are represented in a low-dimensional vector space while preserving the graph information [CWPZ18]. To learn the appropriate heuristics automatically from a graph, researchers have developed embedding-based approaches which showed impressive link prediction performance in most of the graphs (Section 1.2.5, Chapter 1). The downside of embedding-based approaches is that they suffer from the well-known 'black-box' problem. As the link decisions are critical in many applications, a link prediction approach should be sufficiently interpretable to achieve trust among stakeholders [GMC20].

In this chapter, we present our first study on a selection of methods from both categories on several benchmark simple graphs with different properties from various domains (Section 2.2). Beyond the intra and inter category comparison of the performances of the methods, we propose a method to uncover interesting connections between embedding-based approaches and heuristic ones as a means to alleviate the black-box well-known limitation. We then present a supervised link prediction approach where several similarity-based metrics are ensembled to improve prediction performance in biological graphs (Section 2.3). In addition, the link predictions are explained with the help of leading heuristics in a graph.

2.2 Appraisal study on similarity-based and embedding-based approaches

There are some review studies in the literature which focus either on similarity-based approaches [MBC16, LZ11] or embedding-based approaches [CWPZ18, CZC18] for link prediction task in graphs. Thus, to the best of our knowledge, a study including methods from both categories was missing in the literature and we aimed to fill this gap. We first select some similarity-based and embedding-based methods. Then, we evaluate their performance on several simple graphs. We compare their performances on diverse graph groups (sharing characteristics). We also propose a few interesting connections between similarity-based and embedding-based methods.

2.2.1 Selection of the link prediction approaches

The similarity-based approach is the most commonly used approach for link prediction which is developed based on the assumption that two similar nodes interact in a graph. Generally, the links between node pairs with high similarity scores are predicted as truly missing links. Numerous similarity-based approaches exist in the literature to predict links in small to large simple graphs (see Section 1.2.2, Chapter 1). The local similarity-based approaches use the local neighbourhood information to compute the similarity score for a pair of nodes. Another category of similarity-based approaches is global approaches that use the global topological information of the graphs for computing similarity scores. The computational complexity of global approaches makes them unfeasible to be applied on large graphs [MBC16]. For this reason, we consider only local similarity-based approaches in the current study. We have studied six popular similarity-based approaches for link prediction. Considering the count of citations for a duration from publishing year to the running year(*i.e.*, 2020), we define popularity of each approach as the average citation count per year.

Table 2.1 summarizes the approaches with the basic principle and a reference to the similarity function. These approaches in Table 2.1 except CCLP(Clustering Coefficient-based Link

Table 2.1: Summary of studied similarity-based approaches. The similarity functions are defined in Section 1.2.2, Chapter1.

Approach	**Principle**	**Similarity function reference**
Adamic-Adar (AA) [AA03]	Variation of CN where each common neighbour is logarithmically penalized by its degree	Equation 1.11
Resource Allocation (RA) [ZLZ09]	Based on the resource allocation process to further penalize the high degree common neighbours by more amount	Equation 1.12
Preferential Attachment (PA) [BA99]	Based on the rich-get-richer concept where the link probability between two high degree nodes is higher than two low degree nodes	Equation 1.27
Hub Promoted Index (HPI) [RSM$^+$02]	Promoting link formation between high-degree nodes and hubs	Equation 1.18
Local Leicht-Holme-Newman (LLHN) [LHN06]	Utilizing both of real and expected amount of common neighbours between a pair of nodes to define their similarity	Equation 1.19
Clustering Coefficient-based Link Prediction (CCLP) [WLWG16]	Quantification of the contribution of each common neighbour by utilizing the local clustering coefficient of nodes	Equation 1.25

Prediction) [WLWG16] use node degree, common neighborhood or links among common neighborhood information to compute similarity scores. AA, RA and CCLP handcraft the computation of weight of each common neighbours based on their neighbourhood size or clustering coefficient(CC). The clustering coefficient is defined as the ratio of the number of triangles and the expected number of triangles passing through a node [WLWG16]. On the other hand, HPI, PA and LLHN assign equal weights to neighbours. These local similarity-based approaches except PA work well when the graphs have a high number of common neighbours between a pair of nodes. However, LLHN suffers from outlier (infinite similarity score) when one of the end nodes has no neighbour. HPI also suffers from the outlier (infinite similarity score) when both end nodes have no neighbour. Each of these similarity-based approaches is briefly described in Section 1.2.2, Chapter 1.

On the other hand, a graph embedding approach embeds the nodes of a graph into low-dimensional vector space where connected nodes are closer to each other. As described in Section 1.2.5, embeddings of the end nodes of a link are utilized in a prediction model to compute the probability of the link. Random walk-based, neural network-based, and auto-encoder-based embedding are three popular methods of learning embeddings for simple graphs [CWPZ18]. In this study we choose six embedding-based link prediction approaches including one random-walk based (Node2Vec [GL16]), three CNN-based (WLNM [ZC17], SEAL [ZC18], GAT [VCC$^+$18]), and two auto-encoder based (VGAE [KW16b], LVGAE [SHV19]). We choose Node2Vec to represent simple non-deep learning methods, WLNM to represent the methods which learn only structural features, SEAL to represent the methods which maximize the use of available information (structural, node attributes, latent features), GAT to represent the methods which define different roles of different neighbours, VGAE to represent auto-encoder with deep neural networks, and LVGAE to represent simple linear auto-encoder. Each of these methods was

described in Section 1.2.5, Chapter 1.

2.2.2 Experimental settings

Preparation of graph datasets

We perform our comparative study on six similarity and six embedding based link prediction approaches in simple graphs from different domains. To evaluate and describe the performance of the link prediction approaches, we choose ten benchmark graphs from different areas: Ecoli [SZC+01], FB15K [BUGD+13], NS [New06], PB [A+05],Power [WS98], Router [SMW02], USAir [HHB+03], WN18 [BGWB14], YAGO3-10 [MBS13], and Yeast [VMKS+02]. FB1K, WN18 and YAGO3-10 are popular knowledge graphs. These knowledge graphs consist of (head, relation name, tail) triples. However, as the studied approaches are applicable to simple graphs only, we simplify these knowledge graphs by overlooking the relation names and considering links as undirected links.

Some topological statistics of the graph datasets are summarized in Table 2.2. Based on the number of nodes, these graphs are categorized into small/medium graphs with less than 10,000 nodes and large graphs otherwise.

Table 2.2: Topological statistics of ten benchmark simple graph datasets: number of nodes(#Nodes), links(#Links), average node degree (NDeg), clustering coefficient (CC), network diameter (Diam), graph density (GD), and description. Large graphs are shaded with gray color.

Graphs	#Nodes	#Links	NDeg	CC	Diam	GD	Description
Ecoli	1805	42325	46.898	0.350	10	0.0260	Nodes: Operons in E.Coli bacteria Edges: Biological relations between operons
FB15K	14949	260183	44.222	0.218	8	0.0023	Nodes: Identifiers of Freebase KB entity Edges: Link between Freebase entities
NS	1461	2742	3.754	0.878	17	0.0026	Nodes: Researchers who publish papers on network science Edges: Co-authorship of at least one paper
PB	1222	14407	23.579	0.239	8	0.0193	Nodes: US political blog page Edges: Hyperlinks between blog pages
Power	4941	6594	2.669	0.107	46	0.0005	Nodes: Electrical power stations (e.g. generators, transformers) of western US Edges: Power transmission between stations
Router	5022	6258	2.492	0.033	15	0.0005	Nodes: Network router Edges: Router-router interconnection for providing router-level internet
USAir	332	2126	12.807	0.749	6	0.0387	Nodes: US airports Edges: Link between two airports if there is at least one direct flight between them
WN18	40943	75769	3.709	0.077	18	0.0001	Nodes: Entities (or synsets) corresponds to English word senses Edges: Lexical relations between sysnets
YAGO 3-10	113273	758225	18.046	0.114	14	0.0001	Nodes: Entities (such as movies, people, cities, etc.) in YAGO KB Edges: Relations between entities
Yeast	2375	11693	9.847	0.388	15	0.0041	Nodes: Proteins in yeast Edges: Protein-protein interaction in yeast network

We follow a random sampling protocol to evaluate the performance of the studied approaches [PZLH16, ZC18, ZC17]. We prepare train and test set from the experimental graphs. For training dataset, we randomly select 90% observed links (termed as positive train set) and an equal number of randomly selected unobserved links (termed as negative train set). The

remaining 10% observed links (termed as positive test set) and an equal number of randomly selected unobserved links (termed as negative test set) form the test set. At the same time, the graph connectivity of the training set and the test set is checked. We prepare five train and five test sets for evaluating the performance of the approaches. For evaluating the performance of similarity-based approaches, the graph is built from the positive training dataset whereas, for embedding-based approaches, the graph is built from original graph that contains both of positive train and test datasets. However, a link is temporarily removed from the graph to train it to the embedding-based approaches or to predict its existence. The performance is quantified by defining two standard evaluation metrics, precision and AUC (Area Under the Curve) (Section 1.2.1, Chapter 1). All of the approaches are run on a Dell Latitude 5400 machine with 32GB memory and core i7 (CPU 1.90GHz) processor.

Precision and AUC computation

Precision describes the fraction of missing links which are accurately predicted as existent links [PZLH16]. As described in Section 1.2.1, the predicted links from a test set are ranked in decreasing order of their scores to compute the precision (Equation 1.2). An ideal prediction approach has a precision of 1.0 that means all the missing links are accurately predicted. We set L to the number of observed links in the test set. However, there are some challenges with this optimistic way of computing the precision. What if the similarity score is (close to) 0.0 of the lowest ranked link? This issue creates the difficulty to make a separation between some positive and negative test links. Choosing a threshold when defining L_r could be a potential solution to overcome this problem. The distribution of unnormalized similarity scores are different for graphs from different domains and even for two different datasets from the same domain. Moreover, it is nearly impossible to know the distribution of unnormalized similarity score in advance for graph dataset. These two facts make it infeasible for the user to define the threshold. To overcome this problem, we define a threshold as the average of the maximum and minimum score in top-L links. We compute the number of positive test links in top-L links (as L_r) as those having similarity scores above the threshold.

On the other hand, the metric AUC is defined as the probability that a randomly chosen observed link has a higher similarity score than a randomly chosen unobserved link [PZLH16]. As described in Section 1.2.1, the AUC is computed by comparing n observed and n unobserved link scores (Equation 1.3). We consider half of the total links in the positive test set and negative test set to compute AUC.

The precision scores talk about how many actually positive links exist in the set of predicted positive links. However, it is also important to assess the false positive or false negative results of a link prediction model as the higher the false-positive result, the lower the efficiency of the model [TO13, YLC15]. We use the above computed threshold to classify a link into existent or non-existent class. A test link is predicted as existent if the similarity score of the link is greater or equal to the threshold and predicted as non-existent otherwise. For embedding-based methods, the threshold is learned and test links are already predicted as existent or non-existent. Based on the true and predicted link existence, false positive rate (FPR) and false negative rate (FNR) are computed to assess the false results and one metric (accuracy) to assess the overall accuracy of an approach (Equation 1.9).

2.2.3 Results

The prediction approaches are evaluated in each of the five sets (train and test set) of each graph and performance metrics (precision, AUC) are recorded. We measure the precision in two different ways based on the top-L test links as described in Section 1.2.1. We compute the threshold-based precision only for similarity-based approaches as embedding-based approaches do learn the threshold. The auto-encoder based VGAE and LVGAE approaches require the whole adjacency matrix of a graph in memory. Due to the memory constraint, experiments with auto-encoder based VGAE and LVGAE approaches for large scale graphs (WN18, YAGO3-10) failed. The maximum and minimum similarity scores are computed from the top-L for each test set of each graph. Table 2.3 shows the results in each of the seven small/medium and three large-size graphs. Each value of the table is the mean over the five test sets. The standard deviation values of both metrics for all approaches in all graphs are very small and they are not included in the table.

The comparative analysis

It can be clearly seen from Table 2.3 that the ranges of unnormalized similarity scores are different for different similarity-based approaches and also different in different datasets for the same similarity-based approach. Moreover, the minimum similarity scores are very low (close to 0) in some datasets. These observations prove that in real-world applications, it is difficult to choose a threshold and to assess good precision for similarity-based approaches.
From Table 2.1, the similarity-based approaches are mostly defined based on the common neighbourhood. As expected, they show low precision (without defining threshold) and AUC values in sparse graphs (low CC, low node degree) like Power, router and high precision for other well-connected graphs in Table 2.3. Exceptionally, PA shows better prediction performance in sparse graphs as it considers individual node degree instead of common neighbourhood for computing similarity score. The precision scores using the threshold-based method drops drastically in most of the cases as many falsely predicted positive links are identified (*i.e.*, predicted links with very low scores). Surprisingly, HPI shows competitive threshold-based precision value in NS dataset. No single similarity-based approach wins in all small/medium size graphs.

As expected, embedding-based approaches show very good precision and AUC scores across all the small/medium size graphs compared to similarity-based approaches. What about their comparative performances? No single approach wins in all datasets. Node2Vec shows highest precision scores in some datasets though it is simpler than other embedding-based approaches. The consideration of more distant neighbours in embedding computation during random walk could be the most possible reason behind this success. The use of latent information along with structural information in SEAL for the datasets during prediction task likely explains the improvement of the metric AUC. The best tuning of parameters could be the most possible reason behind the best balance between the prediction metrics in GAT. The precision and AUC scores of auto-encoder based approaches are behind the other embedding-based approaches. Improper tuning of parameters could be the possible reason behind the performance degradation. Table 2.3 shows that embedding-based approaches provide high-performance metrics in all graphs while similarity-based approaches perform well in some graphs (in terms of optimistic precision).
Considering the three large graphs (FB15K, WN18 and YAGO3-10), the prediction metrics for similarity-based approaches are much lower than small/medium scale graphs, especially in WN18 and YAGO3-10 graphs. Likewise the results in small/medium size graphs, the precision scores of these approaches further drops drastically to less than 0.1 when applying the threshold. Un-

Table 2.3: AUC and Precision(Prec) values with Max scores(Mx scr) and Min scores (Mn scr). Precision with * mark(Prec*) is computed based on threshold in top-L links. Graph-wise highest metrics are indicated in bold fonts while approach-wise highest metrics are shown in underline. – denotes the failed experiment.

App.	Metric	Ecoli	NS	PB	Power	Router	USAir	Yeast	FB15K	WN18	YAGO 3-10
AA	Prec	0.90	0.87	0.86	0.17	0.07	0.92	0.83	0.77	0.13	0.15
	Prec*	0.06	0.15	0.01	0.02	0.01	0.16	0.06	0.0002	0.0002	0.0018
	Mx scr	32.84	5.83	33.41	3.04	5.60	16.69	23.71	418.60	57.32	24.44
	Mn scr	2.86	1.14	0.58	0.00	0.00	2.70	0.00	0.12	0.00	0.00
	AUC	0.93	0.94	0.92	0.58	0.54	0.94	0.91	0.82	0.56	0.48
PA	Prec	0.78	0.69	0.83	0.49	0.41	0.85	0.79	0.79	0.63	0.83
	Prec*	0.05	0.02	0.01	0.02	0.01	0.13	0.06	0.0003	0.0006	0.0006
	Mx scr	65679	362.0	61052	53.0	2397	8298	10642	9881842	10637	2426939
	Mn scr	3532	12.0	855.7	4.0	1.0	739.3	95.0	942.67	6.33	109.00
	AUC	0.80	0.66	0.90	0.46	0.43	0.90	0.86	*0.88*	*0.64*	0.88
RA	Prec	0.91	0.87	0.86	0.17	0.07	0.92	0.83	0.77	0.13	0.15
	Prec*	0.03	0.15	0.01	0.03	0.01	0.10	0.07	0.0003	0.0002	0.0011
	Mx scr	1.70	1.80	4.19	0.84	1.32	2.83	2.37	72.06	20.67	5.16
	Mn scr	0.19	0.40	0.03	0.00	0.00	0.32	0.00	0.00	0.00	0.00
	AUC	0.94	0.94	0.92	0.58	0.54	0.94	0.91	0.84	0.57	0.57
HPI	Prec	0.90	0.87	0.80	0.17	0.07	0.91	0.83	0.69	0.13	0.15
	Prec*	0.20	0.96	0.15	0.13	0.02	0.45	0.70	0.0959	0.0796	0.0476
	Mx scr	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Mn scr	0.33	0.83	0.21	0.00	0.00	0.77	0.00	0.05	0.00	0.00
	AUC	0.94	0.94	0.85	0.58	0.54	0.91	0.90	0.75	0.56	0.47
LLHN	Prec	0.89	0.87	0.74	0.17	0.07	0.87	0.83	0.64	0.13	0.15
	Prec*	0.001	0.13	0.001	0.03	0.003	0.03	0.01	0.0008	0.0046	0.0003
	Mx scr	0.32	1.00	0.42	2.06	0.83	0.58	0.67	0.28	1.00	1.00
	Mn scr	0.00	0.10	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00
	AUC	0.91	0.93	0.76	0.58	0.53	0.77	0.90	0.57	0.57	0.45
CCLP	Prec	**0.96**	0.73	0.86	0.08	0.07	0.91	0.82	0.78	0.08	0.14
	Prec*	0.06	0.21	0.01	0.01	0.01	0.18	0.06	0.0015	0.0006	0.0013
	Mx scr	30.6	8.0	27.0	1.2	1.1	21.1	39.2	51.74	1.67	20.77
	Mn scr	1.8	0.3	0.3	0.0	0.0	2.9	0.0	0.01	0.00	0.00
	AUC	**0.95**	0.87	0.91	0.54	0.53	0.94	0.90	0.84	0.54	0.57
WLNM	Prec	0.87	0.84	0.78	0.84	**0.89**	0.85	0.87	0.67	**0.84**	0.68
	AUC	0.93	0.95	0.93	0.76	0.92	0.86	0.86	0.68	0.79	0.72
SEAL	Prec	0.81	*0.96*	0.80	0.66	0.80	**0.91**	0.89	0.77	0.61	**0.86**
	AUC	**0.95**	**0.99**	**0.94**	0.77	**0.94**	**0.94**	**0.98**	**0.96**	**0.87**	**0.97**
GAT	Prec	0.84	0.93	0.84	0.72	0.81	0.88	**0.91**	**0.85**	0.74	0.84
	AUC	0.85	0.90	0.86	0.70	0.79	0.87	0.89	0.87	0.79	0.83
Node2Vec	Prec	0.91	**0.97**	**0.91**	**0.86**	0.80	0.81	0.85	0.79	0.83	0.82
	AUC	0.90	0.96	0.90	**0.82**	0.75	0.85	0.94	0.88	0.79	0.80
VGAE	Prec	0.75	0.91	0.86	0.82	0.80	0.78	0.75	0.81	–	–
	AUC	0.76	0.93	0.87	0.83	0.84	0.80	0.75	0.78	–	–
LVGAE	Prec	0.76	0.88	0.73	0.81	0.81	0.79	0.86	0.74	–	–
	AUC	0.77	0.86	0.73	0.86	0.83	0.80	0.83	0.68	–	–

surprisingly, the prediction scores for embedding-based approaches in large graphs are high as in small/medium scale graphs. The notable point in the prediction metrics for large graphs is that Node2Vec is less competitive than other embedding-based approaches in these large graphs.
We further compute the false positive rate, false negative rate and overall accuracy of the approaches which are tabulated in Table 2.4. Many of observed and most of the test links have very low similarity scores. As a result, false positive rates of similarity-based approaches are very low and false negative rates is very high for almost all graph datasets in Table 2.4. The overall accuracy scores of these approaches are around 0.5. Exceptionally, HPI shows good accuracy in some datasets, especially in NS dataset where the accuracy is more than 0.9. On the other hand, embedding-based approaches learn the threshold. The false positive rates of embedding-

based approaches are much higher than the similarity-based approaches, but the false negative rates are much much lower than the similarity-based approaches. Among embedding-based approaches, the auto-encoder based VGAE and LVGAE have higher false negative rates than other embedding-based approaches. Overall, the accuracy scores of similarity-based approaches stay around 0.5 with a few exceptions. As expected, the accuracy scores of embedding based approaches are much higher than the similarity-based approaches. With respect to accuracy, but GAT shows highest/nearly highest scores in nearly half of the total graphs, no single embedding-based link prediction approach wins in all graphs.

Table 2.4: False positive rate(FPR), false negative rate (FNR) and accuracy(ACC) scores. Graph-wise best metrics (lowest FPR, lowest FNR, highest ACC) are indicated in bold fonts. – denotes the failed experiment.

App.	Metric	Ecoli	NS	PB	Power	Router	USAir	Yeast	FB15K	WN18	YAGO 3-10
AA	FPR	0.0007	**0.0001**	**0.0001**	0.0003	**0.0001**	0.0028	**0.0003**	**0.0001**	**0.0001**	**0.0001**
	FNR	0.962	0.898	0.991	0.977	0.996	0.932	0.956	0.998	0.997	0.998
	ACC	0.519	0.551	0.504	0.512	0.502	0.533	0.522	0.499	0.499	0.499
PA	FPR	0.0007	0.0007	0.0004	0.0018	**0.0001**	0.0019	**0.0003**	**0.0001**	**0.0001**	**0.0001**
	FNR	0.967	0.942	0.993	0.988	0.996	0.952	0.956	0.997	0.999	0.999
	ACC	0.516	0.529	0.503	0.505	0.502	0.523	0.522	0.499	0.499	0.498
RA	FPR	0.0002	**0.0001**	**0.0001**	0.0003	**0.0001**	**0.0000**	**0.0003**	**0.0001**	**0.0001**	**0.0001**
	FNR	0.984	0.854	0.995	0.977	0.994	0.973	0.958	0.998	0.999	0.999
	ACC	0.508	0.573	0.502	0.511	0.503	0.514	0.521	0.499	0.499	0.498
HPI	FPR	0.0060	0.0058	0.0345	0.0012	0.0048	0.1175	0.0075	0.0067	0.001	0.0152
	FNR	0.753	0.160	0.863	0.877	0.974	0.531	0.412	0.904	0.920	0.953
	ACC	0.620	0.917	0.551	0.561	0.510	0.676	0.790	0.543	0.538	0.514
LLHN	FPR	**0.0001**	**0.0001**	0.0008	0.0003	0.0026	0.0114	0.0007	0.0013	**0.0001**	0.0002
	FNR	0.999	0.826	0.999	0.985	0.998	0.988	0.994	0.999	0.995	0.999
	ACC	0.501	0.587	0.499	0.507	0.500	0.500	0.503	0.498	0.501	0.498
CCLP	FPR	0.0006	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.0019	0.0003	**0.0001**	**0.0001**	**0.0001**
	FNR	0.959	0.927	0.991	0.992	0.992	0.919	0.952	0.998	0.999	0.989
	ACC	0.520	0.536	0.504	0.504	0.504	0.539	0.524	0.494	0.499	0.499
WLNM	FPR	0.1005	0.1642	0.2059	0.1310	0.1125	0.128	0.1334	0.3767	0.1466	0.4108
	FNR	0.345	0.178	0.304	**0.323**	**0.048**	0.128	0.148	0.240	0.259	**0.145**
	ACC	0.777	0.829	0.745	0.773	**0.92**	0.872	0.859	0.692	0.797	0.722
SEAL	FPR	0.1774	0.0343	0.2214	0.2079	0.2323	0.0938	0.1044	0.2232	0.3492	0.1314
	FNR	0.231	0.070	0.123	0.591	0.083	0.079	0.154	0.251	0.449	0.190
	ACC	0.796	**0.948**	0.828	0.601	0.843	**0.914**	0.871	0.763	0.601	**0.839**
GAT	FPR	0.1828	0.0723	0.1909	0.2109	0.1328	0.1365	0.1025	0.1909	0.1848	0.1444
	FNR	**0.073**	**0.069**	**0.032**	0.461	0.428	**0.046**	**0.015**	**0.032**	0.3014	0.259
	ACC	**0.872**	0.93	**0.888**	0.664	0.72	0.909	**0.941**	**0.889**	0.7569	0.7983
Node2Vec	FPR	0.1457	0.0255	0.0826	0.1141	0.0541	0.1905	0.1454	0.1884	0.1594	0.1654
	FNR	0.221	0.161	0.195	0.334	0.791	0.209	0.202	0.279	**0.210**	0.237
	ACC	0.816	0.907	0.861	**0.776**	0.577	0.800	0.826	0.7663	**0.815**	0.7988
VGAE	FPR	0.2455	0.0358	0.1306	0.0243	0.0352	0.2427	0.2399	0.1881	–	–
	FNR	0.257	0.661	0.316	0.891	0.861	0.151	0.289	0.304	–	–
	ACC	0.749	0.652	0.777	0.542	0.552	0.803	0.736	0.753	–	–
LVGAE	FPR	0.1661	0.0971	0.2129	0.1478	0.1443	0.2464	0.1406	0.178	–	–
	FNR	0.477	0.319	0.435	0.368	0.400	0.082	0.172	0.514	–	–
	ACC	0.678	0.792	0.676	0.742	0.728	0.836	0.844	0.651	–	–

The performance of the studied methods was also assessed in terms of average computational time (data is available on request). As expected, similarity-based approaches are faster as they don't require training. As for embedding-based approaches, Node2Vec requires the smallest time as it does not use deep NN like the other embedding-based methods. The computational time of auto-encoder based VGAE and LVGAE approaches are lower than NN-based WLNM, SEAL, GAT approaches as VGAE (LVGAE) uses shallow (no) neural network. The highest computational time is seen for SEAL as it utilizes the structural and explicit features like WLNM

and GAT along with latent features like Node2Vec. We also noticed that the computational time of embedding-based methods grows with the size of datasets by more amount than the similarity-based methods.

Performance comparison and link agreement analysis for explainability

Embedding-based link prediction approaches show better performance because they learn heuristics from graphs. However, it is not clear which heuristic(s) are learned. We want to take benefit from this study to get insight of such heuristics by comparing the performances of similarity-based heuristics with performances of embedding-based approaches on the same datasets. In one hand, from Table. 2.1 and Table 2.3, AA, RA and CCLP –which heuristically assign high weights to nodes with high degrees or cluster coefficients – show better precision, accuracy on FB15K, PB, NS, USAir, and Yeast compared to other graphs. GAT also shows better precision and accuracy these graphs than other graphs. This may indicate that GAT learns similar heuristics as AA, RA and CCLP. In the other hand, WLNM considers the role of each neighbour equally like HPI, LLHN, and PA. WLNM, HPI, LLHN and PA show better performance scores on Power, Router, and WN18 graphs, confirming that they are heuristically compatible. The auto-encoder based VGAE and LVGAE approaches assign equal weights to all neighbours of a node when aggregating neighbourhood information and show competitive performance on Power and Router graphs. In addition, LVGAE aggregates hop-1 neighbours of a node using a simple linear model whereas VGAE aggregates upto hop-2 neighbours of a node using 2-layer GCN model. These features indicate the possibility of learning HPI, LLHN and PA heuristics by LVGAE and AA, RA, CCLP by VGAE approach.

In order to further explore their connections, we compute the percentage of agreements in link existence between the embedding-based and the similarity-based approaches. Table 2.5 shows the top-2 ranked similarity-based approaches when they are ranked in decreasing order of their percentage of agreements on each graph for each embedding-based approach. Overall, embedding-based approaches show higher percentages of agreements to similarity-based approaches in small/medium graphs than in large graphs. Considering all graphs, HPI, PA and

Table 2.5: Top-2 ranked similarity-based approaches with higher agreement with embedding-based approach for test link decision. Numbers in () represent the agreement percentages.

Graph	**WLNM**	**SEAL**	**GAT**	**Node2Vec**	**VGAE**	**LGVAE**
Ecoli	HPI(69), RA(69)	LLHN(80), RA(79)	HPI(70), RA(69)	RA(70), LLHN(70)	RA(74), CCLP(74)	RA(72), HPI(72)
NS	CCLP(65), AA(63)	AA(70), CCLP(68)	AA(61), PA(61)	AA(70), CCLP(68)	AA(65), RA(65)	PA(72), LLHN(72)
PB	HPI(68), PA(64)	RA(68), PA(66)	LLHN(61), RA(59)	AA(68), RA(68)	RA(76), CCLP(75)	RA(71), AA(71)
Power	HPI(63), LLHN(63)	PA(63), HPI(62)	AA(67), RA(67)	PA(63), RA(62)	PA(54), AA(51)	PA(54), LLHN(54)
Router	PA(52), LLHN(47)	PA(66), LLHN(51)	CCLP(65), RA(65)	CCLP(69), AA(68)	PA(57), CCLP(54)	PA(52), HPI(52)
USAir	AA(78), CCLP(78)	LLHN(90), HPI(88)	CCLP(77), AA(75)	RA(90), LLHN(90)	RA(81), AA(81)	HPI(69), RA(69)
Yeast	CCLP(75), PA(74)	CCLP(70), AA(69)	CCLP(75), AA(71)	CCLP(70), AA(69)	AA(72), RAI(71)	RA(69), HPI(69)
FB15K	RA(32), HPI(31)	LLHN(30), HPI(28)	HPI(28), LLHN(27)	HPI(26), AA(24)	HPI(42), CCLP(39)	HPI(38), LLHN(36)
WN18	PA(44), LLHN(42)	PA(40), HPI(32)	PA(28), AA(26)	PA(36), CCLP(31)	–	–
YAGO 3-10	PA(34), AA(26)	PA(44), AA(24)	PA(38), CCLP(32)	PA(42), RA(34)	–	–

LLHN are three frequent heuristics which have higher agreement to WLNM and SEAL approaches. On the other hand, AA, RA and CCLP show frequent agreements with GAT. Between auto-encoder based approaches, LVGAE has higher percentage of agreement to hop-1 neighbourhood based PA, LLHN, HPI heuristics as LVGAE aggregates hop-1 neighbourhood information of a node during node encoding. For VGAE, CCLP, AA, RA are three frequent heuristics which have higher percentage of agreement. These agreements align to the previous discussion on the nature of learned heuristics in embedding-based methods. However, low agreement percentage values (in Table 2.5) but high precision scores (in Table 2.3) for embedding-based approaches in many graphs like FB15K, Ecoli, NS suggest the existence of other learned heuristics that are not considered in this study.

2.3 Ensembling similarity-based metrics for supervised link prediction

Many complex biological systems can be well-represented with graphs where a node represents a biological entity (e.g. protein, drug, *etc.*) and a link represents the interaction between two entities. Most real-world biological graphs are incomplete in nature. For example, 99.7% of the molecular interactions in human cells are still not known[STDS+08]. The links in biological graphs must be validated by field and/or laboratory experiments, which are expensive and time consuming. Researchers deployed link prediction approaches to compute the plausibility of a link between two unconnected nodes in a graph to avoid the blind checking of all possible interactions. Similarity-based approaches are the simplest and unsupervised methods of link prediction in biological graphs, which define the plausibility of a link by the similarity between its end nodes. The great advantage of these approaches is their interpretibility which is essential for any biological system [ZLW21]. As seen in the above Table 2.3, each of the similarity-based approaches performs well only in some particular graphs and no one wins in all graphs. The lack of universal applicability of similarity-based approaches motivates researchers to study machine learning methods to automatically learn the heuristics from a graph. To learn the appropriate heuristics automatically from a graph, researchers have developed embedding-based approaches which showed impressive link prediction performance in most of the graphs. The downside of embedding-based approaches is that they seriously suffer from the well-known 'black-box' problem. As the link decisions in biological graphs are critical, a link prediction approach should be sufficiently interpretable to achieve trust among stakeholders [GMC20]. Therefore, the interpretablility requirement may limit the use of embedding-based approaches in real-world biological systems. Another group of link prediction approaches is developed based on traditional supervised learning-based approaches. These approaches extract features from a graph and train a traditional classifier for the link prediction task [CHY11, AHCSZ06, BVRdAL15, BKR10, AEB16, SKS12, KBS+20]. These approaches are nearly as performant as embedding-based approaches and as interpretable as similarity-based approaches in many biological graphs.

Here, we intend to investigate whether the existing similarity-based heuristics collaboratively improve the link prediction performance in biological graphs. We study similarity-based heuristics for feature extraction and utilize the features in supervised learning-based classifiers for link prediction in biological graphs. This is not the first attempt to study supervised learning methods to link prediction problem in graphs, but there are important differences between past works [AHCSZ06, LNK07, DSP11] and this study. The existing approaches mostly focus on node attributes for extracting features which are application dependent and not available

in many real-world biological graphs. In contrast, our supervised learning-based approach is developed based on only the topological features (similarity-based metrics). In this study, we define the feature set by including only local similarity-based metrics as global and quasi-local heuristics are computationally expensive. In addition, we extract few other topological features of nodes and derive link-based features based on end node features. We study these features in combination with supervised machine learning methods for link prediction in biological graphs. As for explainability of our approach, we identify dominant graph features which contribute in predicting the links between pairs of nodes.

2.3.1 Our supervised link prediction approach

The proposed supervised approach is based on two main steps: feature extraction and classifier training.

Feature extraction

The most crucial task of a supervised learning-based classifier is to define an appropriate feature set [AHCSZ06]. We are motivated to use only topological features for defining our feature set as they exist in all kinds of graphs. Our feature set contains twenty topological features which are broadly categorized into two categories: similarity-based and derived link features (Figure 2.1). For similarity-based features, we consider well-known local and quasi-local similarity-based metrics as individual features. Here, we ignore global similarity-based metrics as they require high computational time.

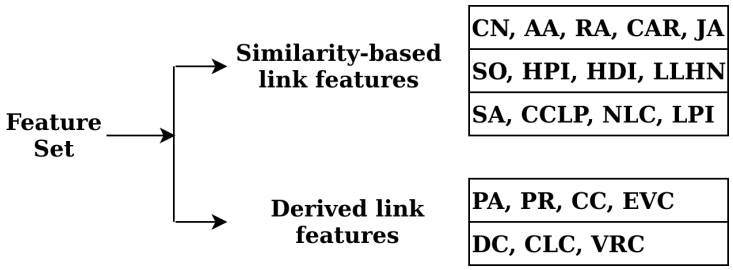


Figure 2.1: Feature set for supervised learning

Similarity-based link features: We define the link-based features as the features which are related to the common topological information of end-nodes of a link. We use thirteen existing similarity metrics as link-based features, which are summarized in Table 2.6. For instance, the number of common neighbours of end nodes of a link is used as the common neighbour(CN) feature.

Derived link features: Few link-based features are derived from the individual features of the link's end nodes. We summarized six derived features in Table 2.7. These features are related to the topological information of individual nodes only. For example, the degree of end nodes is multiplied in Preferential Attachment (PA) to define the similarity score. Note that the link features in Table 2.7 except PA are not directly defined in the literature. We derive the link features based on the end node features. As the voterank centrality computes low ranks for high-influencing nodes in a graph, the reciprocals of the voterank scores of end nodes are summed to define the voterank centrality feature. In general, the magnitude scale for different features in

Table 2.6: Similarity functions (described in Section 1.2.2) for link features

Similarity function/ link feature	**Principle**	**Link feature function**
CN [LW71]	More neighbors between two nodes increase the likelihood of a link between them	Equation 1.10
RA [ZLZ09]	Based on the resource allocation process to further penalize the high degree common neighbours by more amount	Equation 1.12
HPI [RSM+02]	Promoting link formation between high-degree nodes and hubs	Equation 1.18
LLHN [LHN06]	Utilizing both of real and expected amount of common neighbours between a pair of nodes to define their similarity	Equation 1.19
CCLP [WLWG16]	Quantification of the contribution of each common neighbour by utilizing the local clustering coefficient of nodes	Equation 1.25
LPI [LJZ09]	Counting number of length-2 and length-3 paths between end nodes	Equation 1.33
AA [AA03]	Variation of CN where each common neighbour is logarithmically penalized by its degree	Equation 1.11
JA [Jac01]	Normalization of CN where the similarity score is penalized for each non-common neighbour	Equation 1.14
SA [SM83]	Variant of JA which considers common and non-common neighbours to compute the end nodes similarity	Equation 1.15
SO [Sor48]	Describes the overall proportion of common neighbours from a local perspective of end nodes	Equation 1.16
HDI [RSM+02]	Promoting link formation between low-degree nodes and hubs	Equation 1.17
CAR [CALR13]	Consideration up to hop-2 neighbourhood or level-2 link information that connects two common neighbours of end nodes	Equation 1.23
NLC [WLWJ16]	Quantification of the contribution of common neighbours of end nodes based on both their node and link clustering coefficients	Equation 1.26

different graphs varies [IAST21a, IAST21b, IAST20b]. Supervised learning-based approaches are easily affected by the non-uniform scaling as there is a risk that features with higher magnitude play a more decisive role during the training of a classifier. As it is not desirable for the classifier to be biased towards one particular feature, we normalize each feature in the range of [0-1].

Classifier training and link prediction

For the link prediction task, we train a traditional supervised machine learning classifier to classify a link into either existent or non-existent classes. There exist many classifiers in the literature which perform better than others in some particular datasets. In this study, we study three traditional classifiers: Support Vector Machine(SVM) with RBF kernel, Decision Tree

Table 2.7: Summary of derived link features: The derived link feature function $S(x, y)$ is defined based on end nodes features.

Feature	Principle	Link feature function
Preferential Attachment (PA) [BA99]	Based on the rich-get-richer principle, in which the link likelihood between two high-degree nodes is greater than that between two low-degree nodes.	$PA(x, y) = \|\Gamma x\| \times \|\Gamma y\|$
Pager Rank (PR) [LM05]	Pager rank computes a ranking of the nodes based on the structure of the links.	$PR(x, y) = PR(x) + PR(y)$
Clustering Coefficient (CC) [OSKK05]	The clustering coefficient of a node is the fraction of possible triangles through that node that exist	$CC(x, y) = CC(x) + CC(y)$
Degree Centrality (DC) [DWCY13]	The degree centrality for a node is the fraction of nodes it is connected to.	$DC(x, y) = DC(x) + DC(y)$
Eigen vector centrality (EVC) [Bon87]	Eigenvector centrality computes the centrality for a node based on the centrality of its neighbors.	$EVC(x, y) = EVC(x) + EVC(y)$
Closeness Centrality(CLC) [Fre79]	Closeness centrality of a node is the reciprocal of the average shortest path length to other reachable nodes.	$CLC(x, y) = CLC(x) + CLC(y)$
Vote Rank Centrality(VRC) [ZCDZ16]	Ranking of the nodes based on a voting scheme where a node casts votes to its neighbours. A node with the highest votes has the best (lowest) ranking.	$VRC(x, y) = \frac{1}{VRC(x)} + \frac{1}{VRC(y)}$

(DT), and Logistic Regression (LR). We extract the features of the test links and classify them into existent or non-existent classes using a trained classifier to evaluate the link prediction performance.

2.3.2 Experimental settings

To compare the prediction performance of supervised learning approaches, we consider two categories of link prediction approaches: similarity-based and embedding-based approaches. For the similarity-based category, we consider the fourteen similarity-based link prediction approaches (CN, AA, RA, JA, SA, SO, HPI, HDI, LLHN, CAR, CCLP, NLC, PA, LPI) which are considered for defining the feature set. For the embedding-based approaches, we choose two popular approaches: Node2Vec [GL16] and SEAL [ZC18] (described in Section 1.2.5).

The biological graph datasets

In this study, we focus on only simple biological graphs. For evaluating performance, we collect ten biological graphs from the Network Repository [5]. Table 2.8 summarizes some topological statistics and descriptions of the graph datasets.

[5]https://networkrepository.com/bio.php

Table 2.8: The graph datasets: number of nodes($|\mathbf{V}|$), links($|\mathbf{E}|$), average node degree (NDeg), average clustering coefficient (CC), and description.

Graph	**Organism**	**|V|**	**|E|**	**NDeg**	**CC**	**Description**
CE-GT [DA05]	Worm	924	3239	7.01	0.605	Nodes: Genes in C. elegans worm Links: Gene functional associations in C.elegans
CE-HT [DA05]	Worm	2617	2985	2.28	0.008	Nodes: Proteins in C. elegans worm Links: High-throughput protein-protein interactions
C. elegans [DA05]	Worm	453	2040	9.01	0.647	Nodes: Substrates in C. elegans worm Links: Metabolic reactions between substrates in C.elegans
CE-LC [CSH+14]	Worm	1387	1648	2.37	0.076	Nodes: Proteins in C.elegans worm Links: Small/medium-scale protein-protein interactions (compiled from protein-protein interaction data-bases)
Diseasome [GCV+07]	Human	516	1188	4.61	0.636	Nodes: Known genetic disorders in H.sapiens Links: Connections between pair of disorders when they share at least one gene.
DM-HT [GCV+07]	Fly	2989	4660	3.12	0.009	Nodes: Proteins in D.melanogaster fly Links: High-throughput protein-protein interactions
DM-LC [CSH+14]	Fly	658	1129	3.43	0.105	Nodes: Proteins in D.melanogaster fly Links: Small/medium-scale protein-protein interactions (compiled from protein-protein interaction data bases)
HS-HT [CSH+14]	Human	2570	13691	10.65	0.169	Nodes: Proteins in human Links: Protein-protein interactions in human protein network
SC-LC [CSH+14]	Yeast	2004	20452	20.41	0.168	Nodes: Proteins in S.cerevisiae yeast Links: Small/medium-scale protein-protein interactions in yeast network
Yeast [CSH+14]	Yeast	2114	2277	2.15	0.059	Nodes: Proteins in S.cerevisiae yeast Links: Protein-protein interactions in yeast network

The link prediction performance is evaluated using a random sampling validation protocol [ZC17, ZC18]. For a graph dataset, train and test sets are prepared by splitting the existent links. The train set consists of 90% existent and an equal number of non-existent links. The test set contains the remaining 10% existent and equal number of non-existent links. To prepare five train and five test sets for each graph, we repeat the link splitting operation five times independently.

Evaluation metrics for supervised link prediction

The link prediction problem is considered as a binary classification problem [KSSB20]. A traditional classifier-based link prediction approach, in general, learns a threshold to classify links as existent or non-existent. However, for similarity-based link classification approaches, we find no standard approach for computing the threshold. The threshold is calculated in an optimistic manner. We first normalize the link scores to a range of [0-1] and then use the normalized scores to compute a ROC curve. The curve gives the true positive rate (TPR) and false positive rate (FPR) for different score threshold settings. The threshold point with the highest [TPR+(1-FPR)] is computed as the *threshold* as we want to maximize TPR as well as minimize FPR. We classify links based on this threshold. A link with a $score >= threshold$ is classified as existent and non-existent otherwise. Based on link classification outcomes (described in Section 1.2.1), we compute precision, recall, and F1 score (using Equation 1.9) to evaluate our supervised

approach.

2.3.3 Results and discussion

In this section, we describe the prediction performance of supervised learning-based approaches on ten biological graphs. We also illustrate the importance of the features in graphs.

Link prediction performance

The link prediction performance is computed for all approaches over all the five test sets for the ten graphs, and the average scores are recorded. We do not include the standard deviation results as the values are very low in all the experiments. The precision, recall and F1 scores are tabulated Table 2.9, where the best two similarity-based approaches are denoted with Sim^1 and Sim^2. The precision scores of similarity-based approaches (best and second best based on precision scores) are very high and highest among all the approaches in all the graphs, as shown in the table. This demonstrates the ability of similarity-based approaches to predict high-quality links. However, the recall scores are low, implying that these approaches identify the majority of existing test links as non-existent. As a result, the F1 score for similarity-based approaches is very low. We also see that, as expected, the two best-performing similarity-based approaches differ for different datasets. Among the supervised learning approaches (SVM, DT, LR), DT shows the worst prediction results, but it is still much better than similarity-based approaches. The other two classifiers, SVM and LR, have similar performance scores. The performance of these two classifiers in terms of prediction scores is impressive. Yet, supervised learning-based approaches show superior prediction performance than embedding-based approaches in many graphs. Relating the performance to graph properties, we see that traditional classifiers outperform embedding-based approaches in dense graphs. This is intuitive as the majority of the studied similarity-based approaches are based on common neighbours (see Table. 2.6). The performance scores of traditional classifiers are worse in the sparse graphs (CE-HT, CE-LC, Yeast), where embedding-based approaches show better performance scores.

Feature importance for explaining prediction

In this section, we investigate the influence of each feature in a classifier for the link prediction task. To compute the feature importance coefficient, we use the Permutation importance module from the sklearn python-based machine learning tool [6]. When a feature is left out, the coefficient is calculated by looking at how much the score (accuracy) drops [Bre01]. The higher the coefficient, the higher the importance of the feature. In Figure 2.2, we demonstrate the feature importance in the HS-HT biological graph in the logistic regression (LR) classifier to investigate how the importance of features differs in different sets of the same biological graph. In the LR classifier for the HS-HT biological graph, four features dominate. The dominance of multiple heuristics or features in a graph shows that heuristics collaborate well than heuristics that work alone. We can also observe that the feature importance coefficients in all five sets in the HS-HT graph are substantially identical.

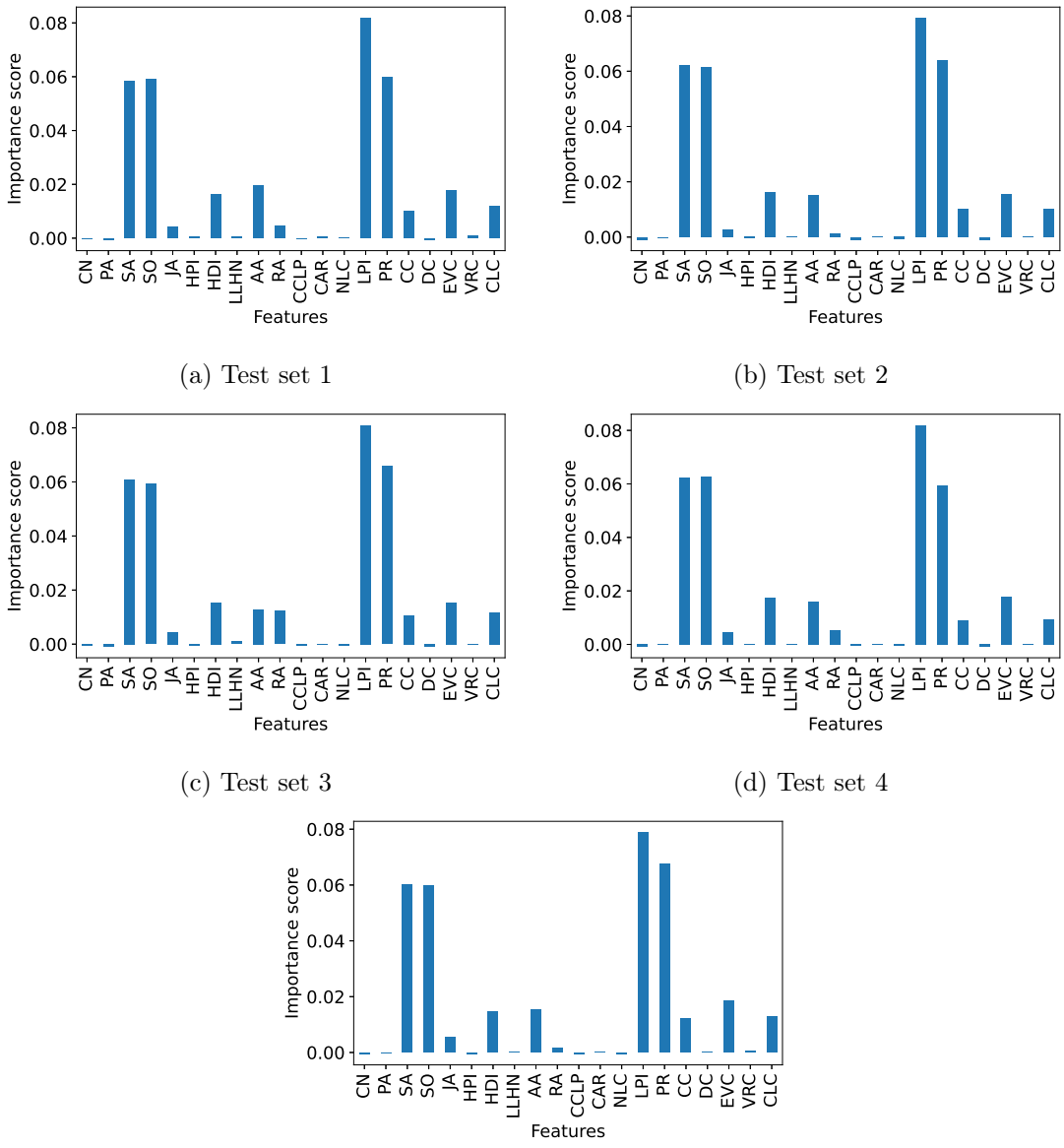
We further investigate the importance score of features in three classifiers (SVM, DT, LR) for three different datasets (Figure 2.3). We evaluate the importance score of features for only one test set for each graph. We see that different classifiers give different importance coefficients to different features in different datasets. Interestingly, all the classifiers compute high coefficient

[6]https://scikit-learn.org/stable/modules/generated/sklearn.inspection.permutation _importance.html

Table 2.9: Performance metrics: The dataset-wise best and second best precision, recall and F1 scores are indicated in bold and underline. The best and second best similarity-based approaches are denoted with Sim^1 andSim^2 respectively. For Sim^1 and Sim^2 approaches, the approaches are specified and the performance scores are given in ().

Datasets	Metric	Sim^1	Sim^2	N2V	SEAL	SVM	DT	LR
CE-GT	Precision	NLC **(0.960)**	JA (0.896)	0.707	0.842	0.842	0.828	0.901
	Recall	NLC (0.039)	JA (0.042)	0.707	**0.931**	0.827	0.776	0.900
	F1	NLC (0.075)	JA (0.078)	0.707	0.885	0.834	0.801	**0.901**
CE-HT	Precision	RA **(0.996)**	AA (0.996)	0.596	0.705	0.753	0.745	0.752
	Recall	RA (0.001)	AA (0.001)	0.593	**0.791**	0.529	0.519	0.510
	F1	RA (0.002)	AA (0.002)	0.594	**0.745**	0.622	0.612	0.608
C. elegans	Precision	RA **(0.938)**	CCLP (0.932)	0.778	0.806	0.899	0.850	0.907
	Recall	RA (0.042)	CCLP (0.041)	0.777	0.888	0.899	0.830	**0.906**
	F1	RA (0.08)	CCLP (0.079)	0.778	0.845	0.899	0.840	**0.906**
CE-LC	Precision	AA **(0.969)**	RA **(0.969)**	0.658	0.763	0.715	0.763	0.789
	Recall	AA(0.009)	RA(0.009)	0.647	**0.794**	0.620	0.584	0.673
	F1	AA(0.028)	RA(0.028)	0.652	**0.778**	0.664	0.662	0.726
Diseasome	Precision	NLC **(0.991)**	AA (0.988)	0.757	0.914	0.926	0.800	0.927
	Recall	NLC (0.035)	AA (0.040)	0.756	0.896	0.919	0.692	**0.920**
	F1	NLC (0.067)	AA (0.078)	0.756	0.905	0.922	0.742	**0.924**
DM-HT	Precision	CCLP **(0.999)**	NLC (0.998)	0.712	0.720	0.780	0.796	0.770
	Recall	CCLP (0.001)	NLC (0.001)	**0.704**	0.703	0.657	0.661	0.644
	F1	CCLP (0.002)	NLC (0.002)	0.708	0.712	0.714	**0.722**	0.701
DM-LC	Precision	PA **(0.979)**	CCLP(0.944)	0.696	0.790	0.829	0.828	0.812
	Recall	PA(0.02)	CCLP(0.007)	0.688	**0.835**	0.771	0.770	0.777
	F1	PA(0.039)	CCLP(0.014)	0.692	0.812	**0.799**	0.798	0.794
HS-HT	Precision	NLC **(0.954)**	CCLP (0.949)	0.797	0.854	0.861	0.847	0.861
	Recall	NLC (0.031)	CCLP (0.031)	0.794	0.815	0.840	0.791	**0.848**
	F1	NLC (0.060)	CCLP (0.061)	0.796	0.834	0.850	0.818	**0.854**
SC-LC	Precision	NLC **(0.893)**	AA (0.873)	0.772	0.784	0.868	0.850	0.853
	Recall	NLC (0.035)	AA (0.036)	0.770	0.815	**0.849**	0.810	0.844
	F1	NLC (0.067)	AA (0.068)	0.771	0.799	**0.859**	0.829	0.849
Yeast	Precision	CCLP **(0.971)**	RA (0.967)	0.699	0.705	0.753	0.746	0.755
	Recall	CCLP (0.006)	RA (0.008)	0.699	**0.726**	0.567	0.551	0.598
	F1	CCLP (0.012)	RA (0.015)	0.699	**0.716**	0.647	0.634	0.668

for LPI feature in the DM-HT dataset and they have close prediction performance (in Table 2.9). In the C. elegans dataset, the HPI feature dominates in SVM and LR classifiers whereas LPI dominates in the DT classifier. In the C. elegans dataset, SVM and LR outperform DT in terms of prediction (in Table 2.9), demonstrating that LR and SVM compute feature importance scores more correctly in this data set. Surprisingly, we see that DT has a tendency to give more importance to the LPI feature in these three datasets.



(a) Test set 1

(b) Test set 2

(c) Test set 3

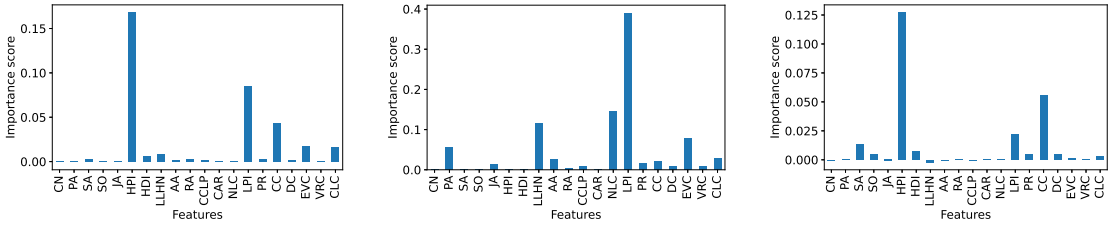
(d) Test set 4

(e) Test set 5

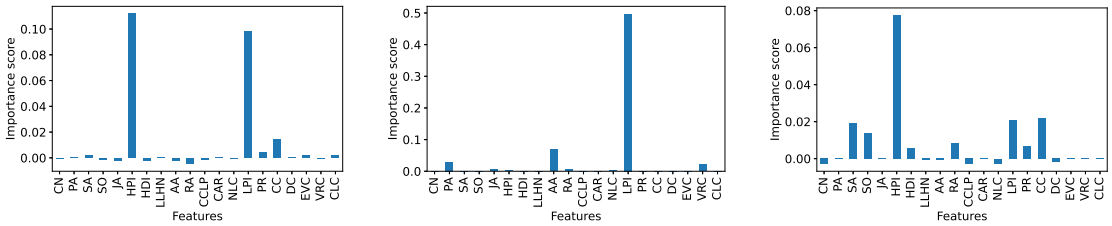
Figure 2.2: Feature importance in HS-HT graph for the LR method

2.4 Conclusion

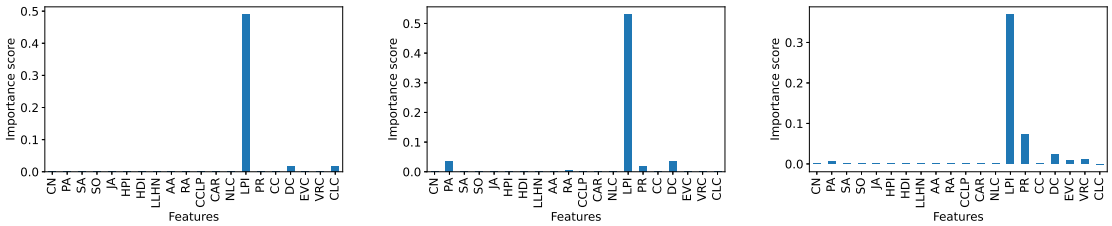
In this study, we first studied several link prediction approaches, looking for their performances and connections. We focused on two categories of approaches: similarity-based approaches and embedding-based learning approaches. The studied approaches were evaluated on ten graph datasets with different properties from various domains. The precision of similarity-based approaches was computed in two different ways to highlight the difficulty of tuning the threshold for deciding the link existence based on the similarity score. The experimental results show the expected superiority of embedding-based approaches. Still, each of the similarity-based approaches is competitive on graphs with specific properties. The possible links between the handcrafted



(a) SVM in C. elegans (b) DT in C. elegans (c) LR in C. elegans



(d) SVM in Diseasome (e) DT in Diseasome (f) LR in Diseasome



(g) SVM in DM-HT (h) DT in DM-HT (i) LR in DM-HT

Figure 2.3: Feature importance in different datasets for different supervised approaches: (a)-(c) in C. elegans, (d)-(f) in Diseasome, (g)-(i) in DM-HT

similarity-based approaches and current embedding-based approaches were explored using (i) prediction performance comparison to get an idea about the learned heuristics and (ii) agreement percentage on the diverse graphs. Our observations constitute a modest contribution to the 'black box' limitation of embedding-based approaches.

We then studied the question, "Do similarity-based heuristics compete or collaborate for link prediction task in graphs?". We developed a supervised link prediction approach for biological graphs where few similarity-based metrics are utilized to define the feature set. In addition, we derive six link features based on the node's topological information. The results show that the similarity-based features collaboratively improve link prediction performance remarkably, even outperforming embedding-based approaches in some graphs. We explain the predictions based on identifying dominant structural features which contribute to link prediction. We leave the study of collaboration of similarity-based heuristics in large biological as well as social graphs as potential future works as the graphs in the current study are small/medium in size.

Chapter 3

Negative sampling and rule mining for explainable link prediction in knowledge graphs

Contents

3.1 Introduction

As we saw in Chapter 1, many embedding models were developed to learn vector representations (or embeddings) of entities and relations in KGs. These models perform the link prediction task based on the learned embeddings. The training of these models requires positive triples as well as negative/non-observed triples. However, only positive triples are available in KGs. Importantly, the quality of negative triples does matter [CW18, ZYSC19]. This statement brings the importance of sampling negative triples. Unfortunately, this important perspective of embedding model is less focused in the literature. One way to sample negatives is designed based on 'closed-world' assumption where all of the non-observed triples are necessarily false and used as negatives. However, this assumption is not entirely true for KGs due to their incompleteness [WMWG17]. Alternatively, most of the KG embedding models sample negatives from non-observed triples under 'open-world' assumption where a non-observed triple may be positive or negative. To the best of our knowledge, there exist four negative sampling methods: uniform-random [BUGD$^+$13], GAN-based [CW18, HLH19, WLP18], NSCaching [ZYSC19], and

self-adversarial [SDNT18a] (described in Section 1.3.4). However, each of them has its own pros-cons and the current state-of-the-art still lacks a good negative sampling method. Indeed, their prediction performance are affected by the 'vanishing gradient' problem because of poor quality of sampled negative triples. In addition, they suffer from computational and memory inefficiency problems. In this chapter, we propose a simple but efficient method called SNS to sample high-quality negative triples in KGs. We design the SNS based on the assumption that the entities which are closer in the embedding space to the corrupted entity are able to provide high-quality negative triples. In sampling, the trade-off between exploration and exploitation is crucial in searching for a high-quality samples [CXP+09]. Exploration corresponds to the capacity of the sampling method to select high-quality negative triples from unexplored areas whereas exploitation favours the utilization of already known negative triples to sample other negatives. SNS makes a good balance between exploration and exploitation to improve the quality of negative triples. We design SNS as a general sampling method that can be plugged to any KG embedding model for link prediction.

Beside negative triple sampling, explainability is another important aspect for KG embedding methods as they lack explainability. This limits their ability to be deployed in many real-world applications which seems to be particularly absurd given the richness of the initial knowledge graphs. Within this context, we concern about not only prediction performance, but also explanations of the predictions. Few researchers studied the reinforcement learning in knowledge graphs to provide explanations of their link predictions [ENRP22, DDZ+18, LHJ+21]. Usually, these approaches define link prediction as a Markov decision process where they learn a policy to walk from heads to tails of triples in a knowledge graph. These approaches are able to find correct explanations for few predictions. However, they suffer from the well-known degeneracy and instability problems [BB09, DBPL19, RLO+21] which are common problems in reinforcement learning methods. Moreover, these approaches generate single explanation of each prediction though the existence of multiple explanations for a prediction is not ignorable. Rossi *et al.* [RFMT22] developed Kelpie, an explainability framework to explain any embedding-based link predictions approaches based on empirical results. For a given prediction, Kelpie identifies a minimum set of facts that is necessary to enable the prediction.

Apart from the aforementioned studies, two methods exist in the literature that aim at mining rules from learned embeddings: (1) EmbedRule [YYH+15a] and (2) RLvLR (Rule Learning via Learning Representations) [OWW21] (described in Section 1.3.2). In this chapter, we add a new rule mining method to the literature that utilizes the KG and its embedding and a strategy to use the rules to explain the embedding-based link prediction. Our method differs from EmbedRule and RLvLR in several points. Firstly, both existing methods mine rules based on closed-path (CP) pattern, whereas our method considers more path patterns for mining better rules. Secondly, our method is able to detect and remove noisy rules early enough to reduce memory and computational time requirements. Finally, the existing methods only consider relation embedding for mining rules, thus losing the entity embedding information as the validity of a triple depends on both entity and relation embeddings. In contrast, our method utilizes the triple scoring function of an embedding method to define the path scoring function so as to exploit both entity and relation embeddings in mining high-quality rules.

As for the evaluation of our SNS sampling method, we perform link prediction task on several KG datasets. Experimental results show that the SNS improves the prediction performance of KG embedding models, and outperforms the existing sampling methods. We also evaluate our rule mining method in terms of prediction recall and quality metrics. We see that our rule mining method is able to extract rules in a minimum time. We also observe that many predicted links by an embedding-based can be explained successfully with the help of our rule mining method.

The rest of the chapter is organized as follows. After describing the SNS sampling method and its comprehensive evaluation in Section 3.2, we present our rule mining method along with an explanation strategy for predicted triples by a KG embedding-based approach in Section 3.3. The last section is devoted to the conclusions and the future directions.

3.2 The SNS method

In this section, we first describe our SNS negative triple sampling method (Section 3.2.1). Then we provide a comprehensive evaluation of the SNS method on several KG datasets (Section 3.2.2).

3.2.1 Negative triple sampling using SNS method

For link prediction, KG embedding models are trained with positive and negative triples to learn embeddings of entities and relations. In this section, we first briefly describe a geometric KG embedding model, and then our proposed SNS negative sampling method. In this chapter, we use $\mathbb{E}$ to denote the set of all entities, $\mathbb{R}$ to denote the set of all relations, $\mathbb{S}$ to denote the set of positive training triples, $\mathbb{D}$ to denote the set of positive test triples, $\mathbb{Q}$ to denote the set of all positive triples, d to denote embedding dimension size, m to denote the batch size, S_m and S'_m to denote a batch of positive and respective negative triples. The architecture of a geometric KG embedding model is given in Figure 1.15 which starts with initializing the embeddings of entities and relations randomly from uniform/Gaussian distributions [WMWG17]. For the training of the model, a batch of positive train triples S_m is fetched from the train triple set, $\mathbb{S}$. A negative sampling method is then used to generate a batch of negative triples. In the architecture, we use our SNS method to generate the batch of negative triples, S'_m for the batch of positive triples,S_m. The batches of positive and negative triples are then used by the pairwise training strategy to learn the embeddings. In pairwise training, the model tries to assign higher plausibility score to a positive triple than its corresponding negative triple. The pairwise loss for a positive $(h, r, t) \in S_m$ and its corresponding negative triple $(h', r, t') \in S'_m$ is defined in Equation 3.1 [WMWG17].

$$L(f(h, r, t), f(h', r, t')) = \Big[\lambda - f(h, r, t) + f(h', r, t')\Big]_+ \tag{3.1}$$

Here, λ is the margin and $[.]_+ = max(0, .)$ is the hinge function. The training objective is to optimize the embeddings of entities and relations for minimizing the total pairwise loss as Equation 3.2.

$$\min_{\Theta} \sum_{\forall_{(h,r,t)\in S_m, (h',r,t')\in S'_m}} L(f(h, r, t), f(h', r, t')) + \lambda reg(\Theta) \tag{3.2}$$

Here, f is the scoring function of the embedding model, $(h, r, t) \in S_m$ is a positive triple and $(h', r, t') \in S'_m$ is the corresponding negative triple. The embedding updating process is repeated for all batches of positive triples, and the whole training process is repeated for T times (or epochs). The model training process is similar to a traditional geometric embedding model training (illustrated in Figure 1.15) except we adapt our negative sampling method.

In the following, we describe our proposed SNS method for negative sample generation (Figure 3.1). The SNS method aims to generate high-quality negative triples while avoiding the 'vanishing-gradient' problem of uniform-random sampling, the complex parameter optimization problem of GAN-based sampling and the excessive memory requirement problems of NSCaching. Figure 3.1a shows the basic steps of SNS sampling. The steps for each picked positive triple $q = (h, r, t) \in \mathbb{S}$ are described in the following.

Step 1. Initial negative triple set generation: From Figure 3.1a, SNS starts with generating an initial set of negative triples for the positive triple (q) by perturbation. This step is similar to other sampling methods. In triple perturbation, the head (h)/tail (t) of the positive triple is corrupted by replacing head/tail with other entities in the entity set($\mathbb{E}$). In the same time, it is checked that the negative set does not contain any positive triple.

For clarity, lets consider only tail perturbation. Corrupting the tail(t) gives the initial negative set $q'_0(t) = \{(h, r, t') \notin \mathbb{Q} | t' \in \mathbb{E}\}$.

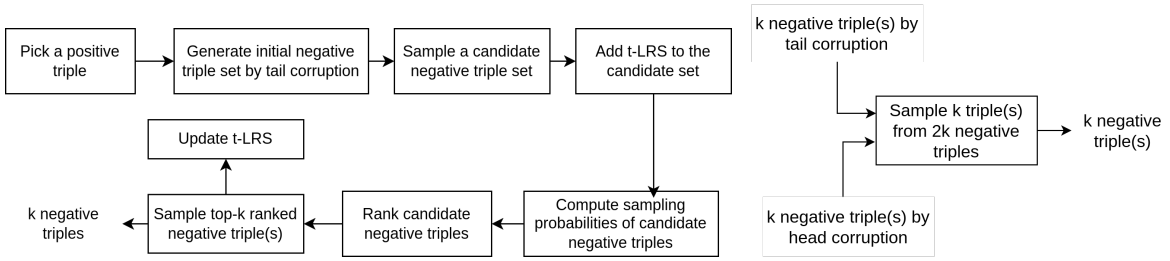


Figure 3.1: The SNS sampling method (a) generation of k high-quality negative triples by corrupting tail (k negative triples are also generated by corrupting head), (b) k negative triple(s) sampling from 2k high-quality negatives

Step 2. Candidate set generation: Generally, the size of the set $q'_0(t)$ is large as KG contains a large number of entities. Zhang et al [ZYSC19] describe that only some of initial negatives of the set are of good-quality. As we need few negatives for each positive triple, we randomly sample N_1 triples from $q'_0(t)$ to generate candidate negative set $q'_1(t)(i.e., q'_1(t) \subseteq q'_0(t))$ for user-defined parameter N_1.

As the training progresses, it is desired that quality of the negative(s) for the next step will be better or close to the quality of negative(s) in the current step. For this purpose, the sampled negatives for each positive triple in the current epoch are stored in a small structure called least recently selected(LRS). Let's say, $LRS_q = (h - LRS_q) \cup (t - LRS_q)$ stores the negative triples for the positive triple (q) where $h - LRS_q$ and $t - LRS_q$ are the sampled negatives by head and tail perturbation respectively in the previous epoch. The saved negatives are used in the current step for sampling high-quality negatives. As we describing the negative generation process for tail perturbation, $t - LRS_q$ is added to the candidate negative set, $q'_1(t)$, $i.e.$, $q'_1(t) := q'_1(t) \cup t - LRS_q$. The use of LRS is intended to favour the exploitation behavior of our SNS method. It ensures that the quality of the negative(s) at current step is better than or at least close to the quality of the negative(s) at the previous step.

Step 3. Sampling probability computation: In this step, we compute the sampling probability of each negative in the candidate negative set $q'_1(t)$. Negatives with higher probabilities are considered as high-quality negatives. The probability is defined based on the distance score of each negatives. The distance score for a negative triple $(h, r, t'_i) \in q'_1(t)$ to the positive triple $(q = (h, r, t))$, is computed as the distance between the corrupted(t) and the new entity(t') as Equation 3.3.

$$d(\ (h, r, t), (h, r, t'_i)\ ) = ||\mathbf{t} - \mathbf{t'_i}|| \tag{3.3}$$

Here, $\mathbf{t}$ and $\mathbf{t'_i}$ are the embeddings of the entities t and t'_i. A softmax function is then used to compute the sampling probability score of each candidate negative $(h, r, t'_i) \in q'_1(t)$ as Equa-

tion 3.4.

$$P(h, r, t'_i) = \frac{exp(\frac{1}{d(t,t'_i)})}{\sum_{j=1}^{N_1} exp(\frac{1}{d(t,t'_j)})} \tag{3.4}$$

The softmax function computes higher sampling probability for the candidate negative triples with lower distance score to support our assumption.

Step 4. Negative triple sampling and LRS updating: The triples from the candidate negative set,$q'_1(t)$ are ranked in decreasing order of their probabilities and k negative(s) are sampled. A natural choice could be sampling top-k (k=1 for pairwise training, $k > 1$ for maximum likelihood training) negative(s). However, sampling top-k ranked negative(s) could arise two problems. Firstly, there is a chance of high repeat sampling of the same negative(s) (even in many consecutive steps) as the current candidate negative set also includes the least recently sampled (LRS) high-quality negative(s). This would affect the exploration capacity of SNS sampling. Secondly, the existence of false negative triples is not ignorable as both hard negative and positive triples have high scores [WMWG17]. To minimize these problems, SNS randomly samples k negative triples from N_2 top ranked triples in $q'_1(t)$ as $q'_t = \{(h, r, t'_i)|rank_{(h,r,t'_i)\leq N_2}\}$ for an user defined parameter N_2 where $N_2 > k$. As SNS samples negative triples from N_2 top ranked negative triples, it avoids the top-1 ranked false negative triples for many positive triples. SNS repeats the above described steps (from initial negative set generation to k negative(s) sampling) for head(h) corruption also to sample k negative(s) as q'_h for the positive triple, q. The LRS_q is updated with the sampled 2k high-quality negative(s) as $h - LRS_q = q'_h,\ t - LRS_q = q'_t$. Finally, we randomly sample k high-quality negative(s) as $q'_{h,r,t}$ from 2k negatives in $q'_h \cup q'_t$ (Figure 3.1b). The sampled k negative(s), q' and the corresponding positive, q are then added to the training set for KG embedding models.

3.2.2 Evaluation of the SNS method

Experimental settings

To evaluate the efficiency of SNS sampling, we plug it to a geometric KG embedding model. Geometric embedding models are broadly categorized into two main categories: (1) translational models and (2) semantic matching models [WMWG17]. Translational models consider each relation as a translation in the embedding space: adding relation to the head gives a close position to the tail [BUGD+13]. On the other hand, semantic matching models define the plausibility of a triple by matching representations of entities and relations embodied in their embeddings [WMWG17]. As for evaluation of our SNS sampling method, we choose TransH [WZFC14] to represent translational and DistMult [YYH+15a] to represent semantic matching model as they are popular baselines for link prediction task in KGs. These models are briefly discussed in Section 1.3.3. Each of the model is evaluated combined with three sampling methods (*i.e.*, random-uniform, GAN-based, self-adversarial, NSCaching) and with the proposed SNS method. For GAN-based method, we choose KBGAN as it is the only GAN-based sampling which has publicly available implementation (to the best of our knowledge).

In the experiments, we use five widely used benchmark KG datasets, *i.e.*, FB15K, FB15K-237, WN18, WN18RR, YAGO3-10 for link prediction task [BUGD+13, CW18, ZYSC19]. WN18, WN18RR were derived from WordNet, FB15K, FB15K-237 were extracted from Freebase and YAGO3-10 was extracted from YAGO knowledges. The Family dataset is a small KG, where the relations are family relationships among people [KD07, SADW19].
We also provide a biological KG dataset, namely FIGHT-HF-23R. FIGHT-HF is a biological

knowledge graph which describes named relations among several biological types such as proteins, genes, diseases, drugs [BLP+18]. The original graph contains 246,672 biological entities and 24,601,110 relations of 37 types. We extract a medium size dataset by considering only those relations which have more than 50 facts and less than 500K facts. As a result, the dataset, named FIGHT-HF-23R, contains 90,430 entities, 23 relation types and 948,298 triples. An example of extracted triple is (Cyclosporin A, drug_indication, Pterygium). A few statistics of this dataset are available in Appendix A.
The KG datasets, except the new FIGHT-HF-23R, come with train/valid/test splits. For FIGHT-HF-23R dataset, we split the dataset into train, validation, test triples. Unfortunately, we do not find any standard rule for splitting a KG dataset. We split the dataset into 90/5/5 for train/validation/test triples so that we have enough triples to train the model. We apply the split rule to relation label where positive triples (facts) with a specific relation are split with 90%/5%/5% ratio. This split ensures that we have all types of relations in all splits. We also check that no isolated entities exist in the training dataset as in this case the entities will have low quality embedding. The split gives train, test and valid triple sizes as in Table 3.1.

Table 3.1: The experimental KG datasets

KG datasets	**#Entity**	**#Relation type**	**#Facts**	**#Train**	**#Valid**	**#Test**
WN18RR	40,943	11	93,003	86,835	3,034	3,134
WN18	40,943	18	151,442	141,442	5,000	5,000
FB15K-237	14,541	237	310,116	272,115	17,535	20,466
FB15K	14,951	1,345	592,213	483,142	50,000	59,071
FIGHT-HF-23R	90,430	23	948,298	853,482	47,402	47,414
YAGO3-10	113,273	37	1,089,040	1,079,040	5,000	5,000
Family	3,007	12	28,356	25,517	1,410	1,429

Results and discussion

Our SNS sampling method is implemented using PyTorch library and the embedding programs were run on a 'NVIDIA A100-PCIE-40GB' GPU. We set training epoch number to 200, embedding dimension to 100, learning rate to 0.0001, margin value to 4.0 for all the experiments. For the embedding optimization task, we use the popular Adam optimizer[KB15] from the PyTorch library. We evaluate link prediction performance of embedding models with SNS sampling, and compare to other sampling methods on seven KG datasets. As for link prediction performance metrics, we compute Hit@z for $z = \{1, 3, 10\}$ and MRR score (defined in Section 1.3.1) of each method on each dataset. We re-scale the Hit@z score from the range 0-1 to 0-100 to facilitate the comparisons. Then, we perform more analyses including parameter sensitivity, performance at different epochs for the WN18RR dataset.

Prediction Performance:

We train the KG embedding models from scratch for each of the sampling methods, Random-uniform, KBGAN, NSCaching, SNS. For parameter setting, we follow the recommendations from the original paper ([ZYSC19] for NSCaching, [CW18] for KBGAN). The prediction performance metrics are tabulated in Table 3.2. Considering translational model(TransH), undoubtedly the proposed SNS sampling method shows the best prediction metrics among all the negative

sampling methods with respect to most of prediction metrics in most of the KG datasets. In WordNet KG datasets (WN18RR, WN18), NSCaching is the second best methods and the performance improved by 2-5% with SNS sampling considering Hit@k metrics when they are used with TransH. We find the best results of SNS sampling for FB15K dataset where the Hit@k scores are improved by 5-8% compared to second best NSCaching sampling method. For the other two datasets, SNS also remains best or second best with respect to almost all metrics. Good balance between exploration and exploitation could be the most suitable reason behind this success. When the sampling methods are used with DistMult model, the hit@10 and hit@3 scores drop in all datasets except FIGHT-HF-23R. Training the model for more epochs might improve the metrics. However, our intention is not to compare different prediction models, rather different sampling methods. Surprisingly, the performance metric on the new FIGHT-HF-23R KG dataset are poor for TransH with all sampling methods, but better for DistMult.

Table 3.2: Link prediction(LP) results: MRR, and Hit@z of different negative sampling(NS) methods with different embedding (KGE) methods on KG datasets. Random and Self-Adv represent the random-uniform and self-adversarial sampling methods respectively. The best and second best metrics for each sampling methods with each embedding methods are marked in bold and underline faces.

KGE model	NS method	WN18RR				WN18			
		MRR	hit@10	hit@3	hit@1	MRR	hit@10	hit@3	hit@1
TransH	Random	0.1520	32.27	23.72	0.11	0.3199	79.43	64.25	11.85
	NSCaching	0.1713	40.68	31.43	0.93	0.4171	88.65	74.05	17.48
	KBGAN	0.1708	40.08	29.35	0.10	0.4183	87.34	73.67	18.09
	Self-Adv	0.1709	41.18	31.16	0.89	0.4191	89.48	75.84	7.08
	SNS	**0.1852**	**43.04**	**33.82**	**1.80**	**0.448**	**91.47**	**79.30**	**19.28**
DistMult	Random	0.1918	32.16	23.88	14.22	0.3453	52.82	37.33	25.75
	NSCaching	0.2262	37.37	**29.11**	17.84	0.3772	56.85	**42.18**	29.04
	KBGAN	0.2285	33.42	27.23	17.34	0.3791	57.28	41.97	28.39
	Self-Adv	0.2279	36.23	26.34	17.05	**0.3940**	**58.43**	41.31	31.29
	SNS	**0.2333**	**37.83**	25.12	**18.49**	0.3931	56.46	42.13	**31.38**
		FB15K237				**FB15K**			
TransH	Random	0.1988	36.68	22.50	11.50	0.3115	52.88	36.68	19.58
	NSCaching	0.2476	40.39	26.59	**17.33**	0.3926	61.22	44.99	25.50
	KBGAN	0.2162	40.58	23.52	15.23	0.3228	53.67	38.34	20.52
	Self-Adv	0.2350	41.01	26.49	14.73	0.3883	61.09	44.05	25.74
	SNS	**0.2514**	**42.90**	**29.44**	15.18	**0.4360**	**66.72**	**51.52**	**30.58**
DistMult	Random	0.1918	31.82	20.71	12.96	0.2188	33.25	21.84	12.95
	NSCaching	0.2205	34.87	25.69	15.72	0.2327	39.89	27.41	16.19
	KBGAN	0.2282	36.23	**27.23**	13.02	0.2203	35.54	23.12	15.92
	Self-Adv	0.2400	37.09	25.82	17.34	0.2493	39.80	28.38	18.53
	SNS	**0.2471**	**38.18**	26.97	**17.80**	**0.2937**	**45.62**	**32.66**	**20.79**
		YAGO3-10				**Family**			
TransH	Random	0.0850	18.96	9.05	1.24	0.3164	88.77	48.67	3.43
	KBGAN	0.1467	26.08	16.03	8.34	0.3742	90.21	57.12	8.63
	Self-Adv	0.1443	26.46	**17.34**	**9.92**	0.3510	89.58	56.36	8.08
	NSCaching	0.1431	**26.76**	15.97	7.49	0.3434	90.34	55.04	4.65
	SNS	**0.1488**	26.72	16.23	8.87	**0.4146**	**92.97**	**62.39**	**13.12**
DistMult	Random	0.0533	10.59	5.44	2.35	0.5002	77.36	60.14	37.45
	KBGAN	0.0712	13.98	7.79	3.19	0.5120	78.01	60.75	38.01
	Self-Adv	0.0868	16.04	8.74	5.07	0.5122	78.59	60.95	**38.33**
	NSCaching	**0.0875**	14.22	**8.86**	**5.64**	**0.5274**	78.55	**61.34**	38.07
	SNS	0.0804	**16.72**	8.39	4.98	0.5190	**79.32**	61.02	37.98
		FIGHT-HF-23R							
TransH	Random	0.0200	3.95	1.86	0.63				
	KBGAN	0.0342	7.24	2.89	0.59				
	NSCaching	0.0294	6.96	1.97	0.66				
	SNS	**0.0410**	**8.92**	2.06	0.64				
DistMult	Random	0.1439	26.01	14.29	8.75				
	KBGAN	0.1681	28.25	18.64	**14.38**				
	NSCaching	0.1863	30.36	20.13	13.10				
	SNS	**0.1899**	**33.78**	**22.19**	12.63				

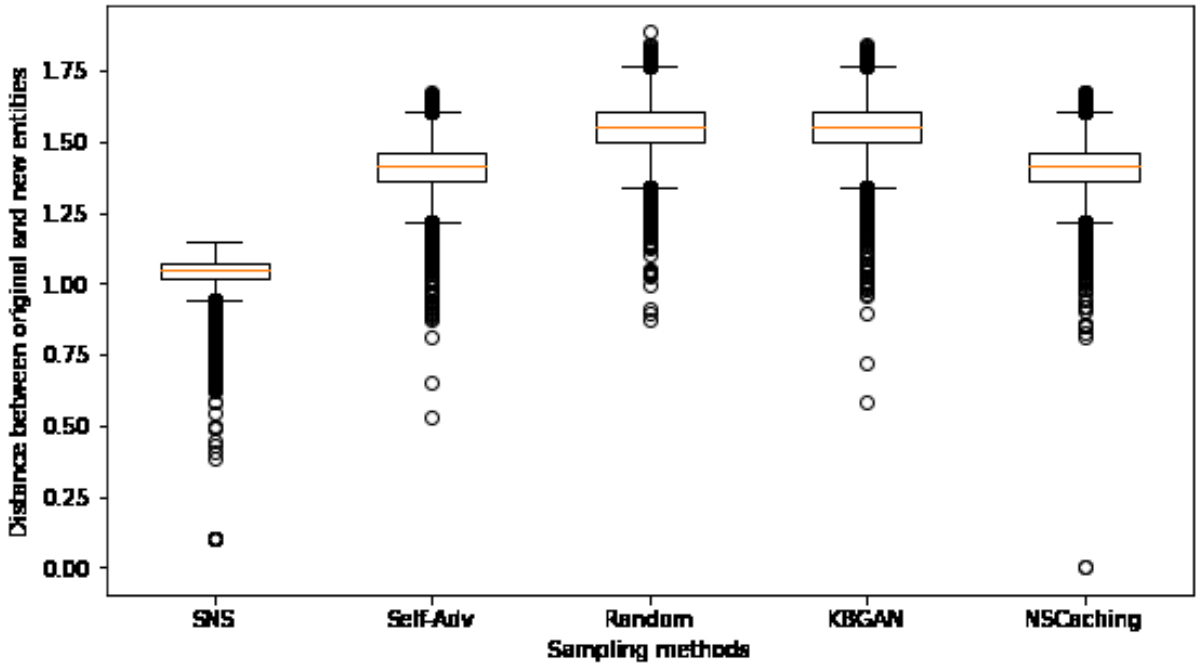


Figure 3.2: The distribution of distances scores (computed by Equation 3.3)) between the positive and their corresponding negative triples in the WN18RR dataset: The embeddings of entities and relations are learned by TransH with different sampling methods.

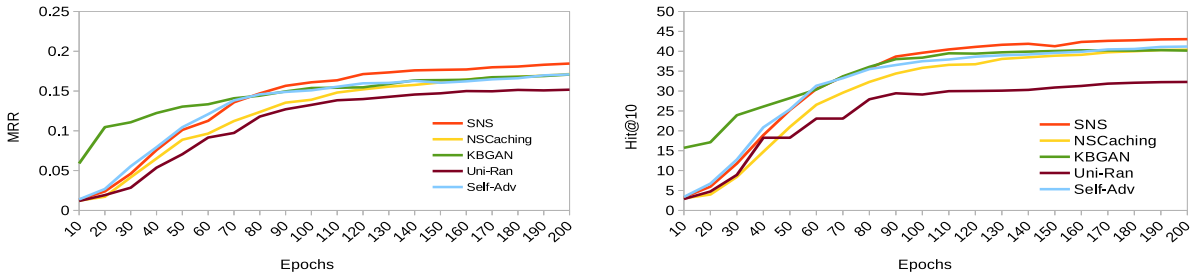
To go further in the exploration of the results, we provide a visualization of the closeness of the original and new entities in positives and their corresponding negatives in the second smallest KG dataset, WN18RR. Different negative sampling methods with TransH are used to learn embeddings. For the visualization, the negatives at the final epoch (*i.e.*, epoch number 200) are analyzed. In Figure 3.2, we show the distribution of distances between the original and the new entities for different sampling methods. The box for SNS sampling shows the minimum median distance. Overall, the minimum median illustrates that the selected entities for negative sample generation are closer to the original entities in SNS than other sampling methods. The selection of close entities in SNS leads to the generation of hard negative triples and improves the prediction performance in turn, as seen in Table 3.2.

In the following, we further analyze the performance of prediction by TransH with different sampling methods in the WN18RR KG dataset.

Change in prediction performance at different epochs:

To illustrate how the prediction performance changes with varying number of training epochs, we plot the MRR and Hit@10 scores of the TransH embedding model with different sampling methods from epoch 10 to 200 with a interval of 10 in Figure 3.3. At the initial point of epoch 10, the MRR score of SNS is nearly the same (around 0.02) as other sampling methods, except KBGAN which has the highest MRR score (around 0.06). As the training epoch number increases, the embeddings quality are improved and consequently the rise in MRR scores of all methods are seen. The MRR improving rate is seen higher for SNS method. At the epoch 70, the MRR score of SNS is the best. Though the improvement rate is not constant, the MRR

scores of the proposed SNS method remain highest among all the sampling methods in the following epochs. We see the worst MRR for 'uniform-random' method as the method does not learn to pick high-quality negative triples. These improvements in rank are reflected in Hit@10



(a) Epoch number vs MRR

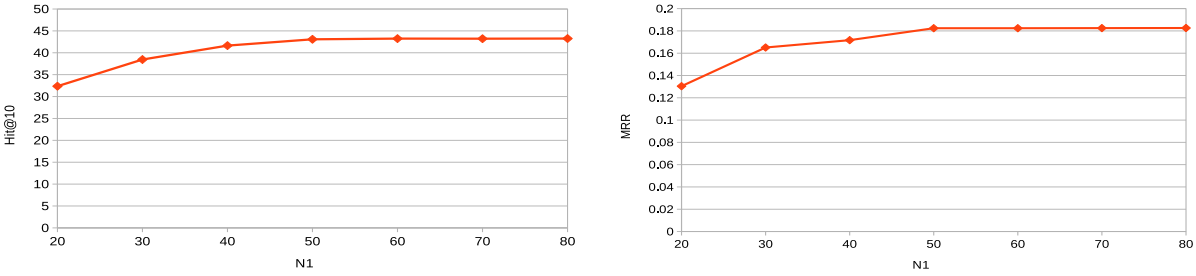
(b) Epoch number vs Hit@10

Figure 3.3: Prediction scores of TransH with different samplings in different epochs

scores curves where SNS has the highest Hit@10 scores in later half of training epochs. These improvements in performance suggest that our sampling method is able to provide better ranks of test triples than the state-of-the-art sampling methods.

Parameter sensitivity analysis:

SNS sampling method has two parameters, N_1 and N_2. To describe the changes in performance for different values of these parameters, we record Hit@10 and MRR scores for different values of N_1 with fixed $N_2 = 5$ which are plotted in Figure 3.4. As the value of N_1 increases, more initial negative triples are explored and the SNS sampling gets better exploration. As a consequence, the prediction performance improves as the value of N_1 increases from 20 to upper as seen from Figs. 3.4a, 3.4b. The prediction performance is nearly stable for $N_1 = 50$ and above. In the point $N_1 = 50$, SNS sampling has good exploration to sample sufficient number of high-quality negatives. And this could be the cause of performance stability for $N_1 \geq 50$. Again,



(a) N_1 vs Hit@10

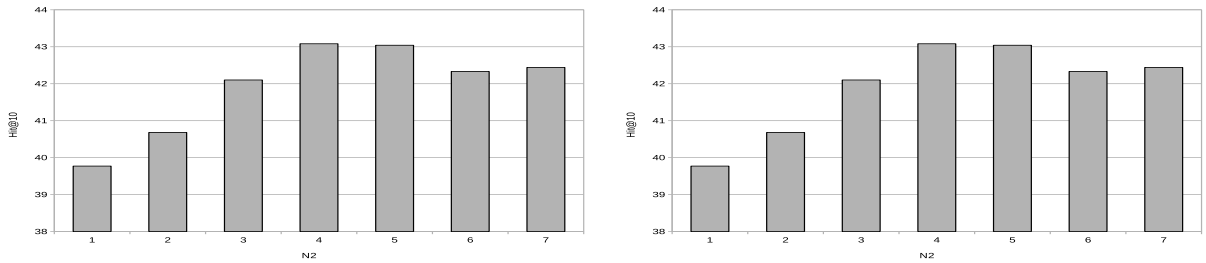
(b) N_1 vs MRR

Figure 3.4: Sensitivity of SNS to N_1, size of candidate negative set

to describe the sensitivity of the parameter N_2, we plot the performance metrics by varying N_2 among {1, 2, 3, 4, 5, 6, 7} with fixed N_1=50 in Figure 3.5. Figs. 3.5a and 3.5b show the change in Hit@10 and MRR for change in N_2. With setting N_2=1, SNS samples the top-most ranked negative(s). In this case, the chance of repeat sampling is high as SNS considers already known high-quality LRS negative(s) in addition to other candidate negatives. In case of high

repeat sampling, SNS suffers from low exploration and high exploitation effect leading to drops in MRR and Hit@10 metrics. With N_2=1, we see lowest prediction performance for SNS. As the value of N_2 increases, SNS performs better exploration and the best balance between exploration and exploitation is found for N_2=5 where the highest prediction metrics are recorded. However as the value of N_2 increases, the chance of sampling of known high-quality triple decreases (poor exploitation) and the chance of sampling less good-quality negative increases. As a result, the performance drops as seen in Figure 3.5 where both hit@10 and MRR drop for $N_2 > 5$.

Relation-wise performance on the WN18RR dataset:



(a) N_2 vs Hit@10 (b) N_2 vs MRR

Figure 3.5: Sensitivity of SNS N_2

To go beyond aggregated performance values, we aim to see the prediction performance on individual relations in the WN18RR dataset. We choose TransH as the embedding method because it is seen from Table 3.2 that TransH with SNS shows better prediction performance (in terms of hit@10 and hit@3) than DistMult with random sampling. We train TransH with different sampling methods and record the MRR and Hit@10 metrics for each relation type for each sampling method. The metrics of 8 relations (only those covering more than 1% of total facts) in the WN18RR dataset are shown in Table 3.3. We see that two relations (hypernym, derivationally_related_form) cover about 75% of the total facts in the training set. From the table, the prediction performance scores are higher for relations with a higher number of facts. We see the best MRR and hit@10 scores for the SNS method in almost 70% and 50% of the relations, respectively, which means that our sampling method ranks the triples better than other sampling methods. The performance scores of other sampling methods are high in some particular relations. For example, KBGAN shows remarkable performance in also_see, and verb_group relations while the performance is poor for the hypernym relation. NSCaching is the second-best option in most cases.

Memory and computational efficiency:

Undoubtedly, random-uniform is the simplest, fastest, memory efficient sampling method as it does not learn or store any parameter. GAN-based sampling makes the prediction model more complex, increases the number of training parameters, and makes the model harder to train due to use of reinforcement learning [ZYSC19]. As a consequence, KBGAN needs extra memory and computational cost to store and optimize the parameters. NSCaching stores a set of high-quality negative triples in each positive triple cache which makes it the worst regarding memory consumption. In addition, the method takes additional time to update the cache periodically. Our SNS sampling method does not increase the training parameter like GAN-based method. It memorizes only the least recently sampled negative triples which takes very small amount of

Table 3.3: Relation-wise performance in WN18RR dataset: MRR, MR and Hit@10 percentage. Number beside a relation denotes the percentage of facts covered by the relation in the test set. The best metrics for each relation are highlighted in bold face.

Relation (fact coverage)	SNS		KBGAN		NSCaching		Random	
	MRR	hit@10	MRR	hit@10	MRR	hit@10	MRR	hit@10
hypernym(39.92%)	**0.092**	**17.92**	0.04	8.18	0.0701	14.26	0.0357	11.28
derivationally_related_form(34.27%)	0.3971	95.44	**0.4051**	94.41	0.3899	**95.53**	0.3755	73.69
instance_hypernym(3.89%)	**0.0718**	**15.57**	0.0452	12.3	0.0439	9.84	0.0461	11.48
also_see(1.79%)	0.0978	26.79	**0.261**	**71.43**	0.0903	21.43	0.0766	25.0
member_meronym(8.07%)	**0.023**	**6.32**	0.0218	5.93	0.0134	3.56	0.012	4.74
synset_domain_topic_of(3.64%)	0.0412	8.77	**0.0465**	**14.04**	0.0375	7.89	0.0374	7.02
has_part(5.49%)	**0.0336**	**8.14**	0.0243	**8.14**	0.0197	4.65	0.019	3.49
verb_group(1.24%)	0.1998	69.23	**0.3571**	**97.44**	0.1497	48.72	0.1888	61.54

memory. Thus, intuitively SNS sampling is more memory efficient than GAN-based sampling and NSCaching. The method does not use complex learning method like GAN-based method or does not take extra cache updating time like NSCaching. It takes very small amount of time to update the LRS structure. We record the training time of each sampling method with the TransH embedding model. The data on training time is available in Figure A.3, Appendix A. We observed that the training time increases as the number of training samples increases, as expected. We found 7-40% and 3-14% improvement in training time for SNS when it is compared to KBGAN and NSCaching respectively.

3.3 The rule mining method

As discussed above (Section 3.2), the geometric embedding methods, sometimes known as neural methods [CDZ+21, ZCZ+21] in a knowledge graph setting, learn embeddings (numerical representations) of KGs. We propose a new rule mining method that utilizes a KG and its embeddings (numerical representations) to learn a set of rules. As our rule mining method uses both numerical and symbolic representations of KGs, we describe it as a neuro-symbolic method of mining rules.

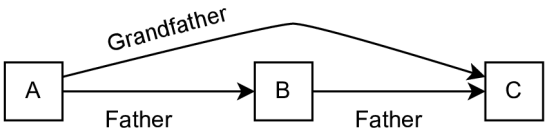
3.3.1 Rule mining from a KG and its embedding

To mine rules from a KG, we consider different path patterns occurring in the KG. Figure 3.6 illustrates four types of paths of length 2 in the 'Family' KG. The existing embedding-based rule mining methods only handle the first type, *i.e.*, closed-path (Figure 3.6a) and the rest three types (Figures 3.6b,3.6c,3.6d) are not handled. We focus on mining rules that consist of a head relation H and a sequence of body relations $B_1, B_2, ..., B_n$ in the following form

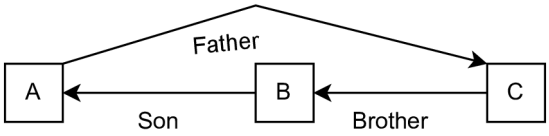
$$B_1(e_0, e_1) \wedge B_2(e_1, e_2) \wedge ..... \wedge B_n(e_{n-1}, e_n) \rightarrow H(e_0, e_n)$$

where e_i is an entity variable, H is the head/target relation from e_0 to e_n and $B_1, B_2, ..., B_n$ are the sequence of body relations. In the rule body, $B_i(e_{i-1}, e_i)$ is either a relation (*i.e.*, (e_{i-1}, r_i, e_i)) or its inverse (*i.e.*, (e_{i-1}, r_i^{-1}, e_i)).

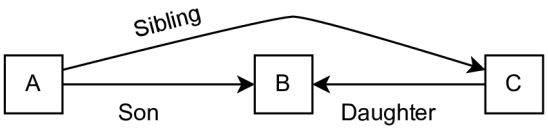
A simple path is a sequence of triples in which one common entity appears in between two consecutive triples with no cycle. We give more importance to shorter paths than longer ones. We compute the score of a path P as the average score over all triples composing the path,



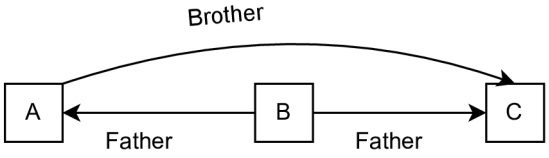
(a) start@head-closed@tail/ closed path



(b) Cycle



(c) start@head-closed@middle



(d) start@middle-closed@tail

Figure 3.6: Different types of paths for defining length-2 rules

penalized by the path length. We define the path scoring function PS as Equation 3.5.

$$PS(P) = \frac{1}{l^{1+\alpha}} \sum_{i=1}^{l} f(h_i, r_i, t_i) \tag{3.5}$$

where f is the embedding scoring function, l is the length of the path, α is the penalizing factor and $0 \leq \alpha \leq 1$. $\alpha = 0$ means we consider all paths equally whatever their length whereas larger values give more importance to shorter paths. We assume that a path with high score is a good path with respect to the learned embedding.

In the following we describe the procedure for rule mining which starts with graph generation, proceeds with triple sampling and ends with mining rules.

Bidirectional graph generation

To facilitate the search of the diverse paths to be used in in the rule bodies, we generate a bidirectional graph G from KG training triples. For each triple $(h, r, t) \in G$, we add a triple with inverted relation, *i.e.*, $G \leftarrow G \cup (t, r^{-1}, h)$ if there is no relation from t to h. In this way we make the graph bidirectional so that for every triple the tail is accessible from head as well as the head is accessible from tail. This transformation helps us to extract diverse path types (examples in Figures 3.6b-3.6c).
We schematize the procedure of mining rules for a specific relation r in Figure 3.7. We describe the procedure in the following sub-sections.

Rule set mining for a target relation r

Step 1. Sampling triples

Real-world KGs contain millions of triples for a relation and it is intractable to process such a huge number of triples for rule mining. We assume that mining rules from a portion of the triple set reflects well the rules one would achieve from the entire triple set for a relation. Hence, to mine rules, we randomly select a subset of triples of the target relation r.

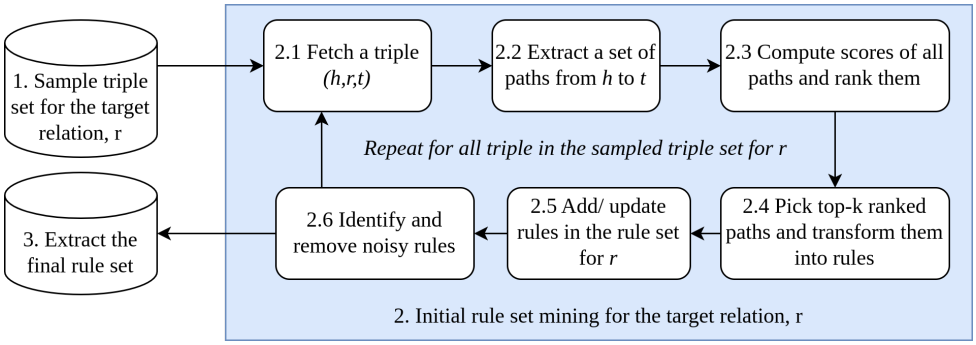


Figure 3.7: Rule mining for a target relation r

Step 2. Initial rule set mining

The initial rule set mining procedure for a relation r is illustrated in Figure 3.7 (shaded by blue color). In the following, we describe each step in details.

Step 2.1 (Fetch a triple): We consider the triples to arrive sequentially from the sampled triple set. The procedure starts with fetching the arrived triple (h,r,t). Note that only one triple is fetched at a time and the next triple is fetched after complete processing of the current triple.

Step 2.2 (Extracting paths): After fetching the triple, the relation r from h to t and its inverse(r^{-1}) (if it exists) are removed temporarily from the graph. Then a set of simple paths up to a length, MaxL from h to t are extracted. Here, MaxL is the predefined maximum body length, or simply maximum rule length. After extracting the path set, the removed relations between h and t are added again to the graph.

Step 2.3 (Computing path scores and ranks): The scores of all extracted paths are computed using the paths scoring function in Equation 3.5. Note that a path may contain the inverse relation of the form (h_i, r_i^{-1}, t_i). In this case, the triple score is computed as $f(t_i, r_i, h_i)$. This computation allows us to include all the cases in Figure 3.6.
The paths are then ranked in decreasing order of their scores. Better ranking of a path means better chance that the path will generate a good rule.

Step 2.4 (Defining rules from paths): We define the top-k ranked paths as strong supports for the current triple where k is a user-defined parameter. In this step, a path is transformed into a rule by replacing the entities of triples with variables. If the head and tail of a triple (h_i, r_i, t_i) are replaced with variables e_{i-1} and e_i then the triple is transformed into the body relation in the form $r_i(e_{i-1}, e_i)$. An inverse relation (h_i, r_i^{-1}, t_i) is transformed in the form $r_i(e_i, e_{i-1})$.

Step 2.5 (Update rule set): If a high-quality rule, *Rule*, does not exist in initial rule set, *Rule_ set* of a relation then it is added by initializing its structure as follows

$$Rule.count \leftarrow 1$$
$$Rule.energy \leftarrow 1.0 + \gamma$$

The 'count' variable of the rule represents the number of occurrences of the rule and the energy variable takes the full energy and an additional decay rate (γ) as initial energy. The rule is then

added to the initial rule set, *Rule_set i.e.*, $Rule\_set \leftarrow Rule\_set \cup Rule$. In case the rule *Rule* already exists in the *Rule_set*, then its count and energy values are updated as

$$Rule\_set[Rule].count \leftarrow Rule\_set[Rule].count + 1$$
$$Rule\_set[Rule].energy \leftarrow 1.0 + \gamma$$

Step 2.6 (Identify and remove noisy rules): The rules which are not useful in mining high-quality rules for a relation are identified as noisy and removed immediately. This removal reduces unnecessary memory and computing time consumption. In our method, a rule falls in one of three statutes.

- A 'potential' rule has positive energy and count equal to 1,
- a 'stable' rule has count above 1, and
- a noisy rule has count equal to 1 and negative energy.

A 'potential' rule may become 'stable' or 'noisy' later. Every time a triple is fetched from the sampled triple set, the energy of each potential rule, *Rule* from the *Rule_set* is decreased by the decay rate γ as

$$Rule\_set[Rule].energy \leftarrow Rule\_set[Rule].energy - \gamma$$

Usually the decay rate is $0 < \gamma < 1$. Small value of γ means potential rules will be alive for longer period before dying if they don't last for long time. Generally, the number of paths between two entities is high in a dense KG and low in a sparse KG. Hence, we argue that the possibility of the presence of noisy rules for a relation is high in a dense KG and low in a sparse KG. We recommend setting a high decay rate for a dense KG and a low decay rate for a sparse KG. All potential rules with negative energies are identified as noisy rules and immediately removed from memory.

Step 3. Final rule set extraction

The above rule mining processes (Steps 2.1-2.6) is repeated until all triples from the sampled triple set are processed. At the end of processing, the *Rule_set* contains 'stable' rules and a few 'potential' rules. The final rule set only contains the 'stable' ones.
Algorithm 1 describes the ranked path set extraction steps and Algorithm 2 describes the rule set mining steps based on the extracted path set.

3.3.2 Abduction for explainable link prediction

Once the rules are learned, they can be exploited to provide plausible explanations to support certain predictions. We propose a procedure inspired by abductive reasoning over the rules to infer the best explanation(s) for a prediction. Indeed abductive reasoning using a $body \rightarrow head$ rule allows us to infer (or at least assume) that body is satisfied whenever head is observed. Hence given a prediction as an observation, we look for rules whose head corresponds to the prediction. Each rule instantiation (using the bidirectional graph paths) constitutes then a plausible explanation for the prediction. Thus we have a way to further rank the top-k predictions based on their explainability.

[7] https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.simple_paths.all_simple_paths.html#networkx.algorithms.simple_paths.all_simple_paths

Algorithm 1: Extract_paths: Extracting paths for a given triple

Input: Bidirectional Knowledge graph(G), a triple (h,r,t), maximum path length (MaxL)
Output: Paths, Ranks

```
G←G-{h,r,t}                          // temporarily remove the relation from h to t
Path_Scores← ∅
Ranks← ∅
Paths=all_simple_paths(G, source=h, target=t, cutoff=MaxL) [7]   /* extract all simple paths from h to t with maximum path length, MaxL */
foreach P∈Paths do
    Score_P ← Score of the path, P                                // using equation 3.5
    Path_Scores ← Path_Scores ∪ Score_P
Ranks ← Ranks of paths based on Path_Scores   // in decreasing order of path scores
G←G∪{h,r,t}
```

Algorithm 2: Learn_rule: Learning rules for a relation

Input: Target relation (r), Bidirectional Knowledge graph (G), Triple set (T_r) with r, Decay rate (γ) , Sampling rate (β), Top-k, maximum path length (MaxL)
Output: *Rule_set*

```
T'_r ← Randomly sample (|T_r| * β) triples from T_r
foreach (h,r,t)∈ T'_r do
    Paths, Ranks =Extract_paths(G,(h,r,t), MaxL)        // Call Algorithm 1 to extract paths
        /* If a path exists then update its info, otherwise add the path to PathMetaInfo if it is ranked in top-k */
    foreach P∈Paths do
        Rule=Transform_path_to_rule(P) // replace entities with literals
        if Rule∈ Rule_set then
            Rule_set[Rule].Count← Rule_set[Rule].Count + 1 // increment counter
            Rule_set[Rule].Energy← 1 + γ // reset energy to full and additional γ
        else
            /* add a new rule                                                    */
            if Ranks[Rule]≤top-k then
                Rule.Count← 1
                Rule.Energy← 1 + γ
                Rule_set[Rule]← Rule_set ∪ Rule
    /* Identify the noisy rules                                                  */
    foreach Rule∈Rule_set do
        Rule.Energy←Rule.Energy - γ
        if Rule.Energy ≤ 0 & Rule.Count≤ 1 then
            Rule_set=Rule_set - Rule // remove the noisy rule
/* generate the final rule set by removing potential rules                       */
foreach Rule∈Rule_set do
    if Rule.Count≤ 1 then
        Rule_set=Rule_set - Rule // remove the potential rule
```

3.3.3 Evaluation of the rule mining method

Our rule mining method is implemented using PyTorch library and run on a Intel(R) Core(TM) i7-1.90GHz CPU machine. We use the embeddings learned by DistMult with SNS in rule mining task due to simplicity of DistMult. We compare the rule-set mined based on DistMult with SNS

and 'random' sampling. We set the sampling rate to 0.5, count threshold to 1, decay rate to 0.1, top-k to 5, and maximum rule length to 3 for experiments.

Experimental settings

To evaluate our rule mining method, we used the embeddings learned by DistMult with SNS and DistMult with random sampling. In the experiments, we use FB15K-237, WN18RR, Family KG datasets for the link prediction task. Table 3.1 presents characteristics of the used datasets in terms of number of entities, number of relations, number of facts and number of triples in the training, test and validation sets.

To assess the quality of learned rules, we rely on two common statistical measures standard confidence (SC) and head coverage (HC) [OWW21]. Both measures are defined based on the support of a rule in the following form.

$$Rule : \underbrace{r_1(e, e_1) \ \wedge \ r_2(e_1, e_2) \ \wedge \ .... \wedge \ r_n(e_{n-1}, e_n)}_{\text{body}} \longrightarrow \underbrace{r(e, e_n)}_{\text{head}}$$

The support of a rule is defined as the number of entity pairs satisfying the rule [OWW21].

$$Support(Rule) = \#(e, e_n) : body(Rule) \cap head(Rule) \tag{3.6}$$

SC is defined as the ratio of rule support and the number of entity pairs satisfying the body. The HC is the ratio of support and the number of entity pairs satisfying the head.

$$SC(Rule) = \frac{Support(Rule)}{\#(e, e') : body(Rule)} \tag{3.7}$$

$$HC(Rule) = \frac{Support(Rule)}{\#(e, e') : head(Rule)} \tag{3.8}$$

Configuration of rule-based explanation of predictions

As for explainable link prediction we need a metric to measure the potential of explanation of the rule set. We consider a positive test triple with a hit@10 score of 1 as a correct prediction. We aim to explain such predictions that are made by the embedding-based method. For each predicted triple, we identify the rules which make the prediction true through abduction-based instantiation (Section 3.3.2). The authors of the literature rule mining methods use precision metric to assess prediction performance [YYH+15a, HSGE+18]. Considering the incompleteness of a KG, we argue that precision could be a misleading metric as an unobserved triple is not necessarily false. Instead, we compute the recall score to evaluate the performance of our rule mining method. We define a true positive prediction as a test triple explained at least by one rule and the number of positives is the number of triples in the test set.

Result and discussion

The proposed rule mining method is implemented in Python with well-known PyTorch and run on a Intel(R) Core(TM) i7-1.90GHz CPU machine. We use the embeddings learned by an embedding method with SNS sampling strategy in rule mining task. To learn embeddings, we choose DistMult embedding method due to its simplicity. We compare the rule-set mined based on DistMult with SNS and 'random' sampling. We set the sampling rate to 0.5, count

threshold to 1, decay rate to 0.1, top-k to 5, and maximum rule length to 3 for experiments. We describe performance of our rule mining method in terms of prediction recall and quality metrics.

Noisy rule identification:

While the existing embedding-based rule mining methods identify noisy rules based on the quality metrics at the end of the mining process (depending on thresholds on quality metrics), our method is able to identify them early. Figure 3.8 shows the amount of identified noisy rules with different amounts of processed triples for 'brother' relation in Family KG. The early identification of noisy rules in our method reduces resource memory consumption as well as processing time.

Illustration of mined rules out-of closed-path pattern:

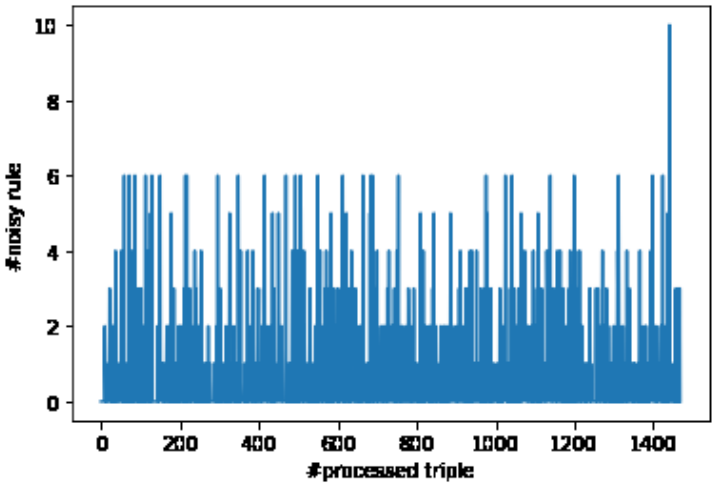


Figure 3.8: Identification of noisy rules for the "brother" relation in the Family KG

While existing embedding-based rule mining methods mine rules based on closed paths, our method is able to mine rules which do not follow closed-path pattern. Below the examples of two mined rules out-of closed-path pattern for 'aunt' relation in Family KG.

$$niece(A,B) \wedge niece(C,B) \wedge aunt(C,D) -> aunt(A,D)$$
$$nephew(B,A) \wedge brother(B,C) -> aunt(A,C)$$

Link prediction performance:

We compute the prediction performance of our rule mining method based on learned embeddings by our SNS and random sampling strategies. Table 3.4 tabulates the recall scores with their average, minimum and maximum numbers for Family, WN18RR and FB15K-237 KGs. Table 3.5 tabulates the quality metrics scores with their average, minimum and maximum numbers for the same datasets. Overall, our proposed rule mining method mines more rules when embeddings are learned using DistMult with SNS than DistMult with random sampling. But we see no significant difference between the performance scores of mined rule sets. We see that the average number of mined rules per relation is highest for the densest FB15K-237 and lowest

for the sparest WN18RR KGs. This is expected because the higher average node degree of a graph leads to generating more paths between pair of nodes and thus more rules to describe the relationship between the pair of nodes. The high number of rules for Family and FB15K-237 KGs give high recall scores in these two KGs. Looking at the minimum recall scores, we see the datasets except Family give a minimum score of 0.0. Investigating the relation-wise recall, we find that the quality of learned rules for a relation with low number of training set is very low and even no rule is generated for few such relations (e.g. similar_to relation in WN18RR, /baseball/.../baseball_team_stats/season in FB15K-237). This situation affects the performance scores as well as the overall rule set quality of a relation. In Table 3.5, we see the worst rule quality metrics for WN18RR. The average node degree of WN18RR is low which causes less number of paths between two entities to generate rules. This is most possible reason is that our method mines low-quality length-2 rules for WN18RR.

Table 3.4: Rule performance metrics: #Rules represents the total number of mined length-2 rules. The recall metric is given in the format (minimum score, average score, maximum score) which are computed over all relations in a KG. 'DistMult' is trained with two different sampling methods: SNS, random.

KGs	Random		SNS	
	#Rule	Recall (min.,avg.max.,std)	#Rules	Recall (min.,avg.max.,std)
Family	235	(0.569, 0.908, 1.0, 0.157)	242	(0.586, 0.910, 1.0, 0.150)
WN18RR	69	(0.0, 0.184, 0.487, 0.162)	76	(0.0, 0.185, 0.487, 0.164)
FB15K-237	8490	(0.0, 0.684, 1.0, , 0.262)	8559	(0.0, 0.689, 1.0, 0.258)

Table 3.5: Rule quality scores: HC and SC denote the head coverage and standard confidence metrics, respectively. The metrics are in given in the format (minimum score, average score, maximum score) which are computed over all relations in a KG. 'DistMult' is trained with two different sampling methods: SNS, random.

KGs	Random		SNS	
	HC(min., avg., max.)	SC(min., avg., max.)	HC(min., avg., max.)	SC(min., avg., max.)
Family	(0.135, 0.208, 0.241)	(0.325, 0.523, 0.674)	(0.119, 0.205, 0.254)	(0.285, 0.531, 0.695)
WN18RR	(0.0, 0.0337, 0.098)	(0.0, 0.0569, 0.179)	(0.0, 0.0311, 0.0869)	(0.0, 0.0689, 0.1433)
FB15K-237	(0.0, 0.176, 0.969)	(0.0, 0.196, 0.905)	(0.0, 0.175, 0.969)	(0.0, 0.199, 0.905)

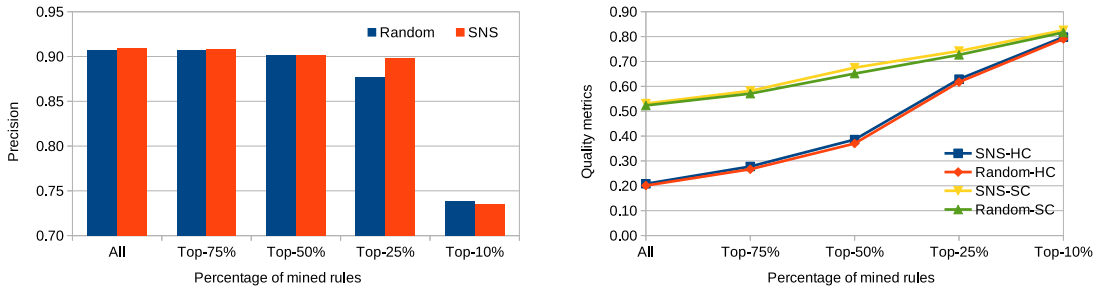
We try to investigate how the quality of embeddings affects the quality of mined rule set as well as the prediction performance in WN18RR datasets. We choose WN18RR datasets because the overall prediction metrics and rule quality metrics in this dataset is lowest. We select the rules of length-2 mined based on embedding learned by DistMult with SNS sampling. We see that the results are nearly consistent with the prediction results by embedding-based method in Table 3.6. We see that no rule is mined for the relation similar_to because the relation has insufficient number of triples in the training set.

To see how the rule mining performance changes with different percentage rules, we sample all, top-75%, top-50%, top-25%, and top-10% ranked length-2 rules for each relation in Family KG and illustrate them in Figure 3.9. The rules for a relation are ranked in decreasing order of their support scores. As low-ranked rules are not considered, triggering rules for a few triples will be missing. As a result, it is seen in Figure 3.9a that the recall scores for both sampling methods drop as the number of rules drops. In contrast, as low-quality rules are removed, the

Table 3.6: Relation-wise performance of mined length-2 rules in the WN18RR dataset: The embeddings are learned by DistMult with SNS sampling. The parameters for rule mining methods are set to default values except the maximum rule length is set to 2. Relations with recall scores above the average(0.184) are marked in boldface.

Relations	#Rule	Recall	HC	SC
hypernym	9	0.089	0.045	0.009
derivationally_related_form	12	0.224	0.087	0.019
instance_hypernym	5	0.197	0.017	0.028
also_see	7	0.214	0.048	0.040
member_meronym	6	0.099	0.013	0.016
synset_domain_topic_of	15	0.447	0.143	0.032
has_part	10	0.238	0.057	0.023
member_of_domain_usage	2	0.042	0.131	0.043
member_of_domain_region	2	0.00	0.142	0.015
verb_group	8	0.487	0.006	0.087
similar_to	0	0.000	0.00	0.00

average rule quality scores in Figure 3.9b improve.



(a) Prediction performance

(b) Rule quality

Figure 3.9: Change in prediction performance and quality with different percentages of mined rules of length-2 in the Family KG

In Table 3.7, we tabulate the top-5 ranked mined rules for the 'Father' relation in Family KG. Looking at the head coverage and standard confidence scores, we find that our method is able to compute better ranks for rules with higher quality.

Scalability of our rule mining method:

To assess the behaviour of our rule mining method for different KGs with different scales of connectivity, we illustrate the computational time for mining length-2 rules for the Family, WN18RR and FB15K-237 KGs in Figure 3.10. Among the three KGs, WN18RR is the most sparse (average node degree<5.0) whereas the other two are denser. As expected, we see that the average rule mining time grows as node degree increases because the number of paths between a pair of nodes is usually higher in a dense graph.

Table 3.7: Top-5 rules of length-3 for the 'father' relationship: ranks are computed in decreasing order of rule support values.

Rank	Rules	HC	SC
1	$husband(A,B) \wedge mother(B,C) -> father(A,C)$	0.765	0.882
2	$father(A,B) \wedge brother(B,C) -> father(A,C)$	0.599	0.407
3	$father(A,B) \wedge son(B,C) \wedge mother(C,D) -> father(A,D)$	0.550	0.298
4	$husband(A,B) \wedge mother(B,C) \wedge brother(C,D) -> father(A,D)$	0.546	0.386
5	$father(A,B) \wedge sister(B,C) -> father(A,C)$	0.524	0.420

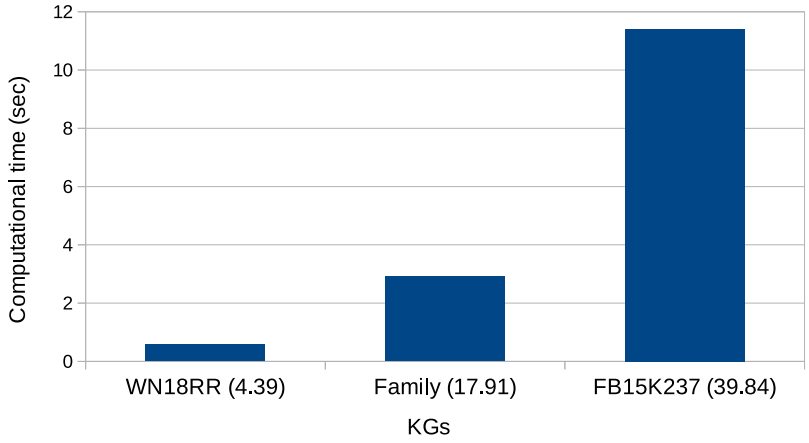


Figure 3.10: Graph complexity vs. computational time in different KGs for mining length-2 rules: The number on the horizontal axis in () denotes the average node degree. The computational time in the vertical axis is the average computational time over all relations in a KG.

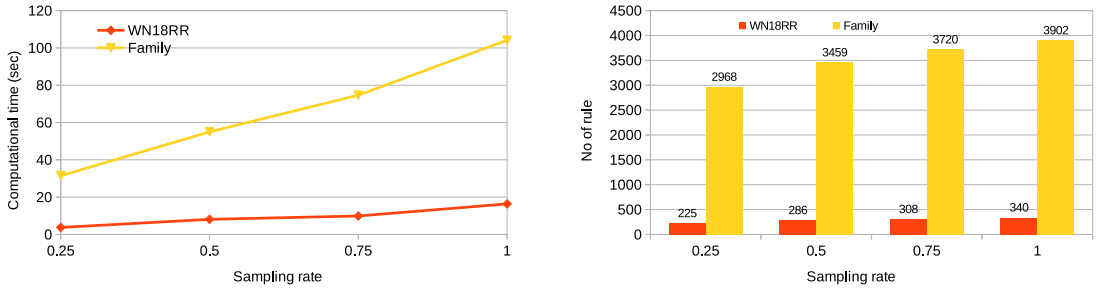
Parameter sensitivity of our rule mining method:

We assess parameter sensitivity of our rule mining method, we implement the method on the WN18RR and Family KGs to learn length-3 rules. In learning embedding, we choose our SNS sampling method. There are four parameters in our rule mining method and they are sampling rate, top-k paths, decay rate, and rule length.

Sampling rate: To assess how the rule mining method behaves for different sampling rates, we plot computational time and performance metrics (recall, HC, SC) in Figure 3.11. As expected, the computational time increases with increasing the sampling rate in Figure 3.11a for both datasets. The chance of generating more rules for a high sampling rate is high, and consequently the recall score is improved in Figure 3.11c. The rule mining time grows faster for the dense Family KG than the sparse WN18RR as the number of paths grows more quickly for the denser Family KG. The recall scores for WN18RR improved as we have more rules for higher sampling rate (Figure 3.11b). On the other hand, the recall scores for Family KG remain very close for different sampling rates. The most suitable reason is that the high-quality embeddings help the proposed rule mining method in extracting high-quality rules frequently, even with low sampling rate. More rules including low-quality rules are mined for higher sampling rates which causes the overall quality metrics to decrease. This situation is seen for both KGs. We see the sampling

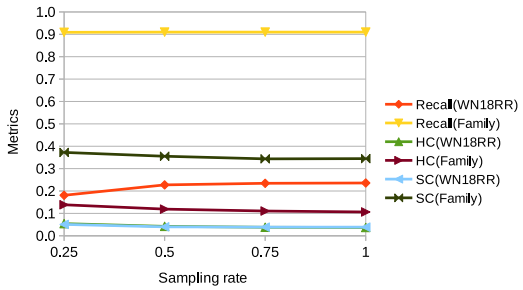
rate has impact on computational time as well as prediction and quality metrics. Undoubtedly, choosing a good sampling rate is a challenging but important task.

Top-k: We evaluate the total number of mined rules and the recall, HC, and SC of rules



(a) Sampling rate vs time

(b) Sampling rate vs #Rule



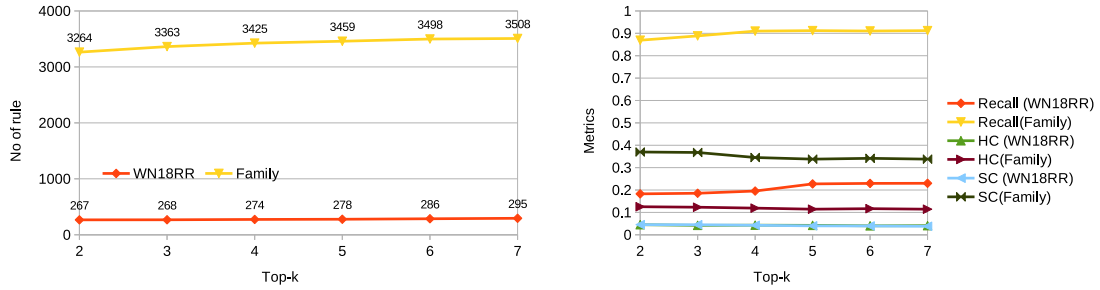
(c) Sampling rate vs metrics

Figure 3.11: Sensitivity to sampling rate in the WN18RR KG

for different top-k values to describe the sensitivity of the "top-k" parameter in our rule mining method. Figure 3.12a shows that the number of mined rules increases with increasing top-k for WN18RR and Family KGs. Consideration of more paths between a pair of nodes, including high-quality and low-quality paths, is the most possible reason behind this rising trend in the number of rules. This assumption is further supported by the quality metrics in Figure 3.12b where both HC and SC are slightly decreased as the top-k increases for both KGs.

Decay rate: To assess the sensitivity of our rule mining method to the "decay rate" parameter, we evaluate the total number of mined rules, recall, and quality metrics of our rule mining method for different decay rates (Figure 3.13). As the decay rate increases, the energy of an unobserved rule drops faster, and as a result, it lives for a low period in the system. For this reason, we see that the total number of mined rules in Figure 3.13a falls and consequently the recall scores dropped for higher decay rate. In contrast, the rule quality scores increase for both KGs as only high-quality rules are extracted for higher decay rates.

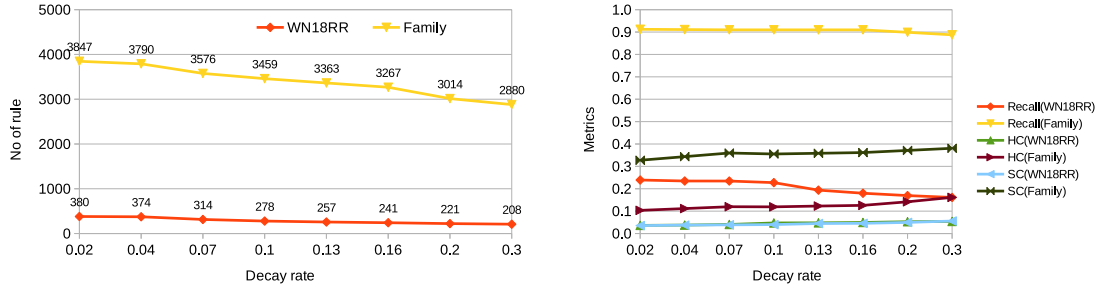
Maximum rule-length: Finally, we assess the sensitivity of the proposed rule mining method to "maximum rule length" parameter in terms of computational time, total number of rules, recall and quality metrics (Figure 3.14). The number of paths between a pair of nodes in a graph increases quickly as the maximum allowable path length increases. This results in rising computational time exponentially for mining rules as seen in Fig 3.14a. As the number of possible paths between a pair of nodes increases with increasing path length, the total number of rules



(a) Top-k vs #Rule

(b) Top-k vs metrics

Figure 3.12: Sensitivity to top-k: The default parameters setting are used (*i.e.*, max. length=3, decay rate=0.1, sampling rate=0.5) except top-k varies from 2 to 7 with an interval 1.



(a) Decay rate vs #rule

(b) Decay rate vs metrics

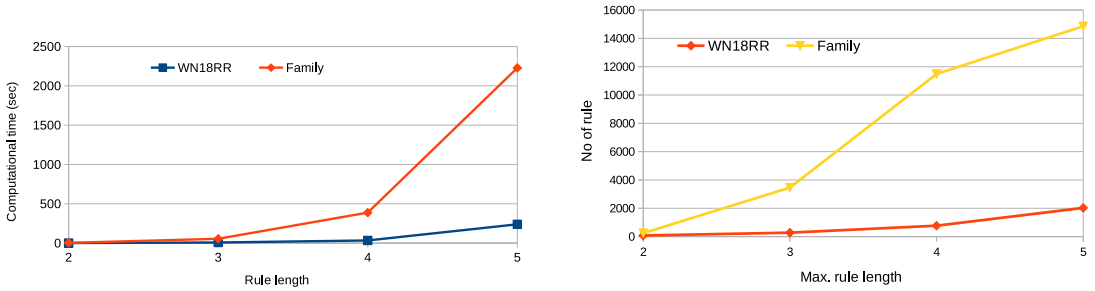
Figure 3.13: Sensitivity to decay rate: The default parameters setting are used (*i.e.*, max. rule length=3, sampling rate=0.5, top-k=5) except decay rates are from {0.02, 0.04, 0.07, 0.1, 0.13, 0.16, 0.2, 0.3}.

also increases for both KGs in Figure 3.14b. As more rules are mined, the recall slightly increases and quality metrics (HC and SC) drop in Figure 3.14c.

3.3.4 Illustration of explainable link predictions in FB15K-237

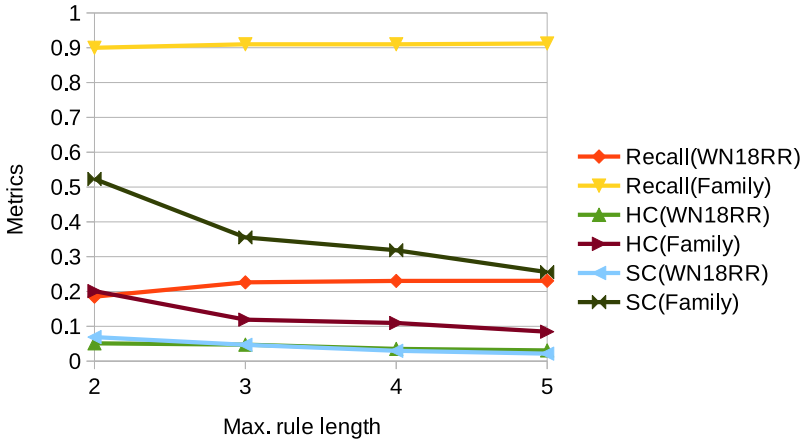
In this section, we take advantage from mined rules to explain the link prediction by the embedding-based method. We consider the test triples which have hit@10 score 1 as correctly predicted triples. For a correctly predicted triple, we may have multiple firing rules. We select the best rule based on length and quality metrics.

We illustrate the link prediction explanation based on the mined length-2 rules in FB15K-237 KG. Note that the entities in this KG come with Freebase identifier and Freebase KG is no more maintained. We map the Freebase identifier to Wikidata identifier using Freebase data dump [Goo13] and then the description of the entities are extracted from Wikidata using the Wikidata identifier. For illustration, we consider few triples with the '/film/film/language' relation from FB15K-237 KG which are correctly predicted by embedding-based method. The heads of these triples are film names and the tails are languages of films. For example, (Contact,../film/language, English) is interpreted as "The language of the film 'Contact' is 'English'". Fig 3.15 shows four such triples with their corresponding paths in the KG and in identified rules. The examples explain which paths help the target triple to be true. For instance, the prediction of the triple (Contact,../film/language,English) (in Fig 3.15a) could be explained as "'Jena Mal-



(a) Rule length vs time

(b) Rule length vs #rule



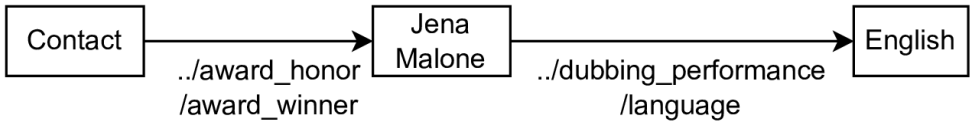
(c) Rule length vs metrics

Figure 3.14: Sensitivity to maximum rule length: The default parameters setting are used (*i.e.*, decay rate=0.1, sampling rate=0.5, top-k=5) except rule length varies from 2 to 5 with an interval 1.

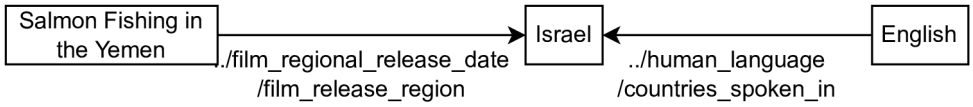
one' is awarded for the 'Contact' film and the dubbing language of 'Jena Malone' is English."

3.4 Conclusion

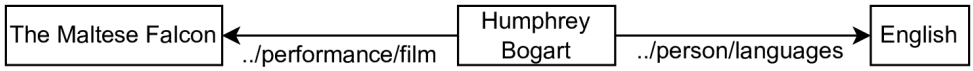
In this chapter, we presented our proposed SNS, a simple and efficient, negative sampling method for knowledge graph embedding. The method is general and can be plugged to any knowledge graph embedding method. The method is able to generate good negative triples and takes low computational time and memory while anticipating the 'vanishing-gradient' problem. Experimentally, we have evaluated our method on seven knowledge graph datasets for link prediction task and also have explored its parameter sensitivity. The empirical results show that the proposed SNS sampling brings consistent improvements in prediction performance.
We also presented our embedding-based rule mining method which is memory efficient and able to mine a diverse set of rules. We performed many experiments to evaluate the rule quality, but also the parameter sensitivity of the rule mining algorithm. We took advantage of the rule mining method to explain the link prediction in order to minimize the 'explainability' limitation of embedding-based link prediction methods to some extent. Although the quality and performance metrics of the learned rules are not satisfactory, the results show that our rule mining



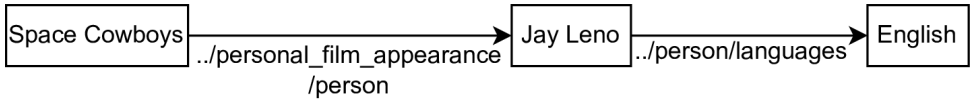
(a) (Contact,../film/language,English)



(b) (Salmon Fishing in the Yemen, ../film/language,English)



(c) (The Maltese Falcon, ../film/language, English)



(d) (Space Cowboys, ../film/language, English)

Figure 3.15: A few examples of correctly predicted triples in FB15K-237 KG by embedding-based method with their triggered paths identified by mined rules. The relation name in the KG is longer as it describes its hierarchy. Due to space constraints, we only include the lower part of the hierarchy and replace the higher parts with dots in describing a relation.

method is able to generate explanations for many predicted triples. Our rule mining method can be used in conjunction with any KG embedding method to explain its link predictions.
The poor performance of all studied embedding models on YAGO3-10 and FIGHT-HF-23R datasets leaves room to future exploration of other embedding models on these datasets. Ensembling several knowledge graph embedding methods could be a solution here (will be studied in a drug repurposing KG in Chapter 4). Implementing embedding models with SNS in a distributed environment to improve computational efficiency was another perspective of this work which will be presented in Chapter 5. As for rule mining method, designing a good strategy for sampling the triples before rule mining should improve the quality and performance metrics of the learned rules low sampling rate.

Chapter 4

Molecular-evaluated and explainable drug repurposing for COVID-19

Contents

4.1 Introduction

The development of a new drug requires a large sum of money (between 2 and 3 billion dollars) and a long time (over 13 years) [SBBW12]. The main source of this huge cost is the testing of a large number of drugs in preclinical stages, as well as the substantial percentage of randomised controlled trials (RCTs) that do not show clinical benefits or have toxicity risks [ZWT$^+$20]. Furthermore, it has a poor success rate, owing to incorrect drug target or response identification [Par19]. Within this context, exploiting approved and investigational drugs for new indications, a method called "drug repurposing", can remarkably reduce development cost and time as the clinical profiles of the studied drugs (pharmacokinetic, pharmacodynamic, and toxicity) are already known [MLW$^+$20].

Virtual screening approaches have shown strong impact in drug discovery and repurposing tasks which model the quality of a target protein-drug complex based on docking the drug against the 3D structure of the target protein [AMH$^+$20, ADK20]. However, they are known to let false positives through *i.e.*, drugs showing good docking results but not showing any activity in experiments [ADK20]. Analyzing drugs and disease targets from a larger perspective rather than just their structures could reduce false positive rate. In this context, a knowledge graph (KG)

is a useful tool which integrates different types of biological data from diverse sources. KGs are multi-relational graphs composed of entities (or nodes) representing several biological concepts (e.g. genes, proteins, drugs) and relations representing physical or biological associations. The integration of diverse data sources enables KG entities to be explored from a larger perspective for capturing complex relationships among diverse biological data and could help to minimize false results in predicted drugs. A KG is represented as a set of triples in the form *(head entity, relation, tail entity)*, also called facts [BUGD+13]. In KGs, the prediction of missing head or tail entities for a triple is known as link prediction [IAST21c, BUGD+13, CWZ+20]. Drug repurposing methods based on KGs have emerged as a prominent tool in recent years [HWC+21, ISM+20, ZHS+21, KS21, CRK+21].

As described in Chapter 1, KG embedding methods learn embeddings of entities and relations while preserving the inherent structure of a KG. The embeddings are then used in the link prediction task. Researchers studied many embedding methods in recent years, such as TransE [BUGD+13], TransH [WZFC14], TransD [JHX+15], DistMult [YYH+15b], RotatE [SDNT18b] (see Section 1.3.3 for details about these methods). Each of them has its own capabilities and limitations for learning embedding of different relations. As discussed in Chapter 1, TransE can not model symmetric, one-to-many and many-to-one relations, TransH can not model symmetric relations, DistMult can not model anti-symmetric and inverse relations. Therefore, these three embedding models are complementary considering different aspects of a KG relations.

In a knowledge graph setting, drug repurposing is formulated as a task of link prediction where the probability of a *Treat* relation is computed from an approved/investigational compound (head) to a disease (tail) *i.e.*, computing the probability of a *(Compound,Treat,Disease)* triple. Based on this formulation, there exist several drug repurposing approaches in the literature. Few of them are generic [MNN20, ZYX+22] and the rest are specific to certain diseases [GDX22, SDG+19, ZCJ+20, ISM+20].

The COVID-19 pandemic, caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2), has already costed the lives of almost 6 million people, and it still continues. To the best of our knowledge, no specific drug is available till now against the COVID-19. There exist few drug repurposing studies for the COVID-19 [ISM+20, ZHS+21, YLY+21, KS21, CRK+21]. Most of the approaches use traditional KG embedding methods for learning entity and relation embeddings in COVID-19 centric biological KGs like Drug Repurposing Knowledge Graph (DRKG) [ISM+20] and then use the embeddings in drug repurposing task. Evaluation of predictions against in-trial drugs for COVID-19 is the only way to asses the efficiency of their approaches.

In this chapter, we propose an integrated study for drug repurposing, evaluation and explanation for COVID-19 disease. The overall study workflow is illustrated in Figure 4.1. We start with collecting and cleaning a COVID-19 centric drug repurposing knowledge graph (DRKG). Then, we propose a novel approach to generate high-quality and compact ensemble embedding of the KG using three traditional embedding methods and the principal component analysis (PCA) method. The embeddings are used to train a deep neural network (DNN)-based prediction model. The trained model is used to predict the probability of all unobserved *(Compound, Treat, COVID-19)* triples where COVID-19 is represented with 27 proteins associated to SARS-CoV-2. The triples are then ranked in decreasing order of their probability values and top-100 compounds are predicted as potential compounds for COVID-19. The top-100 predictions are evaluated based on two groups of methods: (1) cross-matching with in-trial drugs for COVID-19 and (2) molecular evaluation based on compound and protein structures. Beside these evaluations, we learn high quality rules from DRKG and provide possible explanations for selected predictions based on our rule mining and explanation methods (described in the previous Chap-

ter 3).

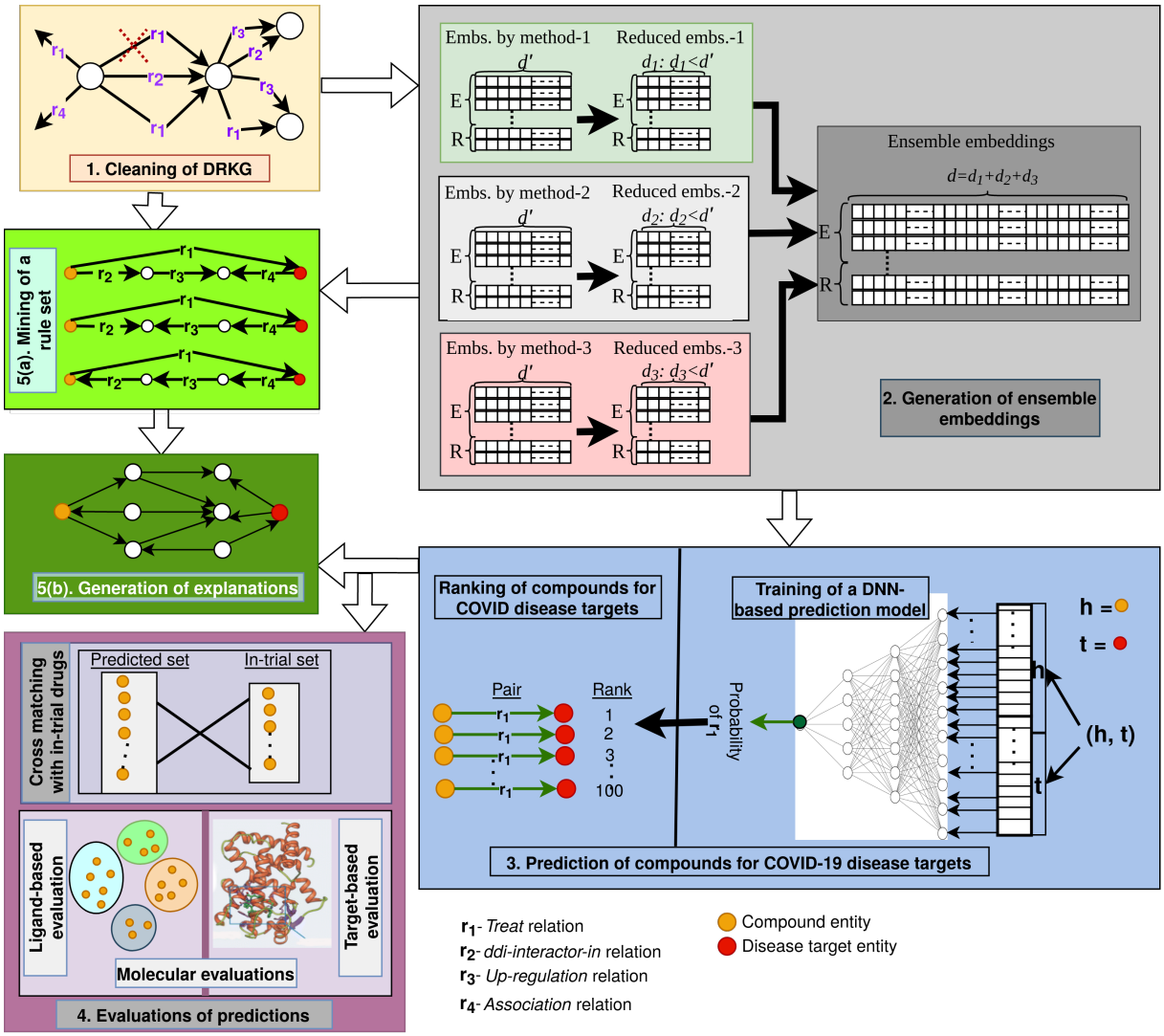


Figure 4.1: Overall study workflow; the major steps are numbered. Step 1 (yellow box); cleaning a COVID-19 centric drug repurposing knowledge graph (DRKG). Step 2 (gray box): learning high-quality and compact ensemble embeddings. Step 3 (blue box): predicting and ranking potential drugs for COVID-19 disease targets. Step 4 (purple box): evaluation of the top-100 compounds based on cross-matching with in-trial drugs (upper panel) and molecular evaluation of the compounds targeting SARS-CoV-2 nsp13 protein (lower panels). Step 5a (light-green box): learning from DRKG a set of explanation rules. Step 5b (dark green box): extracting explanatory paths instantiating the rules for given (Compounds, Disease) pairs of interest.

This study differs from state-of-the-art approaches on three major points. **Firstly**, other studies depend on a single KG embedding model ignoring the fact that a single model can only learn good embedding of certain relation types [RBF+21, WQW21]. On the contrary, we propose an ensemble approach combining multiple embeddings in order to embed different complementary aspects of KG relations. **Secondly**, existing approaches only asses their predictions against in-trial drugs and do not provide any molecular evaluation of their predictions. We provide

molecular evaluation of our predictions by comparison with known ligands of the COVID-19 targets. **Finally**, whereas KG-derived explanations of predictions are missing in most existing approaches, we provide rule-based explanations extracted from the KG and this contributes to improve the reliability of our predictions. We use the words 'compound', 'drug' and 'ligand' interchangeably throughout the chapter.

4.2 The drug repurposing methodology

Hereafter, we describe our methods using the step numbers introduced in Figure 4.1.

Step 1: Cleaning of DRKG

For drug repurposing of COVID-19, we employ DRKG, a COVID-19-centric knowledge graph, built with the support of the Amazon company [ISM+20]. DRKG was built from six biological knowledge bases (DrugBank, Hetionet, String, IntAct, DGIdb, GNBR) and three recent COVID-19 related publications [GTH+20, GJB+20, ZHS+20]. It contains biological entities including genes, chemical compounds, diseases, biological processes, side effects, and symptoms. In addition to SARS-CoV-2 related disease entities, DRKG also includes SARS, MERS related disease entities as SARS-CoV-2 has high similarity of sequence and infection mode with earlier MERS and SARS-CoV viruses. The details of DRKG building procedure can be found in the original article [ISM+20]. The COVID-19 disease is represented by different virus proteins which are involved in different stages of SARS-CoV-2 infection in hosts. There are 97,238 entities in the graph. Of these, 24,313 are compounds, 39,220 are genes, 5,103 are disease-related entities, and the rest are other types of entities. DRKG contains 107 relation names and 5,874,261 triples. Some of the relations are actually biologically equivalent, but DRKG considers them differently based on their sources. For example, 'GNBR:: T:: Compound:Disease', 'DRUGBANK:: Treats:: Compound:Disease' and 'Hetionet:: Ctd:: Compound:Disease' relations represent the treatment relation between compound and disease entities. The same situation happens for 'Hetionet:: GiG:: Gene:Gene', 'STRING:: BINDING:: Gene:Gene', 'INTACT:: DIRECT-INTERACTION:: Gene:Gene', 'INTACT:: PHYSICAL-ASSOCIATION:: Gene:Gene' relations representing the interaction between pair of genes. Because of occurring multiple equivalent relations in the original DRKG, we see redundancies in triples. For example, there are two triples *(Prednisolone, DRUGBANK::treats::Compound:Disease, Subacute thyroiditis)* (extracted from the Drugbank) and *(Prednisolone, GNBR:: treats:: Compound:Disease, Subacute thyroiditis)* (extracted from the GNBR) in the DRKG, but they both illustrate the same knowledge that Prednisolone (Drugbank identifier: DB00860) treats the disease Subacute thyroiditis (MESH identifier: D013968). We merge the equivalent relations and remove redundant triples to clean DRKG. We merge the three equivalent treat relations and the four interaction relations in the original DRKG into one 'Merged:: Treat:: Compound:Disease' and one 'Merged:: Interaction:: Gene: Gene' relation respectively in the cleaned DRKG. For simplicity, we use *Treat* to denote the "Merged:Treat:Compound:Disease" relation throughout the chapter. In addition, the cleaning of the original DRKG reduces 5 redundant relation names to one, thus removing 60, 644 redundant triples. The meta-graph of the cleaned DRKG is illustrated in Figure 4.2.

Step 2: Generation of ensemble embeddings

For training the embedding models for generating embeddings of entities and relations, DRKG triples are split by 90%-5%-5% to prepare positive train-test-valid sets. We apply the split ratio

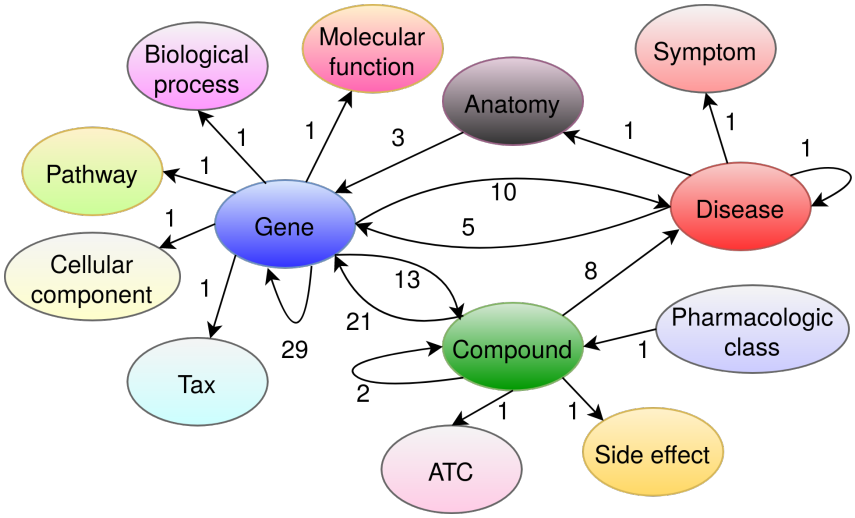


Figure 4.2: The cleaned DRKG metagraph: The number next to an arrow indicates the number of distinct relations between the corresponding entity types. For example, there are 21 distinct relations from compound to gene entities such as *Binds* , *Down-regulation*, *Up-regulation*.

to relation-wise triples to reduce imbalance among train, test and valid sets. In geometric KG embedding methods, the general objective is to give more plausibility scores to positive triples and less scores to negative triples. Let us say $\mathbb{E}$ is the set of entities, $\mathbb{R}$ is the set relations, $\mathbb{S}$ is the set of positive training triples, m is the batch size, S_m is the set of positive triples. KG embedding methods start with initializing embeddings of the entities and relations. The methods then fetch S_m and generate a set of negatives (S'_m) using a negative sampling method as negative triples are not readily available. We use our Simple Negative Sampling (SNS) [IAST21c] method to generate high-quality negatives. S_m and S'_m are combined to form a batch of triples. The embeddings are then improved by minimizing the loss between positives and respective negatives in a batch based on the following objective function (Equation 4.1).

$$\min_{\Theta} \sum_{\forall_{(h,r,t)\in S_m,(h',r,t')\in S'_m}} L(f(h,r,t), f(h',r,t')) + \lambda reg(\Theta) \tag{4.1}$$

Here, L is the loss function, f is the scoring function of an embedding method, λ is the margin, $reg(\Theta)$ is the regularization term, $(h,r,t) \in S_m$ is a positive triple and $(h',r,t') \in S'_m$ is the corresponding negative triple. We described some embedding methods with their scoring functions in Chapter 1, Section 1.3.3. We use the following pairwise loss function (Equation 4.2) due to its suitability in 'open-world' assumption.

$$L(f(h,r,t), f(h',r,t')) = \Big[\lambda - f(h,r,t) + f_r(h',r,t')\Big]_+ \tag{4.2}$$

Here, $[.]_+ = max(0,.)$ is the hinge function. The architecture of a geometric KG embedding method is described in Section 1.3.3.

There are many KG embedding methods in the literature and every method has its own strong and weak points in learning embeddings for relations with different properties. We follow the analysis conducted by Rossi *et al.* (2021) [RBF+21] on relation properties to select embedding method(s) to learn good quality embeddings of the DRKG. Based on triple statistics, we see that the DRKG contains relations with all properties. For example, the 'bioarx::HumGenHumGen::

Gene:Gene' relation is symmetric, the 'bioarx::DrugVirGen::Compound:Gene' is antisymmetric and the 'STRING ::REACTION::Gene:Gene' relation is inverse of the relation CATALYSIS::Gene:Gene. Therefore, embeddings of entities and relations in the cleaned DRKG are learned using the three complementary embedding methods (Figure 4.3). The quality of learned embeddings is evaluated through the link prediction task. The link prediction performance of an embedding method is defined based on ranks of positive test triples with two widely used metrics: Hit@z, and mean reciprocal rank (MRR) [BUGD$^+$13, WQW21, RM20]. The score range of both metrics is 0 to 1 and higher scores demonstrate better prediction performance. As suggested by most of the literature for link prediction in KGs, we consider $z \in \{1, 3, 10\}$. We re-scale the Hit@z scores from the range 0-1 to 0-100 to facilitate comparisons.

For training embedding methods, one of the important hyper-parameters is the latent space dimension size and finding an optimal value of this parameter requires many experiments which is obviously time consuming and computationally expensive. To avoid the difficulty in tuning this hyper-parameter, we initially set a high value (100) to this parameter. After learning embeddings, we apply the well-known principal component analysis (PCA) method to the embeddings learned by each method in order to reduce their dimension. For embeddings from each method, we keep only the dimensions with high variances by setting a variance ratio threshold of 1% of the total variation. In a recent study, Chen *et al.* (2022) [CHW22] argue that embeddings

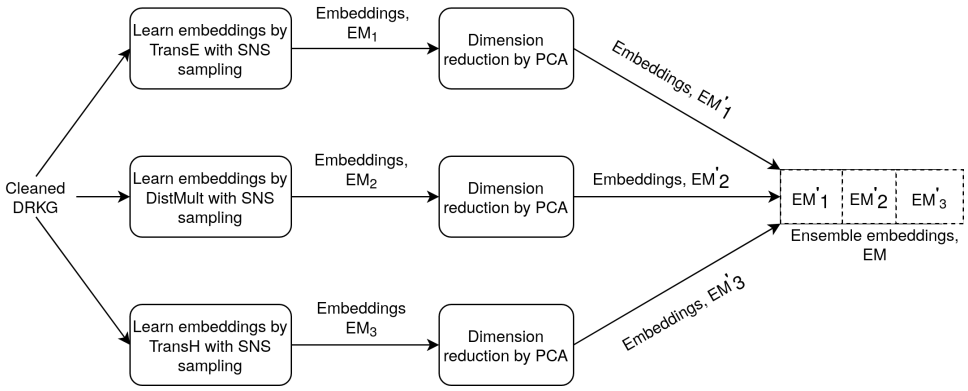


Figure 4.3: Ensemble embedding learning. Three embeddings of entities and relations are learned using three embedding methods: TransE, TransH, and DistMult. PCA method is then applied to each of these embeddings for dimensionality reduction. Finally, the three reduced embeddings of each entity or relation are concatenated to generate ensemble embeddings.

learned by different embedding methods for an object represent different latent features and can be ensembled to one new embedding of the object. As the three embedding methods (TransE, TransH, DistMult) consider different aspects of relations in DRKG, we ensemble the three reduced embeddings to a single new embedding of each entity and relation by simply concatenating them.

Step 3: Prediction of compounds for COVID-19 disease targets

Training of a DNN-based prediction model

For finding repurposable drugs for COVID-19 disease, we design a deep neural network (DNN)-based prediction model which computes the probability of a *Treat* relation for a given *(Compound, Disease)* entity pair. The prediction model is a 4-layer (148x74x37x1) multi-perceptron (MLP)

architecture with a Rectified Linear Unit (ReLU) activation function. For preparing the positive examples, we consider 52,216 *(Compound, Disease)* entity pairs with the *Treat* relation for the training set and 2,900 for the test set. For generating the negative examples, we use our SNS negative sampling method combined with entity-type constraints. For head corruption in a positive pair *(Compound/head, Disease/tail)*, we replace the head/Compound entity with other Compound entities from DRKG. As for tail corruption, we replace the tail/Disease entity by other disease entities from DRKG excluding COVID-19 entities. We also check that the negative examples do not correspond to any positive *(Compound, Disease)* pair in DRKG. Then, we apply SNS to select the best four negative pairs (two by corrupting the head, two by corrupting the tail) for each positive pair in training set. Similarly, we select one negative pair for each positive pair in the test set. As a result, the training set contains 52,216 positive and 208,864 negative *(Compound, Disease)* pairs, and the test set contains 2,900 positive and an equal number of negative *(Compound, Disease)* pairs. As input to the DNN, each *(Compound, Disease)* pair is represented by the concatenation of the ensemble embeddings of head/Compound and tail/Disease entities, and associated with its positive or negative label. We train the prediction model for a maximum of 4,000 epochs with early stopping and 30 epochs patience, Mean Squared Error (MSE) loss, Adam optimizer and a dropout of 0.15 for each layer. The trained model is validated with MSE on the test set.

Ranking of compounds for COVID-19 disease targets

The trained DNN-based prediction model is used to compute probability of an unobserved *(Compound, Treat, Disease)* triple or simply *(Compound, Disease)* pair as the model is trained for the *Treat* relation only. As explained earlier, we are interested to find top-ranked compounds which are expected to treat COVID-19 disease. There are 27 disease entities in the DRKG that represent COVID-19 disease (Appendix B, Table B.2) and none of them is linked to any Compound in DRKG. The authors of the DRKG dataset [ISM$^+$20] have integrated in DRKG a set of 8,103 FDA-approved or investigational drugs referenced in Drugbank, which have a MW greater than or equal to 250 Da, as candidates for drug repurposing. We use this set of candidate compounds in our *Treat* link prediction experiment. As a result, we have 8,103x27=218,781 *(Compound, Disease)* pairs to test. The probability values for all the pairs are computed using the trained prediction model and are ranked in decreasing order. Note that we have 27 *(Compound, Disease)* pairs for each compound and we consider only the best ranked pair among these 27 pairs for each compound. We re-ranked the compounds according to their best rank and obtained a list of 8,103 candidates for drug repurposing, with the indication of the best COVID-19 disease target for each of them. Following most of the aligned works [ISM$^+$20, KS21], the top-100 ranked compounds are proposed as potential compounds to treat COVID-19 disease.

Step 4: Evaluations of predictions

Cross-matching with in-trial drugs

In cross-matching evaluation, the top-100 predicted drugs are cross-matched with the set of in-trial drugs for COVID-19 disease. This evaluation describes the ability of our drug repurposing framework for finding already known drugs for COVID-19. We use the set of in-trial drugs provided by the DRKG authors in 2020 [ISM$^+$20] which consists of 31 compounds (provided in Appendix B, Table B.3). We check the drugs which exist in both sets. A high number of matches indicates better predictions. This is a simple and quick way of evaluating predictions. To the best

of our knowledge, this is the only method used in literature to evaluate a KG embedding-based drug repurposing approach.

Molecular evaluation

Known ligands for the COVID-19 nsp13 protein. As molecular evaluation is an expert and time-consuming task, we perform this type of evaluation only for one disease target: the SARS-CoV-2 nsp13 protein. In the top-100 ranked drugs (or ligands), we see 38 ligands for this disease target. For collecting known ligands for this disease target, we search both literature and PDB database. We collect 38 known ligands for nsp13 through literature screening [WLC20, VAKS22, PLMA$^+$22, Gur20, NBS$^+$21, PLHK22]. We also extract the ligands for nsp13 found in the relevant PDB entries. We obtain 71 entries for nsp13 structures complexed with a ligand in the PDB database and we reduce this number to 48 thanks to a redundancy threshold of Root-Mean-Square Deviation (RMSD) set to 2Å. The list of PDB identifiers (IDs) and corresponding ligands is provided in Appendix B,Table B.4. This table also contains the PDB ID for the structure of the nsp13 protein without any ligand (PDB:7NIO), also known as apo-nsp13. The 48 PDB ligands are designated hereafter with their capitalized PDB abbreviation. In total, our dataset of known nsp13-ligands contains 86 ligands (48 from PDB and 38 from literature). We use the known ligands as ground truths. The 2D structures of all ligands are collected from the PubChem database [KCC$^+$21] in SDF format. We use our 86 known and 38 predicted ligands for two types of molecular evaluation: ligand-based and target-based.

Ligand-based evaluation. The ligand-based evaluation is based on the concept of chemical structure similarity that says that similar ligands or compounds would bind to similar disease targets with almost the same binding affinity and express similar biological responses [AMH$^+$20]. This type of evaluation is quick and takes into account the polypharmacological properties of ligands [AMH$^+$20]. In ligand-based evaluation, we perform cluster analysis of known and predicted ligands to see how the ligands are grouped based on their similarity. In order to see whether our framework can find ligands that are comparable to drugs discovered by popular ligand-based virtual screening techniques, we conduct this evaluation. We use the ChemBioServer web application (https://chembioserver.vi-seem.eu/Dendrogram.php) to find clusters of ligands using their fingerprint (bit string that contains information on the structure) by the Hierarchical clustering method. We select the "Soergel distance" as the distance parameter, "Complete linkage" as the linkage parameter, and different cluster thresholds in {0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95} as we find no standard value in the literature for the threshold. A cluster with at least one predicted and at least one known ligand is an interesting cluster to us. Considering the number of interesting clusters, we find 0.9 as a good cluster threshold value. We then compute the Maximum Common Sub-structure (MCS) of ligands in each cluster to estimate the size of the chemical substructure shared by all ligands in a cluster. The higher the molecular weight (MW) of the MCS in a cluster, the higher the sharing degree between cluster members. As one small ligand could be just part of a large ligand, consideration of molecular weight similarity among ligands in a cluster is also an important aspect. It is expected that better predictions will have higher structure sharing and lower difference in MW when compared to known ligands.

Target-based evaluation. In target-based evaluation, the binding of ligands to target proteins is assessed from a 3D point of view using a computational approach. Molecular docking is a common computational approach to optimize the process of finding the most favorable 3D binding conformations of the ligand to the target protein [AMH$^+$20]. Molecular docking is performed using the GOLD software from Cambridge Crystallographic Data Centre. GOLD stands for Genetic Optimisation for Ligand Docking [JWG$^+$97]. As the average RMSD of collected 49 struc-

tures of the target is small (0.777), we use one structure, the apo-nsp13 structure (PDB: 7NIO) one, of the target which lacks any ligand. Based on the literature [NDY+21, VAKS22, WLC20], we select the ATP binding site of nsp13 as the binding pocket for molecular docking. The ATP binding site composition in terms of amino-acids (or residues) is as follows: GLU261, ASN265, GLY287, LYS288, SER289, HIS290, LYS320, LYS323, TYR324, ASP374, GLU375, GLN404, ARG442, ARG567. A single structure (the apo-nsp13 one) is used for a target because binding site variations across available PDB entries are weak (checked by RSMD calculation, Appendix B,Table B.4). For the ligands, we use the set of known and predicted (from the Top-100 predicted pairs) nsp13 ligands. We extract the 2D structures of these ligands in SDF format from PubChem [KCC+21] and use the Corina tool [GRS90] (purchased from Molecular Network, GmbH, Nürnberg, Germany ; https://mn-am.com/) to transform the 2D SDF format of ligands into 3D structures in MOL2 format. We compute ranks of the ligands in decreasing order of their GOLD docking scores and compare docking ranks with predicted ranks of different ligands. We then explore the interaction maps of few top-ranked predicted and known compounds using PLIP web tool [ALB+21] for deeper analysis and fair comparison.

Step 5a: Mining of a rule set

Using our neuro-symbolic method (described in Chapter 3, Section 3.3), we mine a set of rules from DRKG and use them for generating plausible explanation(s) for the predictions. A rule consists of a rule body (noted *body*(*Rule*)) and a rule head (noted *head*(*Rule*)) in the following form:

$$Rule : \underbrace{r_1(e_0, e_1) \ \wedge \ r_2(e_1, e_2) \ \wedge \ .... \wedge \ r_n(e_{n-1}, e_n)}_{\text{body}} \longrightarrow \underbrace{r(e_0, e_n)}_{\text{head}}$$

where $e_0, e_1, e_2, ..., e_{n-1}, e_n$ are entity variables, $r_1, r_2, ..., r_n$ are any type of relations from DRKG, and r is the head (*Treat*) relation. As we are interested only in the *Treat* relation between compound and disease targets, the *Treat* relation is the only relation accepted in the rule head and the (e_0, e_n) pair is constrained to be a *(Compound, Disease)* pair. Moreover, we limit the size of the rule body to 3 relations only. We use ensemble embeddings of entities and relations in the rule mining method. We use TransE scoring function as TransE shows the best link prediction performance scores. The quality of each rule is evaluated based on a statistical metric named head coverage (HC) [OWW21] (Equation 4.3). HC definition is based on the support of the rule *i.e.*, the number of instances of the rule [OWW21].

$$HC(Rule) = \frac{Support(Rule)}{\#(e_0, e_n) : head(Rule)} \tag{4.3}$$

The HC metric ranges from 0 to 1. A higher HC metric indicates a better rule.

Step 5b: Generation of explanations

The mined rules are used to find evidence or investigate drug action mechanisms. We use the mined rule set to generate explanations of predictions. In DRKG, COVID-19 disease entities are connected to host gene entities by only one relation 'Covid2_acc_host_gene::Disease:Gene' pertaining from a study [GJB+20] providing high-confidence interactions between SARS-CoV-2 proteins and human genes. For sake of consistency with the relations occurring in the rule set, we rename this relation as 'Hetionet::DaG::Disease:Gene'. For each predicted *(Compound, Treat, COVID-19)* triple (COVID-19 represents here one of the 27 SARS-CoV-2 proteins in DRKG), we first consider all the rules having as *head(Rule): Treat(Compound, COVID-19)*. Then, we

extract all the paths from DRKG that start with the corresponding compound and end with the corresponding COVID-19 protein. Finally, we select the paths which match any *body(Rule)* of the considered rules to generate the explanations.

4.3 Results

In this section, we present our experimental results. We first evaluate embedding models for link prediction task. Then we describe the drug prediction results and different evaluations of our predictions. Finally, we provide explanations for a few predictions.

4.3.1 The cleaned DRKG

We first clean the DRKG data [ISM+20] in order to reduce redundant information. The cleaned DRKG contains about 98,000 entities of 13 different types: gene, taxonomy (Tax), pathway, biological process, molecular function, cellular component, anatomy, disease, symptom, compound, pharmacologic class, ATC code, side effect (Figure 4.2). For our drug repurposing goal, 8103 compound entities, corresponding to approved or in-trial drugs are present in the cleaned DRKG graph. The COVID-19 disease is represented by 27 associated virus proteins (Appendix B, Table B.2). The KG contains 102 relation names and 5,813,617 triples. The *Treat* relation exists between compounds and diseases except for COVID-19 disease entities. We provide few statistical information on the cleaned DRKG in Appendix B, Figure B.1. We see that more than 65% triples come from two data sources: STRING (a protein interactions knowledge base) and IntAct (a molecular interaction knowledge base). As for the diversity of entity pairs in the triples collected from each data source (Appendix B, Table B.1), two data sources focus on only one type of pair: *(Gene, Gene)* for STRING and *(Compound, Gene)* for DGIdb. The five other data sources consider from 2 to 15 types of pairs (for Hetionet).

4.3.2 Generation of ensemble embeddings

We use complementary geometric KG embedding methods to generate embeddings of the cleaned DRKG. Beside embedding methods, negative sampling is an important component for learning KG embeddings. We use our recently published the Simple Negative Sampling (SNS) method [IAST21c] for sampling high-quality negative triples. We compared the link prediction performance of SNS with TransE to the frequently used 'uniform-random' sampling [BUGD+13] with TransE (Figure 4.4). As we see approximately 3% improvement in Hit@10 scores, we apply SNS in all embedding methods used here for learning embeddings. To confirm that the cleaning of DRKG does not affect embedding quality, we compare link prediction performance scores in cleaned DRKG to performance scores in original DRKG for three embedding methods: TransE, TransH, and DistMult (Table 4.1). We see that the performance scores for TransE and TransH

Table 4.1: Link prediction results in raw and cleaned DRKG

KGE method	**Original DRKG**				**Cleaned DRKG**			
	MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10
TransE	0.1670	7.93	20.61	31.66	0.1672	7.99	20.60	31.63
TransH	0.1602	7.22	20.09	30.50	0.1623	7.54	20.46	30.60
DistMult	0.1392	4.37	18.63	29.98	0.1417	5.17	19.01	30.03

do not differ significantly. To avoid the difficulty in tuning the hyper-parameter 'dimension size',

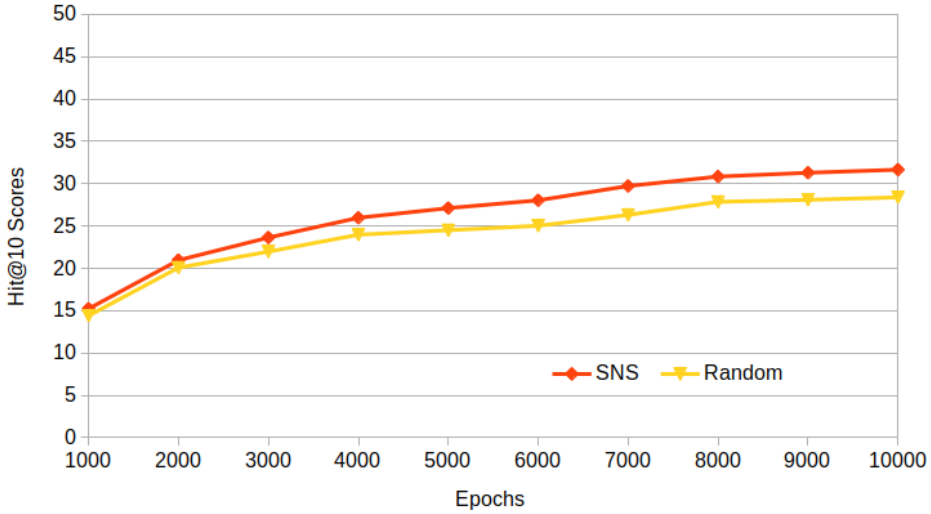


Figure 4.4: Hit@10 scores for SNS and Random sampling with TransE method for different epochs.

we first set it to 100 and then use principal component analysis (PCA) to reduce embeddings dimension for each of the three methods used. The embedding dimension size in TransE, DistMult, and TransH reduces to 26, 23, and 25 respectively. The concatenation of these embeddings gives the final ensemble embedding for each entity and relation of dimension size 74. The combination of three embedding methods helps to overcome the weak points of individual embedding methods. The ensemble embeddings of entities and relations are used in downstream drug prediction task.

4.3.3 Prediction of compounds for COVID-19 disease

We design a DNN-based prediction model to predict the probability that a *Treat* relation exists for a given *(Compound, Disease)* pair. The model is trained with a set of 261,080 training pairs and validated on 5,800 test pairs. We find a MSE of 0.037 on test set for the trained prediction model. The trained model is then used to compute the ranks of the candidate compounds based on their probability values with respect to COVID-19 disease entities. Interestingly, the top-100 predictions do not correspond to a unique but rather to several COVID-19 disease targets. Due to space constraints, we provide only the top-20 predicted compounds in Table 4.2 and the list of all ranked compounds in Appendix B, Table B.6. We also provide the best disease target for each compound. We see that the ranks of in-trial compounds for COVID-19 are improved noticeably compared to state-of-the-art approaches (see Table 4.2). Following aligned studies in literature [ISM+20, KS21], we consider the top-100 predicted compounds for performing evaluation.

4.3.4 Evaluations of predictions

To asses the efficiency of our proposed drug repurposing approach, it is important to evaluate our predictions against recommended / in-trial drugs. We perform two types of evaluation for our top-100 ranked compounds: cross-matching with in-trial drugs for COVID-19 and molecular evaluations (only for SARS-CoV-2-nsp13 disease target).

Cross-matching with in-trial drugs

When we cross-matched the top-100 compounds with the 31 in-trial compounds for COVID-19 (Appendix B, Table B.3), we see that the top-10 ranked compounds are actually in-trial compounds for COVID-19 in Table 4.2. We find 10/31 in-trial drugs in top-100 predictions in our approach which is remarkable compared to most of state-of-the-art aligned approaches. The

Table 4.2: Top-20 ranked drugs: The in-trial compounds are highlighted in blue texts, - represents unavailability of rank of a drug, * highlights in-trial drugs found only by our approach, the last five columns give the rank or prediction (✓) of compounds by different approaches.

Compound	Disease target	Our approach	Tex-Graph[KS21]	TransE-DRKG[ISM+20]	ENSIGN[LBEvH21]	PERM[CRK+21]
Dexamethasone	SARS-CoV-2-nsp6	1	1	4	✓	✓
Methylprednisolone	SARS-CoV-2-nsp6	2	6	16	✓	-
Ruxolitinib*	SARS-CoV-2-nsp13	3	-	-	-	-
Colchicine	SARS-CoV-2-nsp6	4	48	8	-	-
Thalidomide	SARS-CoV-2-nsp5_C145A	5	18	-	-	-
Chloroquine	SARS-CoV-2-nsp5_C145A	6	68	-	-	-
Azithromycin	SARS-CoV-2-nsp6	7	13	-	-	-
Losartan	SARS-CoV-2-nsp13	8	41	-	✓	-
Baricitinib*	SARS-CoV-2-nsp5_C145A	9	-	-	-	-
Hydroxychloroquine	SARS-CoV-2-nsp5_C145A	10	47	-	-	✓
Protirelin (DB09421)	SARS-CoV-2-nsp13	11	-	-	-	-
Telavancin (DB06402)	SARS-CoV-2-nsp13	12	-	-	-	-
Propiomazine (DB00777)	SARS-CoV-2-nsp14	13	-	-	-	-
Hydroxyzine (DB00557)	SARS-CoV-2-nsp13	14	-	-	-	-
Indinavir (DB00224)	SARS-CoV-2-nsp5_C145A	15	-	-	-	-
Nafcillin (DB00607)	SARS-CoV-2-nsp5_C145A	16	-	-	-	-
Bifonazole (DB04794)	SARS-CoV-2-nsp13	17	-	-	-	-
Obeticholic acid (DB05990)	SARS-CoV-2-nsp13	18	-	-	-	-
Meclizine (DB00737)	SARS-CoV-2-nsp13	19	-	-	-	-
Lovastatin (DB00227)	SARS-CoV-2-nsp6	20	-	-	-	-

highest number of compounds (40) in top-100 corresponds to the SARS-CoV-2-nsp13 target. Among these 40 compounds 2 are in-trial and 38 are new. We choose to focus our molecular evaluations on these new compounds.

Molecular evaluation of compounds for SARS-CoV-2-nsp13

For molecular evaluations, we compare our 38 predicted compounds with 86 compounds known to bind the SARS-CoV-2-nsp13 target (38 from literature and 48 from the PDB database [BWF+00]). We provide molecular evaluations based on either ligand or target structures and these are named hereafter ligand-based and target-based evaluations.

Ligand-based evaluation. This evaluation consists of clustering the predicted and known ligands based on their structural similarity. We find a total of thirteen clusters. Among these clusters, ten contain both predicted and known ligands and their content is listed in Table 4.3. We also provide molecular weight (MW) of the maximal common sub-structure (MCS) of ligands in each cluster in the table. The first cluster contains 10 ligands among which 6 are known from the literature and 4 are new predicted compounds. Moreover, this cluster displays the highest MCS, thus revealing a good molecular similarity between all these compounds. Based on the MCS values that display an important decrease between cluster 6 and 7, one could retain all the new compounds from clusters 1 to 6 (*i.e.*, 18 compounds) as potential interesting ligands for the nsp13 target. Interestingly, this set of 18 compounds includes Fosinopril that will also appear in the target-based evaluation.

Target-based evaluation. This evaluation consists of performing molecular docking of the 38 predicted and 86 known ligands in the active site of the nsp13 structure using the GOLD software. We provide complete docking results in Appendix B, Table B.7. Table 4.4 lists the top-20 ligands with respect to molecular docking. Interestingly, 4/38 predicted ligands are present in

Table 4.3: Clusters of top-100 predicted and known (from literature and PDB) ligands: The number in ( ) gives integer molecular weight. The clusters are numbered in decreasing order of their maximum common substructure (MCS) weights. The known ligands from the PDB database are designated by their PDB abbreviation. The corresponding full-names are provided in Appendix B, Table B.4.

No	**Predicted ligands**	**Known ligands**	**MW** (g/mol) for MCS
1	Fosinopril(563), Griseofulvin(352), Telavancin(1755), Ridaforolimus(990)	Simeprevir(749), Dihydroergotamine(583), Paritaprevir(765), Ergoloid(611), Grazoprevir(766), Ergotamine(581)	120
2	Protirelin(362), Teriparatide(4117), Tiagabine(375)	NUA(193), HR5(207), VWM(187)	105
3	Binimetinib(441), Niflumic_acid(282), Moxifloxacin(401)	Picrasidine N(490), Irinotecan(586), Netupitant(578), Lumacaftor(452), Bananin(327), Picrasidine M(490), Nilotinib(529), Zelboraf(489)	98
4	Mesoridazine(386), Perphenazine(403), Oxcarbazepine(252), Tizanidine(253)	SSYA10-001(308), NY7(194), VVD(197), VW7(204), S7G(190), UVA(185), N0E(241), NZG(197), JHJ(243), LJA(193), EJQ(222), VXD(198)	93
5	Meclizine(390), Flunarizine(404), Remoxipride(371), Hydroxyzine(374)	Lifitegrast(615), Emend(534), UR7(203), JOV(227), VWD(200), NX7(211), VWA(153), UQS(191), UXG(239), VW4(199)	92
6	Darifenacin(426), Tetrabenazine(317), Oxybutynin(357), Ritodrine(287), Oxymorphone(301), Naloxone(327), Hydrocodone(299), Tofisopam(382)	Tubocurarine(609), Cepharanthine(606), Differin(412), Isorhoeadine(383), Epiexcelsin(414), Enjuvia(350), Homovanillic acid(182), K2P(206), VXG(233), VVY(205), STV(243), VW1(190)	89
7	Lamotrigine(256)	K34(152), JG4(150)	67
8	Carisoprodol(260)	Clavulanic acid(199), Acetylcysteine(163), VWV(221)	59
9	Macitentan(588), Famotidine(337), Eprosartan(424)	Cefoperazone(645), Cefpiramide(612), Risperdal(410), Cordycepin(251), Pritelivir(402), Dpnh(665), VX4(224)	50
10	Risedronic acid(283), Cinchocaine(343), Bifonazole(310)	RYM(233), NYV(189), VWJ(175), O2A(174), UVJ(203), MUK(199), S7J(191), UJK(203), VWG(188)	50

this list, with docking scores greater than 70. In particular, Fosinopril stands out and is ranked at the second position, with a score (78.86) very similar to the first-ranked ligand Diosmine (79.04), identified as nsp13 ligand by White *et al.* [WLC20]. The three other predicted ligands: Macitentan, Eprosartan, and Dinoprostone are ranked at position 12 to 14, with scores ranging from 70.76 to 71.76 despite of their varying MW, thus excluding an effect of their size on the

number of interactions with the target. Table 4.4 also displays the best disease target and

Table 4.4: List of Top-20 best docked ligands ranked according to decreasing Gold Score. * means that the ligand is not present in DRKG. The results for the ligands from our predictions are highlighted in light cyan colour. The prediction values in DRKG are also indicated with the best disease target and corresponding probability score. When the best target is not nsp13, the second best target and its probability score are indicated only if it is nsp13.

Docking rank	Ligand name	Gold score	MW (g/mol)	Best predicted target	Predicted probability score	Predicted rank	Reference
1	Diosmin	79.04	608.5	nsp9	0.169	6091	White *et al.* (2020) [WLC20]
2	Fosinopril	78.86	563.7	nsp13	0.964	29	This study
3	Nilotinib	76.55	529.5	nps6/nsp13	0.860/0.796	101	White *et al.* (2020) [WLC20]
4	Chromone-4c*	76.17	417.4	NA	NA	NA	Perez-Lemus *et al.* (2022) [PLMA+22]
5	Dpnh	76.04	665.4	nsp9	0.146	6361	White *et al.* (2020) [WLC20]
6	Cromolyn	75.7	468.4	nsp13	0.619	1751	White *et al.* (2020) [WLC20]
7	Picrasidine_N*	74.55	490.5	NA	NA	NA	Vivel-Ananth *et al.* (2022) [VAKS22]
8	Picrasidine_M*	74.52	490.5	NA	NA	NA	Vivel-Ananth *et al.* (2022) [VAKS22]
9	Dihydroergotamine	74.13	583.7	nsp13	0.796	475	White *et al.* (2020) [WLC20]
10	Ergotamine	72.83	581.7	nsp13	0.805	361	White *et al.* (2020) [WLC20]
11	Simeprevir	72.68	749.9	nsp6/nsp13	0.537/0.139	2076	Gurung (2020) [Gur20]
12	Macitentan	71.76	588.3	nsp13	0.95	49	This study
13	Eprosartan	71.33	424.5	nsp13	0.878	71	This study
14	Dinoprostone	70.76	352.5	nsp13	0.959	38	This study
15	Cefoperazone	70.24	645.7	nsp13	0.599	1816	White *et al.* (2020) [WLC20]
16	Scutellarin*	70.19	462.4	NA	NA	NA	Gurung (2020) [Gur20]
17	Irinotecan	70.11	586.7	nsp6/nsp13	0.824/0.796	249	White *et al.* (2020) [WLC20]
18	Paritaprevir	69.61	765.9	orf8	0.099	6990	Gurung (2020) [Gur20]
19	Risperdal	68.76	410.5	nsp6/nsp13	0.807/0.807	349	White *et al.* (2020) [WLC20]
20	Ergoloid	68.59	611.7	nsp13	0.584	1888	White *et al.* (2020) [WLC20]

probability score of the triples formed between the ligands and COVID-19 disease targets, except for four of them (marked with an asterisk) which are not present in DRKG. When the best disease target is not nsp13, the information for the second best target is provided only if it is nsp13. Surprisingly the best docked ligand Diosmine does not display nsp13 as best or second-best target.

We further analyzed our docking results by exploring the interaction maps of our four best-ranked compounds and comparing them with Diosmine (known ligand with the best Gold score), Ergotamine and Risperdal (known ligands with the two highest probability scores with nsp13 in DRKG) to check their binding similarities. Figure 4.5 shows the superposition of the ligands (4 predicted versus 3 known) extracted from the corresponding docking best poses and Table 4.5 compares the list of nsp13 residues concerned by interactions with all these ligands.

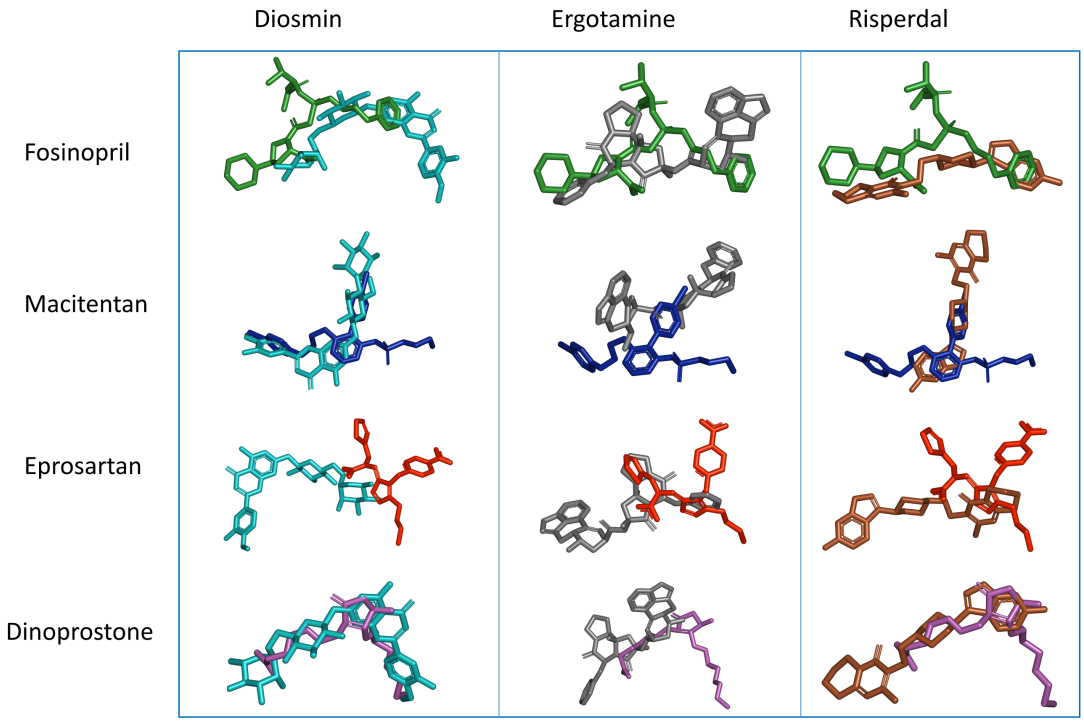


Figure 4.5: Superposition of binding poses with nsp13 target for Fosinopril (in green), Macitentan (in dark blue), Eprosartan (in red) and Dinoprostone (in magenta) against Diosmin (in cyan), Ergotaminin (in grey) and Risperdal (in brown). Ligands along the vertical axis correspond to the predicted ones, while those along the horizontal axis correspond to the known ones.

Table 4.5: List of amino-acid residues in nsp13 structure that interact with the listed ligands (predicted ligands are highlighted in light cyan, the others correspond to known ligands). Ligands are ranked as in Table 4.4. Residue label and position are in bold when they correspond to residues delineating the ATP binding site. All the interaction maps are in Appendix B, Table B.5.

Ligands	GLY285	**GLY287**	**LYS288**	**SER289**	**HIS290**	**LYS320**	**GLU375**	**ARG442**	GLY538	GLU540
Diosmin	X	X	X	X	X	X		X	X	X
Fosinopril	X	X	X	X		X	X	X		X
Ergotamine	X		X	X	X	X	X	X	X	
Eprosartan			X	X		X				
Macitentan					X	X		X		
Dinoprostone	X	X	X	X	X	X		X		
Risperdal			X	X	X	X	X		X	X

These data show that the predicted compounds share with the known ligands several structural elements and two of them (Fosinopril and Dinoprostone) interact with 6/7 amino-acid residues from the nsp13 active site.

4.3.5 Explanations of predictions

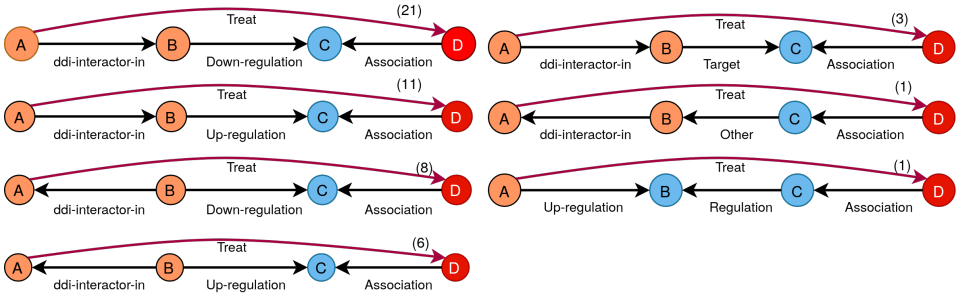
For better interpretability and insights about a *(Compound, Treat, Disease)* triple or simply *(Compound, Disease)* pair, we can explore the paths satisfying the learned rule set for the *Treat* relation. As we are interested only in the *Treat* relation between compound and disease target,

we learn rules for this relation only. The rule set contains 662 high-quality rules (see Chapter 3 for definition). The explanations for a prediction is generated by instantiating the rules in the DRKG (described in Materials and Methods section). Here, we illustrate (Figure 4.6) the paths satisfying the rule set for the best predicted *(Compound, Disease target)* pair in terms of docking rank: (Fosinopril, nsp13). We find 51 paths from the predicted Fosinopril compound to the nsp13 entity satisfying seven different rules from the rule set. We schematize the rules which instantiate at least one path in Figure 4.6a. We see that the first relation in rule body is the interaction between two compounds except in the second last rule and as second relation, one finds different types of regulation between compounds and genes. Among the paths, more than 40% satisfy the first single rule. In Figure 4.6b, we provide a graph showing the 51 paths from the predicted Fosinopril compound to the target SARS-CoV-2-nsp13 entity. We see that TLE family member 1, transcriptional corepressor (TLE1) https://www.ncbi.nlm.nih.gov/gene/7088 is an important nsp13-associated gene and is up-regulated or down-regulated by 15 compounds which interact with the Fosinopril compound. This gene is located in cytosol and nucleoplasm, and enables protein binding and transcription corepressor activities [LDP+96]. PRKACA (https://www.ncbi.nlm.nih.gov/gene/5566) and CENPF (https://www.ncbi.nlm.nih.gov/gene/1063) are two other genes which have interaction with nearly 10 compounds from DRKG.

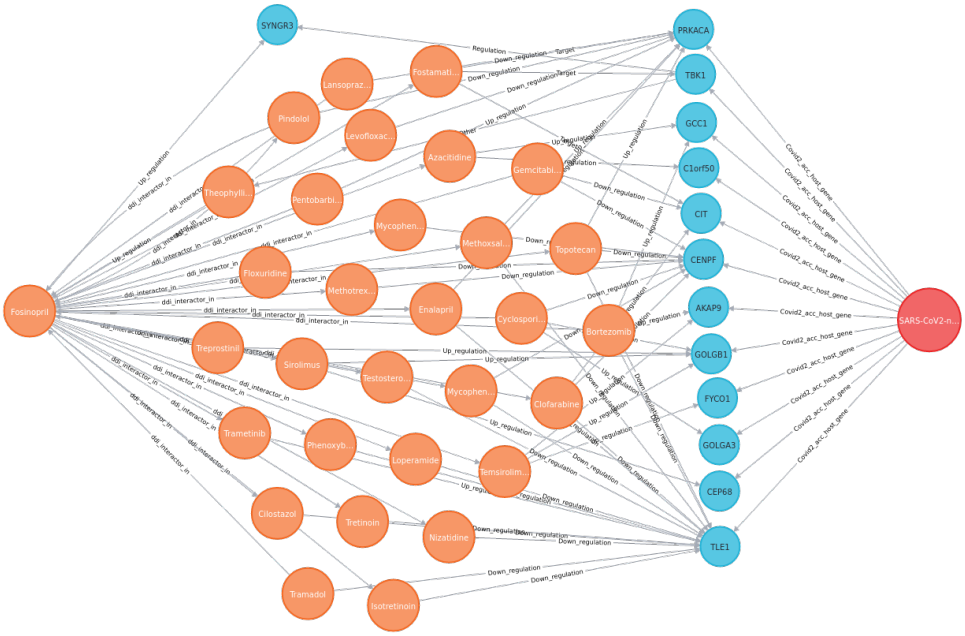
The supporting paths for the prediction are useful for experts interested in assessing the biological relevance of each path to the prediction. Explanatory paths provide working hypotheses for assessing the biological relevance of predicted *(Compound, Disease target)* pairs. Here, the hypotheses shown in Figure 4.6 suggest that the possible paths from Fosinopril to nsp13 in the KG include more than one step. This apparently contradicts the direct binding hypothesis evaluated above by molecular docking. However one should recall here that such direct binding is a new information not present in the KG and therefore this type of relation could not be retrieved as an explanatory path. In fact, we rather believe that both types of explanations may co-exist, reinforcing the value of a repurposed candidate drug.

4.4 Discussion

The cross-matching of our top-100 ranked compounds with in-trial compounds for COVID-19 reveals that our approach finds two distinguished in-trial drugs Ruxolitinib and Baricitinib which were not found by any of the aligned approaches in the literature. On January 14, 2022, two new drugs were recommended by World Health Organization (WHO) and Baricitinib was one of the recommendations [Org22]. WHO recommended Baricitinib strongly for patients with severe or critical COVID-19 as it suppresses the overstimulation of the immune system. For the Ruxolitinib, Iastrebner *et al.* (2021) [ICE+21] found a trend of lower mortality rate with manageable side effects and no direct organ injury among COVID-19 patients taking the drug [ICE+21]. To provide more insightful evaluations of our predictions, we compare them to known compounds for COVID-19 disease targets in terms of molecular evaluations results. Concerning the cluster analysis of known and predicted compounds for the SARS-CoV-2-nsp13 disease target, the MCS for the first cluster has the highest value although ligands from this cluster have diversity in their MWs (molecular weights). Fosinopril and Ridaforolimus are two interesting predicted ligands from this cluster as their MWs are close to few known ligands. The findings from cluster analysis are consistent to some extent to docking findings. Ligands from PDB are smaller in size than predicted and literature-based ligands, this fact limits the number of interactions that can be established with the nsp13 target. In addition, all ligands obtained from the PDB come from a study [NDY+21] using the crystallographic fragment screening technique which is known to be



(a)



(b)

Figure 4.6: Explanations for the best prediction (wrt to docking rank), Fosinopril-nsp13 pair. Orange, sky and red circles represent compounds, genes and disease targets respectively, black edges represent relations in a rule body and red edge represents the *Treat* relation in a rule head. (a) Different rules with at least one path for the pair of interest, () gives the number of paths instantiating a rule. (b) Network representation of the paths instantiating the learned rules for the predicted pair (Fosinopril, SARS-CoV-2-nsp13). We see a total of 51 supporting paths between entities of the pair.

sensitive to low affinity ligands [PBA14]. These could be possible reasons behind the scene that most of the ligands from PDB are ranked lower than ligands from predictions and literature. More than 80% (5 out of 6) known ligands from the first cluster (Table 4.3) appear in the top-20 of the molecular docking results. From the predicted ligands of this cluster, Fosinopril stands out and is ranked in the second position. Originally, Fosinopril is an angiotensin-converting enzyme (ACE) inhibitor and primarily used for the treatment of mild to moderate hypertension and some types of chronic heart failure [KWK+08]. This compound has few rare side effects such as orthostatic hypotension (2.7% cases), sexual dysfunction (1.7% cases), Angina pectoris (1.1-1.6% cases), rash (1-1.4% cases), Angioedema (0.2% cases) and Shock (0.2% cases)

(http://sideeffects.embl.de/drugs/3419/). Thus, it appears as a very promising candidate for drug repurposing against COVID-19.

Our docking results are strengthened by the fact that a good portion of the literature-based nsp13 ligands (24 out of 38) are present in DRKG and could be scored by our prediction model for an association with COVID-19 disease entities. Most of them (15 out of 24) have their best or second-best predicted disease entity being nsp13 (Table 4.4 and Table B.7). However, we also observe a few conflicts between the findings from prediction model and molecular evaluation. The best-ranked Diosmin in docking results is poorly ranked by our prediction model. Moreover, the best predicted disease target for this ligand is 'nsp9' though it was recommended for the nsp13 in the literature [WLC20]. This can be due to a deficit of information about this ligand in DRKG. Indeed, we checked availability of information about this ligand in DRKG in terms of its number of neighbours or degree and we see that its degree is much smaller than the average degree of top-500 compounds in the graph (20 versus 1250). This unfavorable situation for link prediction in KG is confirmed by the relatively low probability score of diosmin (0.162) with its preferred disease target nsp9. The same situation happens to few other conflicts: Dpnh (degree=302), Lifitegrast (degree=21), Paritaprevir (degree=727).

Comparing the binding poses of Diosmin (best docking score, known ligand) and Fosinopril (second-best docking score, predicted ligand), it is noteworthy that considerable segments of both poses have a very similar location (see Figure 4.5). Digging deeper into our results, we find that both ligands have in common 7 interactions with the protein which is an outstanding result (see Table 4.5). Comparing the binding poses and interaction maps between the four best-docked predicted ligands (Fosinopril, Macitentan, Eprosartan and Dinoprostone) and the three best-docked known ligands (Diosmin, Ergotamine and Risperdal), we observe that their 3D position in the nsp13 target largely overlap (Figure 4.5). More precisely, Table 4.5 shows that Dinoprostone and Diosmin share 7 interactions with nsp13 and Fosinopril has not only 7 interactions in common with Diosmin, but also 6 with Ergotamine and 5 with Risperdal. Thus, this analysis of interaction maps shows good consistency between the docked structures of predicted and known ligands, confirming the validity of our predicted nsp13 ligands. These results suggest that our tool is able to make predictions that are structurally consistent with the literature.

Considering now the explanatory paths found for the best predicted compound Fosinopril for the nsp13 disease target (Figure 4.6), we detected an interesting path:

$$Fosinopril \xrightarrow{\text{Up-regulation}} SYNGR3 \xleftarrow{\text{Regulation}} TBK1 \xleftarrow{\text{Covid2\_acc\_host\_gene}} nsp13,$$

where a gene up-regulated by the predicted compound is also regulated by a gene associated with the disease target nsp13. Synaptogyrin 3 (SYNGR3) (https://www.ncbi.nlm.nih.gov/gene/9143) encodes an integral membrane protein. The exact function of this gene is still not clear, but studies on a similar murine protein reveal that this gene is a synaptic vesicle protein that interacts with the dopamine transporter [KPS+98]. TANK binding kinase 1 (TBK1) (https://www.ncbi.nlm.nih.gov/gene/29110) is an important kinase for regulating inflammatory responses to foreign agents [PB99]. According to the LINC_L1000 connectivity map [HBB16], Fosinopril up-regulates SYNGR3 with a dysregulation z-score of 4.476 and TBK1 up-regulates SYNRG3 with a z-score of 5.372. Finally, Gordon *et al.* [GJB+20] provide a protein interaction map where TBK1 interacts with the nsp13 target. This analysis is just an example how to assess biological relevance of different explanations with the help of experts to identify relevant explanations.

Although the current study shows impressive drug repurposing performance for COVID-19, it presents several improvement possibilities. Firstly, the embedding generation approach may

generate low-quality embeddings due to data scarcity problem, a common issue for embedding methods. This can affect the subsequent drug repurposing performance by producing false (positive and negative) results as we have seen false negative result for the 'Diosmin' compound. Improving the knowledge graph by including more information about compounds, diseases, and other concepts may reduce false results. Secondly, the COVID-19 disease is still evolving and naturally the 2020 built DRKG may lack important information. For example, WHO recommended Sotrovimab (Drug-bank id: DB16355) for COVID-19 [Org22] which is missing in the current DRKG version. Lack of important information about the disease and compounds may affect the drug repurposing performance. Inclusion of recent information into the graph could reduce the problem, though this represents a challenging task. Thirdly, choosing the maximum path length, when learning rules and generating explanation(s) is another very challenging task, as high value will generate large number of low-quality rules and low value may miss useful rules. For example, the approach failed to generate any explanation for the predicted (Periciazine, nsp14) pair with maximum path length 3. Lastly, but not the least, both embedding and rule mining methods take high computational time. For example, each of the three embedding method took nearly 7 days to generate embeddings and the rule mining method took nearly 30 hours to learn rules. The implementation of both methods in distributed and parallel setting could minimize the problem. Another possible solution is to reduce the size of the KG by cleaning it. Our cleaning process is a step toward this objective.

4.5 Conclusion

The presented study demonstrates how complementary embedding methods can be used to generate high-quality ensemble embeddings of a KG and how to use embeddings for the drug repurposing task. To the best of our knowledge, this study is the first attempt to combine virtual screening methods with KG embedding methods in predicting and evaluating repurposable drugs for COVID-19. Besides the retrieval of many in-trial drugs, both methods show a converging result that 'Fosinopril' could be a new potential nsp13 inhibitor. Experimental validation of our predicted 'Fosinopril' compound to treat COVID-19 is another potential perspective of this study. The molecular evaluation and explanation(s) of the predictions in this study lead to a trustable conclusion. The rules, learned in this study, could be useful to build query patterns on other similar KG datasets. Although this study focuses on the COVID-19 disease, the designed drug repurposing framework is generic and could be applied to other diseases for which a KG exists. In this study, we provide molecular evaluation of predicted compounds for only nsp13 target. Evaluation of compounds for other COVID-19 disease targets such as nsp6 or nsp5-C145A, which appear as preferred targets in our top-20 predicted compounds, is another possible perspective of this study.

Finally, our ensemble embedding learning is general and could be applied to other applications, such as product recommendation in e-commerce, course recommendation in e-learning, and job recommendations in a job advertising platform.

Chapter 5

Distributed link prediction in knowledge graph

Contents

5.1 Introduction

In recent years, KG embedding methods, representing entities and relations in low dimensional vector space, have attracted much attention due to their advantages in downstream tasks such as Link prediction, entity alignment, entity typing *etc.* A plethora of KG embedding approaches exist in the literature (a few were described in Section 1.3.3, Chapter 1). Geometric embedding approaches, such as TransE [BUGD+13], TransH [WZFC14], DistMult [YYH+15a], RotateE [SDNT18a], have become popular among them due to their simplicity.
Due to huge size of real-world KGs, a single computational unit is not sufficient to store a KG and learn embeddings in a reasonable time. In this context, it would be interesting to implement KG embedding methods in a distributed setting where a KG is distributed to an array of machines and embeddings are learned on the machines in parallel. To date, a few distributed frameworks exist in the literature for training geometric embedding approaches, such as GraphVite [ZXTQ19], Pytorch-BigGraph (PBG) [LWS+19], DGL-KE [ZSM+20]. GraphVite and PBG use PyTorch-Distributed API for implementing communication and data exchange network. However, the use of the core level API is a challenging task for development and deployment [SQG+22]. Recently, Moritz *et al.* [MNW+18] developed the Ray platform to make it easier to develop and deploy distributed applications in a simple manner by hiding the low-level details of development and deployment. In this chapter, we present our distributed framework on the top of the Ray platform for geometric KG embedding approaches, *i.e.*, translational and semantic matching approaches. Our framework uses distributed storage (KG and embeddings), distributed training,

and Ring-Allreduce gradient sharing mechanisms. Overall, the head node (master) makes several partitions of a KG and sends each partition to each worker. Each worker then generates negatives using our simple negative sampling (SNS) method [IAST21c] and computes gradient for a batch of positives and negatives. Each worker shares it gradient to other workers by Ring-AllReduce mechanism and the gradients are then aggregated in each worker to update its embeddings. As all workers contain entity and relation embeddings, the trained model is evaluated on a randomly selected worker. Our framework is different from other frameworks in four major points. **Firstly**, to best of our knowledge, the only attempt to use Ray for distributed KG embedding was by Sheikh *et al.* [SQRL22] which supports distributed training of only GNN-based KG embedding approaches. We are the first ones to use Ray for distributed training of geometric KG embedding approaches. **Secondly,** our framework implements a high-performing SNS method whereas all other frameworks use random-uniform or variants of it for generating negatives. **Thirdly,** our framework uses Ring-AllReduce architecture for gradient sharing and embedding updating whereas the head node collects gradients from workers and update embedding in most existing frameworks. **Lastly,** the existing frameworks use complex graph partitioning methods whereas our framework use simple and fast random partitioning method of KGs.

5.2 The proposed distributed framework

Our proposed distributed framework is designed based on an All-Reduce architecture where all machines work (for embedding learning) as workers in contrast to master-slave architecture where the master manages resources and only workers learn embeddings. The architecture of our framework is illustrated in Figure 5.1.

We briefly describe the underlying Ray platform in Section 5.2.1. In Section 5.2.2, we describe major components of our framework and the embedding learning process.

5.2.1 The Ray platform

Ray is a recent, open-source and general-purpose cluster-computing platform to enable distributed training and simulation of machine learning models [MNW$^+$18]. It provides a simple and high-level abstractions for users and hides all the low-level tasks. Ray enables seamlessly scaling-up of Python-based applications. Ray considers each machine as single worker for executing a submitted task. In a distributed environment, Ray comprises a set of workers. The top-level components of Ray are illustrated in Figure 5.2.

Each worker manages

- one or more processes for job execution, and

- a raylet that consists of a job scheduler and a shared-memory, known as Plasma Object Store to communicate among all processes in the worker.

One of the workers is designated as the head node to initialize the distributed training process and to manage the communications among workers. In addition to the above components (Raylet), the head node also maintains a Global Control Store (GCS) for managing system-level metadata and a driver process to execute the top-level application. We refer to the article by Moritz *et al.* [MNW$^+$18] and Ray website for details about the Ray platform.

[8]Source: https://docs.ray.io/en/latest/ray-contribute/whitepaper.html
[8]https://docs.ray.io/en/latest/#

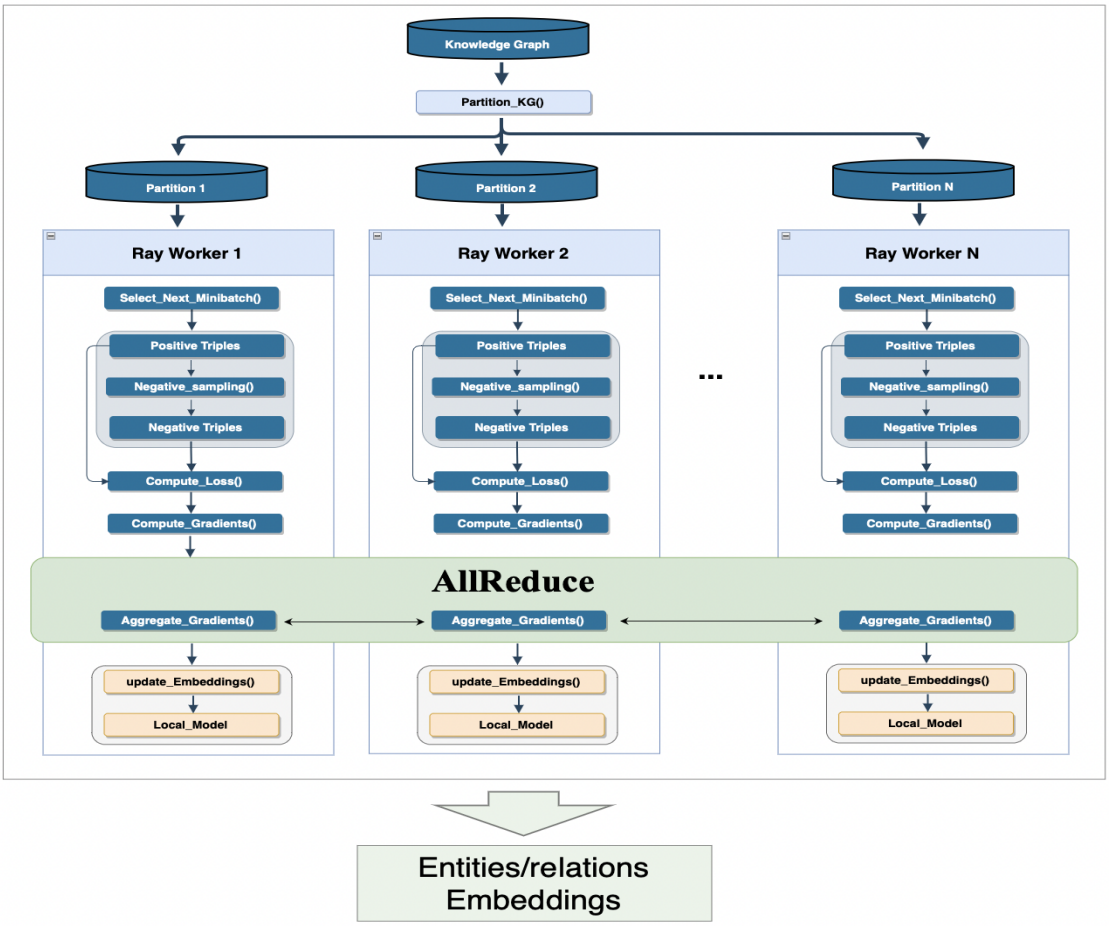


Figure 5.1: Architecture of the proposed distributed framework

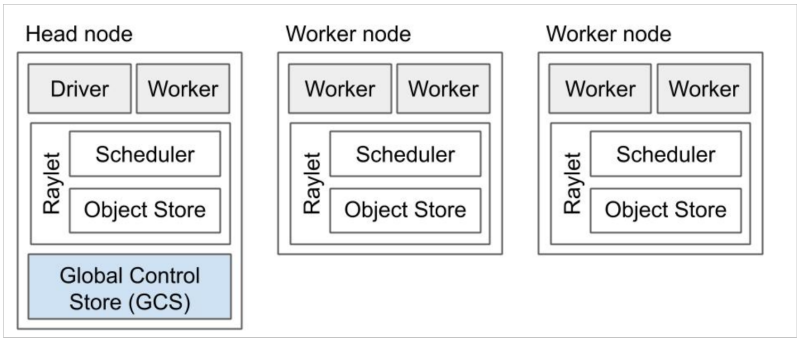


Figure 5.2: Top-level components of a Ray cluster [8]

5.2.2 Our distributed system for learning geometric embeddings of large KGs

In a broad sense, our framework consists of two components: a *head node/machine* and few *worker nodes/machines*. The head node is a special node that has a few special tasks like a master in master-slave architecture. The head node is responsible for fetching the KG and for

partitioning it into small partitions. The head node also initiates communications among all workers, sends one partition to each worker, and keeps one partition for itself. At the end of the training process, the head node is also responsible for returning the learned embeddings. During the learning phase, the head node also acts as a worker node. Each worker node receives a KG partition and is responsible for computing gradients and learning embeddings of KG entities and relations. Each worker contains an AllReduce sub-component to receive gradients from other workers and to compute its final gradients.

Let us now describe in more details the main steps of the distributed system.

Step 1. Establishing communication network

Our framework starts with establishing a communication network among all machines/ workers in a distributed environment. Let N be the total number of machines. Two popular architecture of communication networks are as following:

1. Shared parameter server architecture: In this architecture, N-1 machines (workers) are used for training on N-1 partitions of data and one machine is used as a parameter server. The data on the parameter server is shared among all the workers. The arrangement of machines in the network could be imagined as star topology where the parameter server machine stays at the centre and all workers are at end points. The workers use their data partitions to compute losses and gradients, and send their gradients to the parameter server to store and update parameters of training machine learning models [LASY14].

2. AllReduce architecture: In this architecture, each machine works as a worker and stores all trainable parameters [BH93, TRG05]. All workers compute losses and gradients based on their data partitions and share their gradients with all workers through a special component called AllReduce in each worker. All workers aggregate all gradients and they have the same gradients (Appendix C Figure C.1). Unlike shared parameter server, AllReduce has N workers for N machines in the distributed systems.

The advantage of better resource (computing machine) utilization of AllReduce architecture over parameter server architecture motivates us to design the communication network based on an AllReduce architecture. For the AllReduce case, two well-known architectures are Tree-AllReduce and Ring-AllReduce. In Tree-AllReduce architecture, every worker communicates with its left and right workers [YA18]. In the Ring-AllReduce architecture, worker nodes are arranged as a ring where every worker communicates with its proceeding node [SDB18]. Gloo [Fac21] and NCCL [NVI21] are two popular backend libraries to support the establishment of AllReduce architectures. The Gloo library supports only CPU while NCCL supports GPU nodes [MNW+18]. As we consider only GPUs for our worker nodes, we use NCCL as the backend for building our communication network.

Step 2. KG partitioning

The learning process starts with the head node fetching the triples of the input KG. Although there are many algorithms for partitioning simple graphs, MCS [ZWLL18] is the only KG partitioning algorithm in the literature to the best of our knowledge. As the source code of this algorithm is not publicly available, we simply implement a random graph partitioning algorithm. In random partitioning, we randomly split KG triples into a required number of partitions where each partition has nearly equal number triples. The head node produces N partitions of a KG for N nodes ($N-1$ workers and 1 head node) in our distributed environment.

Algorithm 3: The head node

Input: A knowledge graph (KG), number of workers (N), embedding method (KGE), Random partitioner (KG_Splitter), total number of epochs (T), number of batches (B), embedding optimizer (Optimizer), simple negative sampling (SNS)

```
/* Partition the KG using a KG partitioning method.                    */
Partitions[1 : N] ← KG_Splitter(KG,N)
Workers ← ∅
/* Create a temporary KGE model                                        */
model_tmp ← KGE(KG)
model_tmp.embeddings ← initialize_embeddings()
foreach Part_KG ∈ Partitions[2 : N] do
    /* Initialize a worker                                             */
    worker ← ∅
    worker.triples ← Part_KG.triples
    worker.model ← model_tmp
    worker.negative_sampler ← SNS
    Workers.insert(worker)
/* Transform the head node to a normal worker                          */
worker ← head_node
worker.triples ← Partitions[1].triples
worker.model ← model_tmp
worker.negative_sampler ← SNS
Workers.insert(worker)
/* Send signal to Ray platform to start training of all workers        */
Signal_parallel_training(Workers)
```

Step 3. Embedding model training

After the KG partitioning, the head node instantiates an embedding model by initializing the embeddings of KG entities and relations randomly from uniform/Gaussian distributions [WMWG17]. Our framework allows training of geometric embedding models such as TransE [BUGD+13], TransH [WZFC14], TransD [JHX+15], DistMult [YYH+15a], CompleX [TWR+16], RotatE [SDNT18a]. The head node sends copies of this model to other $N-1$ workers. The head node sends a signal to the Ray backend to start parallel training. At the same time it is transformed into a normal worker. The working steps in the head node are written in Algorithm 3. Hence, all N workers in our framework have same embeddings for all entities and relations. Each worker follows the same embedding learning process which is similar to some extent to a centralized embedding model. Each worker processes its triples in batch mode, *i.e.*, it creates a fixed number of batches of triples. Each worker then follows the following training steps for T times (epochs) to update its embedding model.

Computing gradients:

The worker fetches a batch of positive train triples S_m of size m from its partition and generates a corresponding batch of negative triples S'_m using a negative triple sampling method. We use our SNS method to generate high-quality negative triples where only the entity set of

the current partition is used to generate the candidate set. The batches of positive and negative triples are then used to train the embedding model. We consider the pairwise training strategy where the objective is to optimize the embeddings of entities and relations by minimizing the total pairwise loss (L) for the batch as defined in Equation 5.1.

$$\min_{\Theta} \sum_{\forall_{(h,r,t)\in S_m,(h',r,t')\in S'_m}} L(f(h,r,t), f(h',r,t')) + \lambda reg(\Theta) \tag{5.1}$$

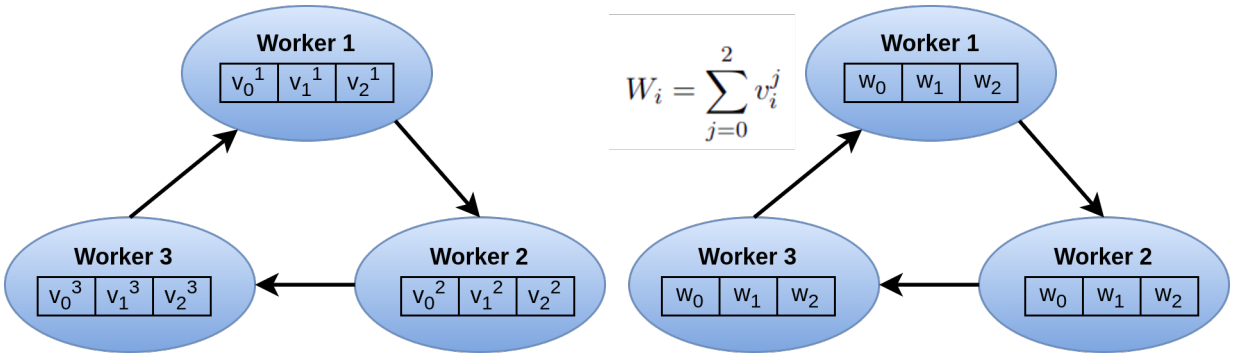
Here, f is the scoring function of the embedding model, $(h,r,t) \in S_m$ and $(h',r,t') \in S'_m$ are a positive and its corresponding negative triples. The loss for the pair of positive and negative triples is defined in Equation 5.2 [WMWG17].

$$L(f(h,r,t), f(h',r,t')) = \Big[\lambda - f(h,r,t) + f(h',r,t')\Big]_+ \tag{5.2}$$

Here, λ is the user defined margin and $[.]_+ = max(0,.)$ is the hinge function. The loss for the batch is then propagated to an optimizer to compute gradients. We use the popular Adam optimizer to compute gradients.

Aggregating gradients and update embeddings:

After calculating the gradients, the AllReduce component of each worker shares the gradients of its model to other workers. An illustration of gradient sharing in Ring-AllReduce architecture is provided in Appendix C, Figure C. We also refer to the work by Sergeev & Mike [SDB18] for details about gradient sharing in this architecture. After gradient sharing, each worker will have copies of gradients from all workers. The worker aggregates these gradients by simply summing them. Figure 5.3 illustrates the before and after sharing gradients among three workers where each worker contains three gradients. The mechanism of gradient sharing is illustrated in Appendix C, Figure C.1. Hence, it can be checked that all workers have exactly the same gradients after the gradient aggregation step. Each worker updates its model based on its aggregated



(a) Before gradient sharing (b) After gradient sharing and aggregation

Figure 5.3: Illustration of Ring-AllReduce gradient aggregation mechanism for a specific entity/relation in a 3-dimensional latent space. (a) Each worker i computes the gradient vector v_0^i, v_1^i, v_2^i for the entity/relation. (b) Each worker shares its gradients with other workers and aggregates all gradients.

gradients. Note that the learned models in all workers have the same embeddings as all workers use the same gradients to update them.

The worker repeats the above described embedding training step for all batches for all epochs. Algorithm 4 summarizes all the steps of an worker node. As all workers have the same learned model, our framework select the embedding model from the head node as the output model. The learned model is used in the downstream link prediction task.

Algorithm 4: The worker

Input: A worker (worker), number of epochs (T), number of batches (B), an embedding optimizer (Optimizer)
Output: Trained embedding model ($\text{KGE}_{trained}$)

```
/* Initialize the SNS method using information for the local partition.
   */
worker.SNS.initialize()
foreach epoch ∈ {1, 2, 3, ....., T} do
    foreach batch ∈ {1, 2, 3, ....., B} do
        gradient ← ∅
        /* Sample a batch of positive and negative triples.              */
        Triples.positive ← Extract_samples(worker.triples, batch)
        Triples.negative ← worker.SNS.sample_negative(Triples.positive)
        /* Compute total loss for the batch of triples                   */
        Loss_batch ← Total_pairwise_loss(Triples.positive,Triples.positive)
        gradients_batch ← Optimizer (Loss_batch, worker.model.embeddings)
        Gradients ← ∅
        /* Use All_Reduce to share and collect gradients from all workers
           for the current batch                                         */
        foreach wrkr ∈ Workers do
            gradient_wrkr ← worker.AllReduce.get_gradients(batch, wrkr)
            Gradients.insert(gradient_wrkr)
        /* Aggregate all collected gradients to update embeddings in the
           local embedding model                                         */
        gradient ← AllReduce.aggregate_gradients(Gradients)
        worker.model.update_embeddings(gradients)
KGE_trained ← worker.model
```

5.3 Experimental design

We use Pytorch 1.11.0 to implement our KG embedding methods and Ray 1.13.0 platform to manage distributed environment. We use TransE [BUGD+13] and DistMult [YYH+15a] to represent translational and semantic matching categories of geometric KG embedding methods. As for KG datasets, we use two large KG datasets, FB15K and YAGO3-10 (described in Table 3.1, Chapter 3).

We evaluate our distributed framework against link prediction performance and training time for embedding methods. We compute Hit@z for $z \in \{1, 3, 10\}$ and MRR metrics to track the link prediction performance of embedding methods (described in Section 1.3.1). The range of both scores is 0 to 1. Higher performance metrics mean better predictions.

To describe scalability of our framework, we compute the embedding method training time per

epoch. To demonstrate the gain in computational time, we compute speed-up metric as the ratio of total time in a distributed training and total training time in a centralized training as Equation 5.3 [ZSM+20].

$$Speed - up = \frac{Training - time_d}{Training - time_c} \tag{5.3}$$

Here, $Training - time_d$ and $Training - time_c$ are the total time (in seconds) for distributed training and centralized training of an embedding model on a KG dataset.

To represent centralized environment, we set the worker number to 1.

5.4 Experimental results

We use Grid'5000 testbed [BCAC+13], an array of high performance computing machines for the study of large-scale distributed applications, for executing our framework. All experiments are performed on cluster(s) with Nvidia GeForce RTX 2080 Ti GPUs (11 GB). For the centralized experiments, we use only one GPU without Ray.

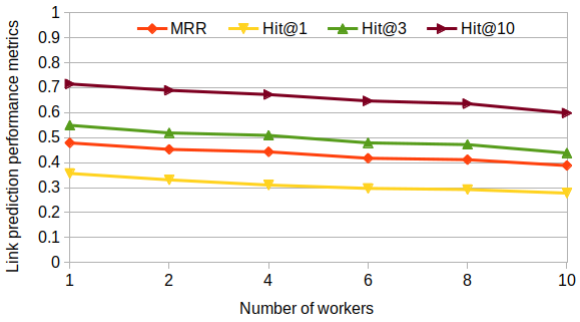
To compare our distributed framework with centralized framework, we run the centralized framework for 500 epochs for all datasets. In our distributed framework, we choose 2,4,6,8,10 as the number of workers. We set the number of epoch to 500, the number of batches to 120 for FB15K and to 250 for YAGO3-10 datasets. As for optimizer we use Adam optimizer and we set the learning rate to 0.001 and the margin to 4.0.

Figure 5.4 illustrates the prediction metrics and the computational time per epoch for FB15K dataset. Both TransE and DistMult show very close performance metrics for all numbers of workers. The centralized training ($worker = 1$) for FB15K shows performance metrics $MRR \approx 0.5, Hit@10 \approx 0.7, Hit@3 \approx 0.55, Hit@1 \approx 0.35$ (Figures 5.4a,5.4b). For distributed training ($workers \geq 2$), the results show a small decrease in the link prediction performance metrics. This is not observed with the larger YAGO3-10 dataset for which we see that the performance metrics in distributed training remain close to centralized training. This may be imputed to the difference in the size of the datasets. For example, the YAGO3-10 is nearly double in size compared to FB15K. As a result, the size of each partition is still high enough for YAGO3-10 dataset to learn good embeddings.

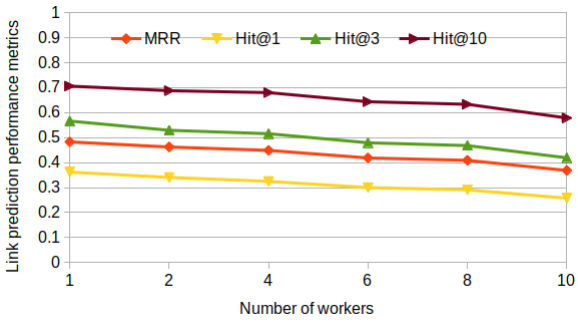
Figure 5.5 illustrates the computational time of our distributed framework per epoch for different numbers of workers. It is clearly seen that as the number of workers increases, the computational time decreases. To demonstrate the grain in computational time, we present the speed-up results in Figure 5.6. We see that the speed-up increases linearly for the embedding models. We find the best speed-up for TransE embedding model on FB15K dataset. Although the speed-up metrics for DistMult model on the larger YAGO3-10 dataset is lower than on FB15K, the speed metric is still linearly increasing. All in one, the Figures 5.4,5.5 and 5.6 demonstrate the scalability of our distributed framework to train a geometric embedding model for large scale KG datasets, with a good balance between performance loss and time gain.

5.5 Conclusion

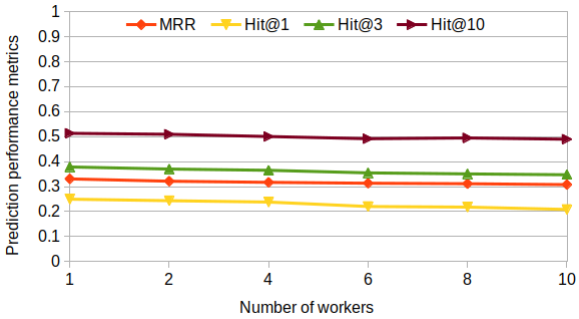
In this chapter, we proposed a preliminary distributed framework for training geometric embedding models for large scale KG datasets combined with our SNS method for sampling high-quality negative triples. The evaluation of our framework on large datasets demonstrate that our framework is able to train geometric embedding models for large scale KG datasets. The results we got concerning the prediction performance metrics in centralized versus distributed settings are



(a) Performance metrics of TransE on FB15K



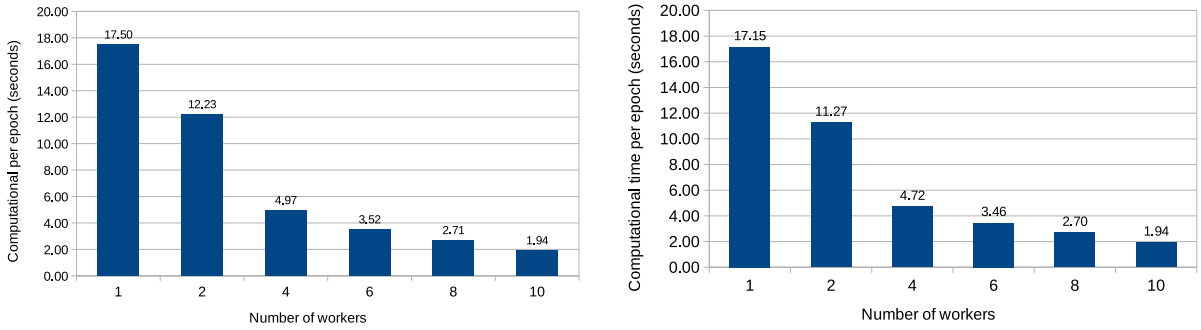
(b) Performance metrics of DistMult on FB15K



(c) Performance metrics of DistMult on YAGO3-10

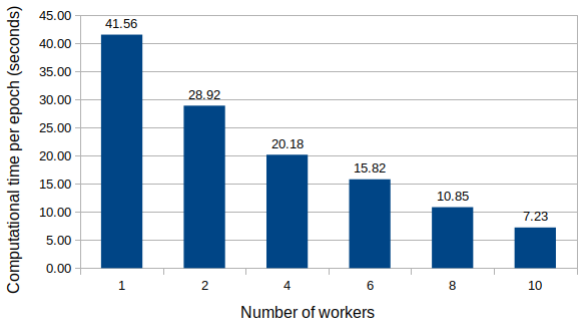
Figure 5.4: Link prediction performance metrics FB15K and YAGO datasets for different number of workers

not conclusive. Indeed we need to conduct additional experiments with more large KGs but also to test smarter partitioning strategy of the triple set. Nevertheless, the first results demonstrate both the feasibility of our designed architecture on the Ray platform and a good speedup when increasing the number of workers (machines).



(a) Computational time per epoch for TransE on FB15K

(b) Computational time per epoch for DistMult on FB15K



(c) Computational time per epoch for DistMult on YAGO3-10

Figure 5.5: Computational time per epoch on FB15K and YAGO3-10 datasets for different number of workers

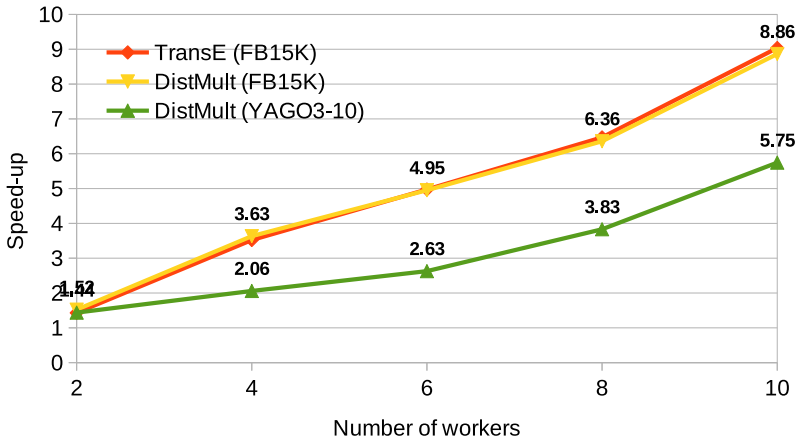


Figure 5.6: Speed-up of distributed training of two embedding models on the datasets. The numbers at each line show the speed-up metrics.

Chapter 6

Conclusion and perspectives

Contents

6.1 Conclusion

In this research, we addressed the link prediction problem in both simple graphs and knowledge graphs. We studied strategies for explaining link prediction to alleviate the 'black-box' limitation of embedding-based link prediction approaches. We formally defined the link prediction problem, and conducted an extensive review on state-of-the-art approaches for link prediction in simple and knowledge graphs (Chapter 1). Based on the current-state-of-the-art, we have raised a set questions during the development of this research and developed solutions to tackle them. To be precise, let us summarize the question-solution pairs here.

- **Question 1:** How do the similarity-based and the recent embedding-based approaches perform in different simple graphs with different properties? Is there any strategy to alleviate the 'black-box' limitation of the embedding-based approaches?

  Solution: In Chapter 2 Section 2.1, we studied a set of well-known similarity-based and embedding-based link prediction approaches in graphs with different structural properties from various domains. We identified a problem in the traditional way of computing precision in similarity-based link prediction approaches and provided a solution to minimize the problem. The presented comparison results show that embedding-based approaches offer better prediction performance in all graphs as expected although similarity-based approaches also out-performed in graphs with specific characteristics. This study also gives a common ground to compare the unsupervised similarity-based approaches with the semi-supervised embedding-based approaches. We then explored the possible links between the similarity-based and recent embedding-based approaches based on prediction performance comparison and link agreement percentage on the diverse graphs to get the idea about the learned heuristics in each embedding-based approach. Our analyses constitute a modest contribution to the 'black box' limitation of embedding-based approaches.

- **Question 2:** Do similarity-based heuristics compete or collaborate for link prediction task in simple graphs?

  Solution: In Chapter 2 Section 2.2, we ensembled a set of similarity-based metrics to design a set of structural features for links in simple graphs. Utilizing those features of links, we train classical supervised algorithms to classify the links. Our results revealed that the prediction performance of the supervised approach is very competitive compared to more complex embedding-based approaches in many graphs. To provide explanation of predictions, we computed importance scores of features to get the idea about which similarity-based metrics are dominant and contribute in link formation in simple graphs.

- **Question 3:** How to generate high-quality negative triples for training embedding methods in knowledge graphs?

  Solution: In Chapter 3 Section 3.2, we presented a new negative triples sampling method, SNS for knowledge graphs. SNS makes a good balance between exploitation and exploration for sampling high-quality negative triples for an embedding method. Our results shows that an embedding method provides better link prediction performance when SNS is used compared to other negative sampling methods.

- **Question 4:** How to explain the link predictions of an KG embedding-based method?

  Solution: In Chapter 3 Section 3.3, we described a new neuro-symbolic method to explain the link predictions of a KG embedding method. The embedding method learns embeddings of the KG entities and relations, and predicts a set of unobserved triples. The neuro-symbolic method then mines a set of rules based on the KG and its embeddings. Then we propose an abductive strategy to use the rule set for generating explanations of the predictions relying on the embedding method. Our results reveal that our explanation method is able to find explanations of many predicted triples.

- **Question 5:** How to apply the explainable link prediction method to drug repurposing task?

  Solution: In Chapter 4, we present a new drug repurposing framework for the current COVID-19 disease based on the link prediction task in a dedicated knowledge graph. The framework develops an ensemble embedding generation method for the drug repurposing knowledge graph (DRKG) to overcome limitations of an embedding method. Then a drug prediction model is designed to find potential drugs for the COVID-19 disease. Our framework is able to find more in-trial drugs for COVID-19. In addition, our framework provides an advanced molecular evaluation procedure of our predictions which is not provided by the aligned frameworks in the literature. Our framework also generates explanations of our predictions. The molecular evaluation and explanation of our predictions led to at least one reliable drug repurposing for COVID-19.

- **Question 6:** How to design a distributed framework for scalable training of embedding methods on large scale knowledge graphs?

  Solution: Chapter 5 presents a new distributed framework for training geometric embedding methods in a distributed environment. The distributed framework is designed based on a recent distributed environment management tool, namely Ray. Our initial experiments demonstrate the feasibility and the scalability of our framework to large knowledge graphs.

6.2 Current limitations and future perspectives

Although this thesis work provides satisfactory results for explainable link prediction in simple graphs and knowledge graphs, we identify a few limitations along with interesting perspectives to explore (in order to improve the impact of our work):

- **Fine tuning of hyperparameters:** Setting optimal values to hyperparameters (*e.g.* embedding dimension, number of epochs, batch size) in embedding methods for both simple graphs and knowledge graphs is an important aspect as it has an impact on learned embedding quality as well as on downstream link prediction task. Indeed, searching many combinations of hyperparameters may give the optimal parameter setting. However, this is a very time-consuming job and we only tried a few combinations of hyperparameters for searching the best parameter setting for embedding methods in our experiments.

- **Robustness checking of embedding models on large KGs:** It is an important task to verify the robustness of an embedding method. We executed the embedding methods on simple graphs multiple times for checking their robustness (in Chapter 2). However, the knowledge graphs in our experiments are much bigger and training an embedding method is computationally expensive. Hence, we were not able to check the robustness of knowledge graph embedding models. Thirdly, the distributed training of knowledge graph embedding methods shows decreasing trend in link prediction performance but increasing trend in speed-up in computational time with increasing number of machines (workers). Within this context, an important aspect is to achieve a good trade-off between prediction performance and computational time for a knowledge graph in our distributed framework. This would require to conduct many more experiments, which are resource (time and GPUs)-consuming.

- **Transferability of learned KG embedding models:** A promising paradigm of efficient machine learning is transfer learning, which focuses on transferring learned knowledge across domains. The general idea of transfer learning is to train a model in a domain (source domain) for a specific task and to reuse it for similar tasks on similar domains (target domains) to improve performance in the target domains [PY09, ZQD+20]. The transfer learning paradigm has been investigated and applied to image and language processing domains with performance guarantee [LGB+19, ARS+15]. Recently, few researchers also study the transferability or use of pre-trained models in simple graphs [LHB+21, QCD+20, HLG+19]. Studying transfer learning for KG embedding methods could be relevant in KGs as well to improve link prediction performance, where the embedding methods fail to provide satisfactory performance. For example, our experiments show unsatisfactory link prediction performance on DRKG (Table 4.1). An alternative strategy could be learning embeddings of proteins based on the triples coming from two dominant data-sources, namely STRING and IntAct. The protein embeddings are then used as initial embeddings while learning embeddings of DRKG, excluding the triples coming from the two dominant data-sources. This would correspond to transfer learning in knowledge graph embedding to improve DRKG embeddings by splitting it in smaller parts and making more visible less frequent data-sources.

- **Multiview KG embedding:** Multiview embedding has successfully been applied to the computer vision domain where multiple views of an image object are defined to learn a single embedding of the object [CICG17, ZHH+18, LGF+22]. A typical multiview embedding method consists of three major steps: (i) identifying multiple views of the object; (ii)

embedding learning learning on each view; and (iii) combining the view-specific embeddings [LYZ18].

The available biological KGs contain biological entities and different relations among them, mainly interactions between biomolecules and their functions. However, these KGs lack structural information of proteins and compounds, such as protein domains and chemical structures. An interesting perspective is to include structural information in those biological KGs in addition to interaction and function information to learn better embeddings through multiview embedding. Hence, three views could be defined namely structural view, interaction view, and functional view. A multiview KG embedding method could then be designed to learn high-quality embeddings. This could in particular improve the downstream drug-repurposing performance.

- **KG partitioning method:** There exists a few simple graph partitioning methods in the literature, such as METIS [KK98] and KaHIP [SS11]. These methods can not be adapted to knowledge graphs partitioning due to three additional challenges. Firstly, KGs have named relations between entity pairs. It is important to consider the relation names when partitioning the KGs as ignoring relation names can lead to imbalance in number of triples among different partitions for a certain relation. Secondly, KGs allow multiple directed relations between entity pairs. It is important to make balance among different partitions from the view of number of relations and entities and graph connectivity. Thirdly, most of the real world KGs are huge in size and a single machine may not be enough to store the whole KG during partitioning task. To the best of our knowledge, Message Cluster and Streaming (MCS) is the only method to make partitions of KGs based on clustering of the KG. However, this method still overlooks the presence of different relation names in KGs. Therefore, the development of a KG partitioning method is an urgent and potential perspective of this research. A simple possibility could be performing relation-wise partitioning of triples. Another possibility would be using expert-defined views (discussed in the previous paragraph) on a KG to partition it.

Bibliography

[A+05] Robert Ackland et al. Mapping the us political blogosphere: Are conservative bloggers more prominent? In *BlogTalk Downunder Conference, Sydney*, 2005.

[AA03] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social Networks*, 25(3):211–230, 2003.

[ADK20] Yusuf O Adeshina, Eric J Deeds, and John Karanicolas. Machine learning classification can reduce false positives in structure-based virtual screening. *Proceedings of the National Academy of Sciences*, 117(31):18477–18488, 2020.

[AEB16] Cherry Ahmed, Abeer ElKorany, and Reem Bahgat. A supervised learning approach to link prediction in twitter. *Social Network Analysis and Mining*, 6(1):1–11, 2016.

[AHCSZ06] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. Link prediction using supervised learning. In *Workshop on Link Analysis, Counter-terrorism and Security, Sixth SIAM International Conference on Data Mining, Maryland*, volume 30, pages 798–805, 2006.

[ALB+21] Melissa F Adasme, Katja L Linnemann, Sarah Naomi Bolz, Florian Kaiser, Sebastian Salentin, V Joachim Haupt, and Michael Schroeder. PLIP 2021: Expanding the scope of the protein–ligand interaction profiler to DNA and RNA. *Nucleic Acids Research*, 49(1):530–534, 2021.

[AMH+20] Francis E Agamah, Gaston K Mazandu, Radia Hassan, Christian D Bope, Nicholas E Thomford, Anita Ghansah, and Emile R Chimusa. Computational/in silico methods in drug target and lead prediction. *Briefings in Bioinformatics*, 21(5):1663–1675, 2020.

[ARS+15] Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. Factors of transferability for a generic convnet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(9):1790–1802, 2015.

[ASGR+21] Jacob Al-Saleem, Roger Granet, Srinivasan Ramakrishnan, Natalie A Ciancetta, Catherine Saveson, Chris Gessner, and Qiongqiong Zhou. Knowledge graph-based approaches to drug repurposing for COVID-19. *Journal of Chemical Information and Modeling*, 61(8):4058–4067, 2021.

[BA99] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[BAH19] Ivana Balavzevic, Carl Allen, and Timothy Hospedales. Tucker: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5185–5194, 2019.

[BB09] Alberto Giovanni Busetto and Joachim M Buhmann. Stable bayesian parameter estimation for biological dynamical systems. In *2009 International Conference on Computational Science and Engineering*, volume 1, pages 148–157. IEEE, 2009.

[BCAC+13] Daniel Balouek, Alexandra Carpen Amarie, Ghislain Charrier, Frédéric Desprez, Emmanuel Jeannot, Emmanuel Jeanvoine, Adrien Lèbre, David Margery, Nicolas Niclausse, Lucas Nussbaum, Olivier Richard, Christian Pérez, Flavien Quesnel, Cyril Rohr, and Luc Sarzyniec. Adding virtualization capabilities to the Grid'5000 testbed. In Ivan I. Ivanov, Marten van Sinderen, Frank Leymann, and Tony Shan, editors, *Cloud Computing and Services Science*, volume 367 of *Communications in Computer and Information Science*, pages 3–20. Springer, 2013.

[BCG+18] Bradley R Bebee, Daniel Choi, Ankit Gupta, Andi Gutmans, Ankesh Khandelwal, Yigit Kiran, Sainath Mallidi, Bruce McGaughy, Mike Personick, Karthik Rajan, et al. Amazon neptune: Graph data management in the cloud. In *Seventeenth International Semantic Web Conference (ISWC'18), Monterey*, 2018.

[BCW14] Antoine Bordes, Sumit Chopra, and Jason Weston. Question answering with subgraph embeddings. In *2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*, pages 615–620. Association for Computational Linguistics (ACL), 2014.

[BGWB14] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259, 2014.

[BH93] Jehoshua Bruck and Ching-Tien Ho. Efficient global combine operations in multiport message-passing systems. *Parallel Processing Letters*, 3(04):335–346, 1993.

[BHC+09] Russell Bell, Alan Hubbard, Rakesh Chettier, Di Chen, John P Miller, Pankaj Kapahi, Mark Tarnopolsky, Sudhir Sahasrabuhde, Simon Melov, and Robert E Hughes. A human protein interaction network shows conservation of aging processes between human and invertebrate species. *PLoS Genetics*, 5(3):e1000414, 2009.

[BHT11] Meng Bai, Ke Hu, and Yi Tang. Link prediction based on a semi-local similarity index. *Chinese Physics B*, 20(12):128902, 2011.

[BKR10] Nesserine Benchettara, Rushed Kanawati, and Celine Rouveirol. A supervised machine learning link prediction approach for academic collaboration recommendation. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, pages 253–256, 2010.

[BLP+18] Emmanuel Bresso, Claire Lacomblez, Anne Pizard, Patrick Rossignol, Faiez Zannad, Malika Smaïl-Tabbone, and Marie-Dominique Devignes. A data science approach for exploring differential expression profiles of genes in transcriptomic studies-application to the understanding of ageing in obese and lean rats in the fight-hf project. In *JOBIM 2018-Journées Ouvertes Biologie, Informatique et Mathématiques*, 2018.

[BLW86] Norman Biggs, E Keith Lloyd, and Robin J Wilson. *Graph Theory, 1736-1936*. Oxford University Press, 1986.

[Bon87] Phillip Bonacich. Power and centrality: A family of measures. *American Journal of Sociology*, 92(5):1170–1182, 1987.

[Bre01] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[BSCH18] Dan Busbridge, Dane Sherburn, Pietro Cavallo, and Nils Y Hammerla. Relational graph attention networks. In *International Conference on Learning Representations*, 2018.

[BUGD+13] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems-Volume 2*, pages 2787–2795, 2013.

[BVRdAL15] Lilian Berton, Jorge Valverde-Rebaza, and Alneu de Andrade Lopes. Link prediction in graph construction for supervised and semi-supervised learning. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2015.

[BWF+00] Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, January 2000.

[BWU14] Antoine Bordes, Jason Weston, and Nicolas Usunier. Open question answering with weakly supervised embedding models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 165–180. Springer, 2014.

[CALR13] Carlo Vittorio Cannistraci, Gregorio Alanis-Lobato, and Timothy Ravasi. From link-prediction in brain connectomes and protein interactomes to the local-community-paradigm in complex networks. *Scientific Reports*, 3(1):1–14, 2013.

[CBK+10] Xiongcai Cai, Michael Bain, Alfred Krzywicki, Wayne Wobcke, Yang Sok Kim, Paul Compton, and Ashesh Mahidadia. Collaborative filtering for people to people recommendation in social networks. In *Australasian Joint Conference on Artificial Intelligence*, pages 476–485. Springer, 2010.

[CDZ+21] Huajun Chen, Shumin Deng, Wen Zhang, Zezhong Xu, Juan Li, and Evgeny Kharlamov. Neural symbolic reasoning with knowledge graphs: Knowledge extraction, relational reasoning, and inconsistency checking. *Fundamental Research*, 1(5):565–573, 2021.

[CHW22] Yen-Liang Chen, Chen-Hsin Hsiao, and Chia-Chi Wu. An ensemble model for link prediction based on graph embedding. *Decision Support Systems*, 157:113753, 2022.

[CHY11] William Cukierski, Benjamin Hamner, and Bo Yang. Graph-based features for supervised link prediction. In *The 2011 International Joint Conference on Neural Networks*, pages 1237–1244. IEEE, 2011.

[CICG17] Guanqun Cao, Alexandros Iosifidis, Ke Chen, and Moncef Gabbouj. Generalized multi-view embedding for visual recognition and cross-modal retrieval. *IEEE Transactions on Cybernetics*, 48(9):2542–2555, 2017.

[CLS+19] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 257–266, 2019.

[CLX15] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the Twenty-fourth ACM International on Conference on Information and Knowledge Management*, pages 891–900, 2015.

[CLYY16] Mo Chen, Feng Li, Ge Yu, and Dan Yang. Extreme learning machine based point-of-interest recommendation in location-based social networks. In *Proceedings of the International Conference on Extreme Learning Machines*, volume 6 of *Proceedings in Adaptation, Learning and Optimization (PALO)*, pages 249–261. Springer, 2016.

[CMN08] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.

[CMX18] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations*, 2018.

[Cog16] Cognonto. Kbpedia knowledge graph version 1.10 released. https://kbpedia.org/resources/news/kbpedia-knowledge-graph-version-1.10-released/, 2016.

[CPS10] Michael Chertkov, Feng Pan, and Mikhail G Stepanov. Predicting failures in power grids: The case of static overloads. *IEEE Transactions on Smart Grid*, 2(1):162–172, 2010.

[CRK+21] Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan Reddy. Probabilistic entity representation model for reasoning over knowledge graphs. In *Advances in Neural Information Processing Systems*, volume 34, pages 23440–23451, 2021.

[CSH+14] Ara Cho, Junha Shin, Sohyun Hwang, Chanyoung Kim, Hongseok Shim, Hyojin Kim, Hanhae Kim, and Insuk Lee. Wormnet v3: a network-assisted hypothesis-generating server for caenorhabditis elegans. *Nucleic Acids Research*, 42(1):76–82, 2014.

[CW18] Liwei Cai and William Yang Wang. KBGAN: Adversarial learning for knowledge graph embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1470–1480, 2018.

[CWFJ22] Guangfu Chen, Haibo Wang, Yili Fang, and Ling Jiang. Link prediction by deep non-negative matrix factorization. *Expert Systems with Applications*, 188:115991, 2022.

[CWPZ18] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 31(5):833–852, 2018.

[CWZ+20] Zhe Chen, Yuehan Wang, Bin Zhao, Jing Cheng, Xin Zhao, and Zongtao Duan. Knowledge graph completion: A review. *IEEE Access*, 8:192435–192456, 2020.

[CXP+09] Jie Chen, Bin Xin, Zhihong Peng, Lihua Dou, and Juan Zhang. Optimal contraction theorem for exploration–exploitation tradeoff in search and optimization. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 39(3):680–691, 2009.

[CXW+19] Guangfu Chen, Chen Xu, Jingyi Wang, Jianwen Feng, and Jiqiang Feng. Graph regularization weighted nonnegative matrix factorization for link prediction in weighted complex network. *Neurocomputing*, 369:50–60, 2019.

[CXW+20] Guangfu Chen, Chen Xu, Jingyi Wang, Jianwen Feng, and Jiqiang Feng. Robust non-negative matrix factorization for link prediction in complex networks using manifold regularization and sparse learning. *Physica A: Statistical Mechanics and its Applications*, 539:122882, 2020.

[CZC18] H. Cai, V. W. Zheng, and K. C. C. Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, 2018.

[DA05] Jordi Duch and Alex Arenas. Community detection in complex networks using extremal optimization. *Physical Review E*, 72:027104, 2005.

[DAHS+20] Nur Nasuha Daud, Siti Hafizah Ab Hamid, Muntadher Saadoon, Firdaus Sahran, and Nor Badrul Anuar. Applications of link prediction in social networks: A review. *Journal of Network and Computer Applications*, 166:102716, 2020.

[DBPL19] Vibhavari Dasagi, Jake Bruce, Thierry Peynot, and Jürgen Leitner. Ctrl-z: Recovering from instability in reinforcement learning. *arXiv preprint arXiv:1910.03732*, 2019.

[DDZ+18] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *International Conference on Learning Representations*, 2018.

[DKWW11] Yuxiao Dong, Qing Ke, Bai Wang, and Bin Wu. Link prediction based on local information. In *2011 International Conference on Advances in Social Networks Analysis and Mining*, pages 382–386. IEEE, 2011.

[DSP11] Hially Rodrigues De Sa and Ricardo BC Prudencio. Supervised link prediction in weighted networks. In *The 2011 International Joint Conference on Neural Networks*, pages 2281–2288. IEEE, 2011.

[DWCY13] Werner Dubitzky, Olaf Wolkenhauer, Kwang-Hyun Cho, and Hiroki Yokota. *Encyclopedia of systems biology*, volume 402. Springer New York, 2013.

[EI18] Takuma Ebisu and Ryutaro Ichise. Toruse: Knowledge graph embedding on a lie group. In *Thirty-second AAAI Conference on Artificial Intelligence*, 2018.

[ENRP22] Gavin Edwards, Sebastian Nilsson, Benedek Rozemberczki, and Eliseo Papa. Explainable biomedical recommendations via reinforcement learning reasoning on knowledge graphs. In *Workshop on Machine Learning on Graphs, Twenty-second International Conference on Data Mining (ICDM)*. IEEE, 2022.

[Fac21] Facebook. Gloo: a collective communications library. https://github.com/facebookincubator/gloo, 2021.

[FCFP21] Giulia Fiscon, Federica Conte, Lorenzo Farina, and Paola Paci. Saverunner: A network-based algorithm for drug repurposing and its application to COVID-19. *PLoS Computational Biology*, 17(2):e1008686, 2021.

[Fre79] L Freeman. Centrality in networks: I. conceptual clarifications. social networks. *Social Network*, 1979.

[GCL+17] Man Gao, Ling Chen, Bin Li, Yun Li, Wei Liu, and Yong-cheng Xu. Projection-based link prediction in a bipartite network. *Information Sciences*, 376:158–171, 2017.

[GCV+07] Kwang-Il Goh, Michael E. Cusick, David Valle, Barton Childs, Marc Vidal, and Albert-László Barabási. The human disease network. *Proceedings of the National Academy of Sciences*, 104(21):8685–8690, 2007.

[GDX22] Zhenxiang Gao, Pingjian Ding, and Rong Xu. Kg-predict: A knowledge graph computational framework for drug repurposing. *Journal of Biomedical Informatics*, 132:104133, 2022.

[GJB+20] David E Gordon, Gwendolyn M Jang, Mehdi Bouhaddou, Jiewei Xu, Kirsten Obernier, Kris M White, Matthew J O'Meara, Veronica V Rezelj, Jeffrey Z Guo, Danielle L Swaney, et al. A sars-cov-2 protein interaction map reveals targets for drug repurposing. *Nature*, 583(7816):459–468, 2020.

[GL16] Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. In *Proceedings of the Twenty-second ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864, 2016.

[GMC20] Sara Gerke, Timo Minssen, and Glenn Cohen. Ethical and legal challenges of artificial intelligence-driven healthcare. In *Artificial Intelligence in Healthcare*, pages 295–336. Elsevier, 2020.

[Goo13] Google. Freebase data dumps. https://developers.google.com/freebase/data, 2013.

[GPAM+14] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, pages 2672–2680, 2014.

[GRS90] J. Gasteiger, C. Rudolph, and J. Sadowski. Automatic generation of 3D-atomic coordinates for organic molecules. *Tetrahedron Computer Methodology*, 3(6, Part C):537–547, January 1990.

[GTH+20] Yiyue Ge, Tingzhong Tian, Suling Huang, Fangping Wan, Jingxin Li, Shuya Li, Hui Yang, Lixiang Hong, Nian Wu, Enming Yuan, et al. A data-driven drug repositioning framework discovered a potential therapeutic agent targeting COVID-19. *BioRxiv*, 2020.

[GTHS13] Luis Antonio Galarraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the Twenty-second International Conference on World Wide Web*, pages 413–422, 2013.

[GTHS15] Luis Galarraga, Christina Teflioudi, Katja Hose, and Fabian M Suchanek. Fast rule mining in ontological knowledge bases with AMIE. *The VLDB Journal*, 24(6):707–730, 2015.

[Gur20] Arun Bahadur Gurung. In silico structure modelling of sars-cov-2 nsp13 helicase and nsp14 and repurposing of fda approved antiviral drugs as dual inhibitors. *Gene Reports*, 21:100860, 2020.

[H+89] Lieve Hamers et al. Similarity measures in scientometric research: The jaccard index versus saltons cosine formula. *Information Processing and Management*, 25(3):315–18, 1989.

[HBB16] Daniel Himmelstein, Leo Brueggeman, and Sergio Baranzini. Consensus signatures for lincs l1000 perturbations. *Figshare. doi: https://doi. org/10.6084/m9. figshare*, 3085426:v1, 2016.

[HBC+21] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. Knowledge graphs. *ACM Computing Surveys (CSUR)*, 54(4):1–37, 2021.

[He16] Qi He. Building the LinkedIn knowledge graph. https://engineering.linkedin.com/blog/2016/10/building-the-linkedin-knowledge-graph#, 2016.

[HHB+03] Mark S Handcock, David R Hunter, Carter T Butts, Steven M Goodreau, and Martina Morris. statnet: An R package for the statistical modeling of social networks. http://www.csde.washington.edu/statnet, 2003.

[HL94] Richard A Harshman and Margaret E Lundy. Parafac: Parallel factor analysis. *Computational Statistics & Data Analysis*, 18(1):39–72, 1994.

[HLF+15] Chaobo He, Hanchao Li, Xiang Fei, Yong Tang, and Jia Zhu. A topic community-based method for friend recommendation in online social networks via joint non-negative matrix factorization. In *2015 Third International Conference on Advanced Cloud and Big Data*, pages 28–35. IEEE, 2015.

[HLG+19] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2019.

[HLH19] Kairong Hu, Hai Liu, and Tianyong Hao. A knowledge selective adversarial network for link prediction in knowledge graph. In *International Conference on Natural Language Processing and Chinese Computing*, pages 171–183. Springer, 2019.

[HSGE+18] Vinh Thinh Ho, Daria Stepanova, Mohamed H Gad-Elrab, Evgeny Kharlamov, and Gerhard Weikum. Rule learning from knowledge graphs guided by embedding models. In *International Semantic Web Conference*, pages 72–90. Springer, 2018.

[HTL+15] Yonghang Huang, Yong Tang, Chunying Li, Zhengyang Wu, and Haoye Dong. A method for latent-friendship recommendation based on community detection in social network. In *2015 12th Web Information System and Application Conference (WISA)*, pages 3–8. IEEE, 2015.

[Hua10] Zan Huang. Link prediction based on graph topology: The predictive value of generalized clustering coefficient. In *Proceedings of the Workshop on Link Analysis*, 2010.

[HWC+21] Kanglin Hsieh, Yinyin Wang, Luyao Chen, Zhongming Zhao, Sean Savitz, Xiaoqian Jiang, Jing Tang, and Yejin Kim. Drug repurposing for COVID-19 using graph neural network and harmonizing multiple evidence. *Scientific Reports*, 11(1):1–13, 2021.

[HZ94] Geoffrey E Hinton and Richard S Zemel. Autoencoders, minimum description length, and helmholtz free energy. *Advances in Neural Information Processing Systems*, 6:3–10, 1994.

[HZLL19] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. Knowledge graph embedding based question answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 105–113, 2019.

[IAST20a] Kamrul Islam, Sabeur Aridhi, and Malika Smaïl-Tabbone. A comparative study of similarity-based and gnn-based link prediction approaches. In *GEM (Graph Embedding and Mining) workshop, ECML-PKDD*, 2020.

[IAST20b] Md Kamrul Islam, Sabeur Aridhi, and Malika Smaïl-Tabbone. A comparative study of similarity-based and gnn-based link prediction approaches. In *Graph embedding (GEM) workshop*, pages 1–18. ECML-PKDD, 2020.

[IAST21a] Md Kamrul Islam, Sabeur Aridhi, and Malika Smaïl-Tabbone. Appraisal study of similarity-based and embedding-based link prediction methods on graphs. In *10th International Conference on Data Mining & Knowledge Management Process (CDKP 2021)*, pages 81–92. AIRCC Publishing Corporation, 2021.

[IAST21b] Md Kamrul Islam, Sabeur Aridhi, and Malika Smaïl-Tabbone. An experimental evaluation of similarity-based and embedding-based link prediction methods on graphs. *International Journal of Data Mining & Knowledge Management Process*, 11:1–18, 2021.

[IAST21c] Md Kamrul Islam, Sabeur Aridhi, and Malika Smaïl-Tabbone. Simple negative sampling for link prediction in knowledge graphs. In *International Conference on Complex Networks and Their Applications*, pages 549–562. Springer, 2021.

[ICE+21] Marcelo Iastrebner, Joaquin Castro, Edgardo Garcia Espina, Carolina Lettieri, Silvio Payaslian, Maria Celia Cuesta, Pablo Gutierrez, Araceli Mandrile, Angela Paola Contreras, Sebastian Gervasoni, et al. Ruxolitinib in severe COVID-19: Results of a multicenter, prospective, single arm, open-label clinical study to investigate the efficacy and safety of ruxolitinib in patients with COVID-19 and severe acute respiratory syndrome. *Revista de la Facultad de Ciencias Medicas*, 78(3):294, 2021.

[ISM+20] Vassilis N Ioannidis, Xiang Song, Saurav Manchanda, Mufei Li, Xiaoqin Pan, Da Zheng, Xia Ning, Xiangxiang Zeng, and George Karypis. Drkg-drug repurposing knowledge graph for COVID-19. *Github https:// github.com/ gnn4dr/ DRKG*, 2020.

[Jac01] Paul Jaccard. Etude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37:547–579, 1901.

[JHX+15] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (volume 1: Long papers)*, pages 687–696, 2015.

[Joa02] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–142, 2002.

[JWG+97] Gareth Jones, Peter Willett, Robert C Glen, Andrew R Leach, and Robin Taylor. Development and validation of a genetic algorithm for flexible docking. *Journal of Molecular Biology*, 267(3):727–748, 1997.

[JWL+16] Yantao Jia, Yuanzhuo Wang, Hailun Lin, Xiaolong Jin, and Xueqi Cheng. Locally adaptive translation for knowledge graph embedding. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[Kat53] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[KB15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representation*, 2015.

[KBS+20] Anisha Kumari, Ranjan Kumar Behera, Kshira Sagar Sahoo, Anand Nayyar, Ashish Kumar Luhach, and Satya Prakash Sahoo. Supervised link prediction using structured-based feature extraction in social network. *Concurrency and Computation: Practice and Experience*, page e5839, 2020.

[KBS+22] Anisha Kumari, Ranjan Kumar Behera, Kshira Sagar Sahoo, Anand Nayyar, Ashish Kumar Luhach, and Satya Prakash Sahoo. Supervised link prediction using structured-based feature extraction in social network. *Concurrency and Computation: Practice and Experience*, 34(13):e5839, 2022.

[KBV09] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[KCC+21] Sunghwan Kim, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, Leonid Zaslavsky, Jian Zhang, and Evan E Bolton. PubChem in 2021: new data content and improved web interfaces. *Nucleic Acids Research*, 49(1):1388–D1395, 2021.

[KD07] Stanley Kok and Pedro Domingos. Statistical predicate invention. In *Proceedings of the 24th International Conference on Machine Learning*, pages 433–440, 2007.

[KHS18] Sanja Krakan, Luka Humski, and Zoran Skočir. Determination of friendship intensity between online social network users based on their interaction. *Tehnički Vjesnik*, 25(3):655–662, 2018.

[KK98] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[KLS+19] Istvan A Kovacs, Katja Luck, Kerstin Spirohn, Yang Wang, Carl Pollis, Sadie Schlabach, Wenting Bian, Dae-Kyum Kim, Nishka Kishore, Tong Hao, et al. Network-based prediction of protein interactions. *Nature Communications*, 10(1):1–8, 2019.

[Kly04] Graham Klyne. Resource description framework (RDF): Concepts and abstract syntax. *W3C Recommendation*, 2004.

[KMR18] Praveen Kumar and G Ram Mohana Reddy. Friendship recommendation system using topological structure of social networks. In *Progress in Intelligent Computing Techniques: Theory, Practice, and Applications*, pages 237–246. Springer, 2018.

[KPS+98] Darek Kedra, Hua-Qin Pan, Eyal Seroussi, Ingegerd Fransson, Cecile Guilbaud, John E Collins, Ian Dunham, Elisabeth Blennow, Bruce A Roe, Fredrik Piehl, et al. Characterization of the human synaptogyrin gene family. *Human Genetics*, 103(2):131–141, 1998.

[KS21] Charilaos I Kanatsoulis and Nicholas D Sidiropoulos. Tex-graph: Coupled tensor-matrix knowledge-graph embedding for COVID-19 drug repurposing. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pages 603–611. SIAM, 2021.

[KSBBSTA22] Mohamed Hassen Kerkache, Lamia Sadeg-Belkacem, Fatima Benbouzid-Si Tayeb, and Amri Ali. Supervised learning using community detection for link prediction. In *International Conference on Computing Systems and Applications*, pages 85–94. Springer, 2022.

[KSSB20] Ajay Kumar, Shashank Sheshar Singh, Kuldeep Singh, and Bhaskar Biswas. Link prediction techniques, applications, and performance: A survey. *Physica A: Statistical Mechanics and its Applications*, 553:124289, 2020.

[KW14] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[KW16a] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of International Conference on Learning Representations*, pages 4700–4708, 2016.

[KW16b] Thomas N Kipf and Max Welling. Variational graph auto-encoders. In *Bayesian Deep Learning Workshop, Advances in Neural Information Processing Systems*, 2016.

[KWK+08] Ilka Knütter, Claudia Wollesky, Gabor Kottra, Martin G Hahn, Wiebke Fischer, Katja Zebisch, Reinhard HH Neubert, Hannelore Daniel, and Matthias Brandsch. Transport of angiotensin-converting enzyme inhibitors by h+/peptide transporters revisited. *Journal of Pharmacology and Experimental Therapeutics*, 327(2):432–441, 2008.

[LASY14] Mu Li, David G Andersen, Alexander J Smola, and Kai Yu. Communication efficient distributed machine learning with the parameter server. *Advances in Neural Information Processing Systems*, 27, 2014.

[LBEvH21] Dimitri Leggas, Muthu Baskaran, James Ezick, and Brendan von Hofe. Filtered tensor construction and decomposition for drug repositioning. In *2021 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–7. IEEE, 2021.

[LDP+96] Yanling Liu, Ghassan Dehni, Karen J Purcell, Joshua Sokolow, Maria Luisa Carcangiu, Spyros Artavanis-Tsakonas, and Stefano Stifani. Epithelial expression and chromosomal location of humantlegenes: Implications for notch signaling and neoplasia. *Genomics*, 31(1):58–64, 1996.

[LGB+19] Nelson F Liu, Matt Gardner, Yonatan Belinkov, Matthew E Peters, and Noah A Smith. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1073–1094, 2019.

[LGF+22] Zheng Li, Caili Guo, Zerun Feng, Jenq-Neng Hwang, and Xijun Xue. Multi-view visual semantic embedding. In *International Joint Conferences on Artificial Intelligence*, 2022.

[LHB+21] Ron Levie, Wei Huang, Lorenzo Bucci, Michael Bronstein, and Gitta Kutyniok. Transferability of spectral graph convolutional neural networks. *Journal of Machine Learning Research*, 22(272):1–59, 2021.

[LHJ+21] Yushan Liu, Marcel Hildebrandt, Mitchell Joblin, Martin Ringsquandl, Rime Raissouni, and Volker Tresp. Neural multi-hop reasoning with logical rules on biomedical knowledge graphs. In *European Semantic Web Conference*, pages 375–391. Springer, 2021.

[LHN06] Elizabeth A Leicht, Petter Holme, and Mark EJ Newman. Vertex similarity in networks. *Physical Review E*, 73(2):026120, 2006.

[LHWH20] Hai Liu, Kairong Hu, Fu-Lee Wang, and Tianyong Hao. Aggregating neighborhood information for negative probability sampling for knowledge graph embedding. *Neural Computing & Applications*, 2020.

[LIJ+15] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[LJLB17] Shuxin Liu, Xinsheng Ji, Caixia Liu, and Yi Bai. Extended resource allocation index for link prediction of complex network. *Physica A: Statistical Mechanics and its Applications*, 479:174–183, 2017.

[LJZ09] Linyuan Lü, Ci-Hang Jin, and Tao Zhou. Similarity index based on local paths for link prediction of complex networks. *Physical Review E*, 80(4):046122, 2009.

[LL10] Weiping Liu and Linyuan Lü. Link prediction based on local random walk. *Europhysics Letters*, 89(5):58007, 2010.

[LLS+15] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI Conference on Artificial Intelligence*, 2015.

[LM05] Amy N Langville and Carl D Meyer. A survey of eigenvector methods for web information retrieval. *SIAM Review*, 47(1):135–161, 2005.

[LNK07] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007.

[LPM15] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the Empirical Methods in Natural Language Processing*, pages 1412–1421, 2015.

[LS00] Daniel Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems*, 13, 2000.

[LTCL21] Xiyang Liu, Huobin Tan, Qinghong Chen, and Guangyan Lin. Ragat: Relation aware graph attention network for knowledge graph completion. *IEEE Access*, 9:20840–20849, 2021.

[LW71] Francois Lorrain and Harrison C White. Structural equivalence of individuals in social networks. *The Journal of Mathematical Sociology*, 1(1):49–80, 1971.

[LWF+21] Longjie Li, Hui Wang, Shiyu Fang, Na Shan, and Xiaoyun Chen. A supervised similarity measure for link prediction based on knn. *International Journal of Modern Physics C*, 32(09):2150112, 2021.

[LWS+19] Adam Lerer, Ledell Wu, Jiajun Shen, Timothee Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich. Pytorch-biggraph: A large scale graph embedding system. *Proceedings of Machine Learning and Systems*, 1:120–131, 2019.

[LWZH18] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Adaptive graph convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[LYL+22] Zongwei Liang, Junan Yang, Hui Liu, Keju Huang, Lin Cui, Lingzhi Qu, and Xiang Li. Hrer: A new bottom-up rule learning for knowledge graph completion. *Electronics*, 11(6):908, 2022.

[LYZ18] Yingming Li, Ming Yang, and Zhongfei Zhang. A survey of multi-view representation learning. *IEEE Transactions on Knowledge and Data Engineering*, 31(10):1863–1883, 2018.

[LZ11] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and Its Applications*, 390(6):1150–1170, 2011.

[MBC16] Victor Martinez, Fernando Berzal, and Juan-Carlos Cubero. A survey of link prediction in complex networks. *ACM Computing Surveys*, 49(4):1–33, 2016.

[MBS13] F. Mahdisoltani, J. Biega, and F. M. Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *7th Biennial Conference on Innovative Data Systems Research*, 2013.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*, 2013.

[MCRS19] Christian Meilicke, Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt. Anytime bottom-up rule learning for knowledge graph completion. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3137–3143, 2019.

[ME11] Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 437–452. Springer, 2011.

[MGW+13] Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 777–782, 2013.

[MKS+21] Tareq B Malas, Roman Kudrin, Peter AC Starikov, Dorien JM Peters, Marco Roos, and Kristina M Hettne. Drug repurposing using a semantic knowledge graph. *Data Driven Knowledge Discovery in Polycystic Kidney*, page 75, 2021.

[MLW+20] Roman Martin, Hannah F Löchel, Marius Welzel, Georges Hattab, Anne-Christin Hauschild, and Dominik Heider. Cordite: the curated corona drug interactions database for sars-cov-2. *Iscience*, 23(7):101297, 2020.

[MNN20] Sameh K Mohamed, Vit Novavcek, and Aayah Nounu. Discovering protein drug targets using knowledge graph embeddings. *Bioinformatics*, 36(2):603–610, 2020.

[MNW+18] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. Ray: A distributed framework for emerging AI applications. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 561–577, 2018.

[MR95] Michael Molloy and Bruce Reed. A critical point for random graphs with a given degree sequence. *Random Structures & Algorithms*, 6(2-3):161–180, 1995.

[Mug95] Stephen Muggleton. Inverse entailment and progol. *New Generation Computing*, 13(3):245–286, 1995.

[Nar20] Harsha Holenarasipura Narasanna. Graph convolutional network based friendship recommender system for a social network. KTH, School of Electrical Engineering and Computer Science (EECS), 2020.

[NBS+21] Rajat Nandi, Deep Bhowmik, Rakesh Srivastava, Amresh Prakash, and Diwakar Kumar. Discovering potential inhibitors against sars-cov-2 by targeting nsp13 helicase. *Journal of Biomolecular Structure and Dynamics*, pages 1–13, 2021.

[NDY+21] Joseph A. Newman, Alice Douangamath, Setayesh Yadzani, Yuliana Yosaatmadja, Antony Aimon, José Brandão-Neto, Louise Dunnett, Tyler Gorrie-stone, Rachael Skyner, Daren Fearon, Matthieu Schapira, Frank von Delft, and Opher Gileadi. Structure, mechanism and crystallographic fragment screening of the SARS-CoV-2 NSP13 helicase. *Nature Communications*, 12(1):4848, 2021.

[New06] Mark EJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3):036104, 2006.

[NGJ+19] Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. Industry-scale knowledge graphs: Lessons and challenges: Five diverse technology companies show how it's done. *Queue*, 17(2):48–75, 2019.

[NMTG15] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2015.

[NRP16] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. Holographic embeddings of knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[NSC21] Yan Ling Ng, Cyrill Kafi Salim, and Justin Jang Hann Chu. Drug repurposing for COVID-19: Approaches, challenges and promising candidates. *Pharmacology & Therapeutics*, page 107930, 2021.

[NTK11] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *International Conference on Machine Learning*, 2011.

[NVI21] NVIDIA. Nvidia collective communications library (nccl). https://developer.nvidia.com/nccl, 2021.

[NVNP22] Dai Quoc Nguyen, Thanh Vu, Tu Dinh Nguyen, and Dinh Phung. Quatre: Relation-aware quaternions for knowledge graph embeddings. In *Companion Proceedings of the Web Conference 2022*, pages 189–192, 2022.

[Org22] World Health Organization. WHO recommends two new drugs to treat COVID-19. https://www.who.int/news/item/14-01-2022-who-recommends-two-new-drugs-to-treat-covid-19, 2022. Accessed: 5-9-2022; Published: 14-01-2022.

[OSKK05] Jukka-Pekka Onnela, Jari Saramaki, Janos Kertesz, and Kimmo Kaski. Intensity and coherence of motifs in weighted complex networks. *Physical Review E*, 71(6):065103, 2005.

[OWW21] Pouya Ghiasnezhad Omran, Kewen Wang, and Zhe Wang. An embedding-based approach to rule learning in knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering*, 33(4):1348–1359, 2021.

[Par19] Kyungsoo Park. A review of computational drug repurposing. *Translational and Clinical Pharmacology*, 27(2):59–63, 2019.

[PARS14] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 701–710, 2014.

[Pau17] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8(3):489–508, 2017.

[PB99] Joel L Pomerantz and David Baltimore. Nf-κb activation by a signaling complex containing traf2, tank and tbk1, a novel ikk-related kinase. *The EMBO journal*, 18(23):6694–6704, 1999.

[PBA14] Disha Patel, Joseph D. Bauman, and Eddy Arnold. Advantages of crystallographic fragment screening: Functional and mechanistic insights from a powerful platform for efficient drug discovery. *Progress in Biophysics and Molecular Biology*, 116(0):92–100, 2014.

[Pea05] Karl Pearson. The problem of the random walk. *Nature*, 72(1865):294–294, 1905.

[PI07] Milen Pavlov and Ryutaro Ichise. Finding experts by link prediction in co-authorship networks. In *Proceedings of the Second International ISWC+ASWC Workshop on Finding Experts on the Web with Semantics*, volume 290, pages 42–55. Citeseer, 2007.

[Pla95] Tony A Plate. Holographic reduced representations. *IEEE Transactions on Neural networks*, 6(3):623–641, 1995.

[PLHK22] Eleni Pitsillou, Julia Liang, Andrew Hung, and Tom C Karagiannis. The sars-cov-2 helicase as a target for antiviral therapy: Identification of potential small molecule inhibitors by in silico modelling. *Journal of Molecular Graphics and Modelling*, 114:108193, 2022.

[PLMA+22] Gustavo R Perez-Lemus, Cintia A Menendez, Walter Alvarado, Fabian Bylehn, and Juan J de Pablo. Toward wide-spectrum antivirals against coronaviruses: Molecular characterization of sars-cov-2 nsp13 helicase inhibitors. *Science advances*, 8(1):eabj4526, 2022.

[PY09] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2009.

[PZLH16] L. Pan, T. Zhou, L. Lü, and C. K. Hu. Predicting missing links and identifying spurious links via likelihood analysis. *Scientific Reports*, 6(1):1–10, 2016.

[QCD+20] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1150–1160, 2020.

[Qui90] J. Ross Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3):239–266, 1990.

[RB03] Erzsebet Ravasz and Albert-Laszlo Barabasi. Hierarchical organization in complex networks. *Physical Review E*, 67(2):026112, 2003.

[RBF+21] Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo. Knowledge graph embedding for link prediction: A comparative analysis. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(2):1–49, 2021.

[RDGF16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[RFMT22] Andrea Rossi, Donatella Firmani, Paolo Merialdo, and Tommaso Teofili. Explaining link prediction systems based on knowledge graph embeddings. In *Proceedings of the 2022 International Conference on Management of Data*, pages 2062–2075, 2022.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[RLO+21] Tim GJ Rudner, Cong Lu, Michael A Osborne, Yarin Gal, and Yee Teh. On pathologies in kl-regularized reinforcement learning from expert demonstrations. *Advances in Neural Information Processing Systems*, 34:28376–28389, 2021.

[RM20] Andrea Rossi and Antonio Matinata. Knowledge graph embeddings: Are relation-learning models learning relations? In *EDBT/ICDT Workshops*, 2020.

[RSM+02] Erzsebet Ravasz, Anna Lisa Somera, Dale A Mongru, Zoltan N Oltvai, and A-L Barabasi. Hierarchical organization of modularity in metabolic networks. *Science*, 297(5586):1551–1555, 2002.

[SADW19] Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. Drum: end-to-end differentiable rule mining on knowledge graphs. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 15347–15357, 2019.

[SBBW12] Jack W Scannell, Alex Blanckley, Helen Boldon, and Brian Warrington. Diagnosing the decline in pharmaceutical r&d efficiency. *Nature Reviews Drug discovery*, 11(3):191–200, 2012.

[SDB18] Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in tensorflow. *arXiv preprint arXiv:1802.05799*, 2018.

[SDEW10] Stefan Schoenmackers, Jesse Davis, Oren Etzioni, and Daniel Weld. Learning first-order horn clauses from web text. In *Proceedings of the 2010 Conference on Empirical Methods on Natural Language Processing*, pages 1088–1098, 2010.

[SDG+19] Daniel N Sosa, Alexander Derry, Margaret Guo, Eric Wei, Connor Brinton, and Russ B Altman. A literature-based knowledge graph embedding method for identifying drug repurposing opportunities in rare diseases. In *Pacific Symposium on Biocomputing 2020*, pages 463–474. World Scientific, 2019.

[SDNT18a] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*, 2018.

[SDNT18b] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*, 2018.

[SGT+08] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[SHV19] Guillaume Salha, Romain Hennequin, and Michalis Vazirgiannis. Keep it simple: Graph autoencoders without graph convolutional networks. In *Graph Representation Learning Workshop, Advances in Neural Information Processing Systems*, 2019.

[SHY+17] Qingshuang Sun, Rongjing Hu, Zhao Yang, Yabing Yao, and Fan Yang. An improved link prediction algorithm based on degrees and similarities of nodes. In *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, pages 13–18. IEEE, 2017.

[Sin12] Amit Singhal. Introducing the knowledge graph: things, not strings. https://blog.google/products/search/introducing-knowledge-graph-things-not/, 2012.

[SKB+18] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

[SKBH22] Sola Shirai, Aamod Khatiwada, Debarun Bhattacharjya, and Oktie Hassanzadeh. Rule-based link prediction over event-related causal knowledge in wikidata. In *The 3rd Wikidata Workshop, 21st International Semantic Web Conference*, 2022.

[SKS12] Naoki Shibata, Yuya Kajikawa, and Ichiro Sakata. Link prediction in citation networks. *Journal of the American Society for Information Science and Technology*, 63(1):78–85, 2012.

[SKW07] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the Sixtieth International Conference on World Wide Web*, pages 697–706, 2007.

[SLGA21] Francisco Javier Somolinos, Carlos León, and Sara Guerrero-Aspizua. Drug repurposing using biological networks. *Processes*, 9(6):1057, 2021.

[SM83] Gerard Salton and Michael J McGill. *Introduction to modern information retrieval*. Mcgraw-hill, 1983.

[SMW02] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring isp topologies with rocketfuel. *ACM SIGCOMM Computer Communication Review*, 32(4):133–145, 2002.

[SNM11] Salvatore Scellato, Anastasios Noulas, and Cecilia Mascolo. Exploiting place features in link prediction on location-based social networks. In *Proceedings of the Seventieth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1046–1054, 2011.

[Sor48] Th A Sorensen. A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on danish commons. *Biol. Skar.*, 5:1–34, 1948.

[SQG+22] Nasrullah Sheikh, Xiao Qin, Yaniv Gur, Berthold Reinwald, Qiao Xiang, Haitao Yu, et al. Distributed training of knowledge graph embedding models using ray. In *Twenty-fifth International Conference on Extending Database Technology*, pages 2–549, 2022.

[SQRL22] Nasrullah Sheikh, Xiao Qin, Berthold Reinwald, and Chuan Lei. Scaling knowledge graph embedding models. *arXiv preprint arXiv:2201.02791*, 2022.

[SRM+18] Tommaso Soru, Stefano Ruberto, Diego Moussallem, Andre Valdestilhas, Alexander Bigerl, Edgard Marx, and Diego Esteves. Expeditious generation of knowledge graph embeddings. *arXiv*, 2018.

[SS11] Peter Sanders and Christian Schulz. Engineering multilevel graph partitioning algorithms. In *European Symposium on Algorithms*, pages 469–480. Springer, 2011.

[STDS+08] Michael PH Stumpf, Thomas Thorne, Eric De Silva, Ronald Stewart, Hyeong Jun An, Michael Lappe, and Carsten Wiuf. Estimating the size of the human interactome. *Proceedings of the National Academy of Sciences*, 105(19):6959–6964, 2008.

[SWF+14] Zhesi Shen, Wen-Xu Wang, Ying Fan, Zengru Di, and Ying-Cheng Lai. Reconstructing propagation networks with natural diversity and identifying hidden sources. *Nature Communications*, 5(1):1–10, 2014.

[SZC+01] Heladia Salgado, Alberto S Zavaleta, Socorro G Castro, Dulce M Zarate, Edgar D Peredo, Fabiola S Solano, Ernesto P Rueda, Cesar B Martinez, and Julio C Vides. Regulondb (version 3.2): transcriptional regulation and operon organization in Escherichia coli K-12. *Nucleic Acids Research*, 29(1):72–74, 2001.

[TFP06] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. Fast random walk with restart and its applications. In *Sixth International Conference on Data Mining (ICDM)*, pages 613–622. IEEE, 2006.

[TO13] Sho Tsugawa and Hiroyuki Ohsaki. Effectiveness of link prediction for face-to-face behavioral networks. *Plos One*, 8(12):e81727, 2013.

[Tor12] Nicolas Torzec. The yahoo! knowledge graph. https://research.yahoo.com/publications/9194/yahoo-knowledge-graph, 2012.

[TQW+15] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the Twenty-fourth International Conference on World Wide Web*, pages 1067–1077, 2015.

[TRG05] Rajeev Thakur, Rolf Rabenseifner, and William Gropp. Optimization of collective communication operations in mpich. *The International Journal of High Performance Computing Applications*, 19(1):49–66, 2005.

[Tuc66] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[TWR+16] Theo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080. PMLR, 2016.

[TXZ14] Fei Tan, Yongxiang Xia, and Boyao Zhu. Link prediction in complex networks: a mutual information perspective. *PloS One*, 9(9):e107056, 2014.

[VAKS22] RP Vivek-Ananth, Sankaran Krishnaswamy, and Areejit Samal. Potential phytochemical inhibitors of sars-cov-2 helicase nsp13: a molecular docking and dynamic simulation study. *Molecular Diversity*, 26(1):429–442, 2022.

[VCC+18] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[VMKS+02] C. Von Mering, R. Krause, B. Snel, M. Cornell, S. G. Oliver, S. Fields, and P. Bork. Comparative assessment of large-scale datasets of protein- protein interactions. *Nature*, 417(6887):399–403, 2002.

[WC16] Zhifeng Wu and Yixin Chen. Link prediction using matrix factorization with bagging. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, pages 1–6. IEEE, 2016.

[WCZ16] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the Twenty-second ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1225–1234, 2016.

[WI08] Till Wohlfarth and Ryutaro Ichise. Semantic and event-based approach for link prediction. In *International Conference on Practical Aspects of Knowledge Management*, pages 50–61. Springer, 2008.

[Wil92] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.

[WL68] Boris Weisfeiler and Andrei A Lehman. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16, 1968.

[WLC20] Mark Andrew White, Wei Lin, and Xiaodong Cheng. Discovery of COVID-19 inhibitors targeting the sars-cov-2 nsp13 helicase. *The Journal of Physical Chemistry Letters*, 11(21):9144–9151, 2020.

[WLP18] Peifeng Wang, Shuangyin Li, and Rong Pan. Incorporating GAN for negative sampling in knowledge representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[WLS+22] Xuemei Wei, Yezheng Liu, Jianshan Sun, Yuanchun Jiang, Qifeng Tang, and Kun Yuan. Dual subgraph-based graph neural network for friendship prediction in location-based social networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2022.

[WLWG16] Zhihao Wu, Youfang Lin, Jing Wang, and Steve Gregory. Link prediction with node clustering coefficient. *Physica A: Statistical Mechanics and its Applications*, 452:1–8, 2016.

[WLWJ16] Zhihao Wu, Youfang Lin, Huaiyu Wan, and Waleed Jamil. Predicting top-l missing links with node and link clustering information in large-scale networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2016(8):083202, 2016.

[WMWG17] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[WPC+20] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[WQW21] Meihong Wang, Linling Qiu, and Xiaoli Wang. A survey on knowledge graph embeddings for link prediction. *Symmetry*, 13(3):485, 2021.

[WS98] Duncan J Watts and Steven H Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, 1998.

[WSGG22] Haixia Wu, Chunyao Song, Yao Ge, and Tingjian Ge. Link prediction on complex networks: An experimental survey. *Data Science and Engineering*, pages 1–26, 2022.

[WSP07] Chao Wang, Venu Satuluri, and Srinivasan Parthasarathy. Local probabilistic models for link prediction. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 322–331. IEEE, 2007.

[WV21] Xing Wang and Alexander Vinel. Benchmarking graph neural networks on link prediction. *arXiv preprint arXiv:2102.12557*, 2021.

[WWL+19] Ruijie Wang, Meng Wang, Jun Liu, Weitong Chen, Michael Cochez, and Stefan Decker. Leveraging knowledge graph embeddings for natural language question answering. In *International Conference on Database Systems for Advanced Applications*, pages 659–675. Springer, 2019.

[WWNdS+21] Shen Wang, Xiaokai Wei, Cicero Nogueira Nogueira dos Santos, Zhiguo Wang, Ramesh Nallapati, Andrew Arnold, Bing Xiang, Philip S Yu, and Isabel F Cruz. Mixed-curvature multi-relational graph neural network for knowledge graph completion. In *Proceedings of the Web Conference 2021*, pages 1761–1771, 2021.

[WXWZ15] P. Wang, B. Xu, Y. Wu, and X. Zhou. Link prediction in social networks: the state-of-the-art. *Science China Information Sciences*, 58(1):1–38, 2015.

[WZFC14] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

[XHZ16] Han Xiao, Minlie Huang, and Xiaoyan Zhu. Transg: A generative model for knowledge graph embedding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2316–2325, 2016.

[Xie10] Xing Xie. Potential friend recommendation in online social network. In *2010 IEEE/ACM International Conference on Green Computing and Communications & International Conference on Cyber, Physical and Social Computing*, pages 831–835. IEEE, 2010.

[XPY16] Zhongqi Xu, Cunlai Pu, and Jian Yang. Link prediction based on path entropy. *Physica A: Statistical Mechanics and its Applications*, 456:294–301, 2016.

[XRK+20] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Product knowledge graph embedding for e-commerce. In *Proceedings of the thirteenth International Conference on Web Search and Data Mining*, pages 672–680, 2020.

[Xu18] Chonghuan Xu. A novel recommendation method based on social network using matrix factorization technique. *Information Processing & Management*, 54(3):463–474, 2018.

[YA18] Carl Yang and AWS Amazon. Tree-based allreduce communication on mxnet. *Technical Report*, 2018.

[YHY+21] Qing Ye, Chang-Yu Hsieh, Ziyi Yang, Yu Kang, Jiming Chen, Dongsheng Cao, Shibo He, and Tingjun Hou. A unified drug–target interaction prediction framework based on knowledge graph and recommendation system. *Nature Communications*, 12(1):1–12, 2021.

[YLC15] Yang Yang, Ryan N Lichtenwalter, and Nitesh V Chawla. Evaluating link prediction methods. *Knowledge and Information Systems*, 45(3):751–782, 2015.

[YLY+21] Vincent KC Yan, Xiaodong Li, Xuxiao Ye, Min Ou, Ruibang Luo, Qingpeng Zhang, Bo Tang, Benjamin J Cowling, Ivan Hung, Chung Wah Siu, et al. Drug repurposing for the treatment of COVID-19: A knowledge graph approach. *Advanced Therapeutics*, 4(7):2100055, 2021.

[YQZG22] Yonghong Yu, Weiwen Qian, Li Zhang, and Rong Gao. A graph-neural-network-based social network recommendation algorithm using high-order neighbor information. *Sensors*, 22(19):7122, 2022.

[YTZ+19] Shihui Yang, Jidong Tian, Honglun Zhang, Junchi Yan, Hao He, and Yaohui Jin. Transms: Knowledge graph embedding for complex relations by multidirectional semantics. In *International Joint Conferences on Artificial Intelligence*, pages 1935–1942, 2019.

[YYH+15a] Bishan Yang, Scott Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations*, 2015.

[YYH+15b] Bishan Yang, Scott Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the International Conference on Learning Representations (ICLR) 2015*, 2015.

[ZC17] Muhan Zhang and Yixin Chen. Weisfeiler-lehman neural machine for link prediction. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 575–583, 2017.

[ZC18] M Zhang and Y Chen. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, pages 5165–5175, 2018.

[ZCDZ16] Jian-Xiong Zhang, Duan-Bing Chen, Qiang Dong, and Zhi-Dan Zhao. Identifying a set of influential spreaders in complex networks. *Scientific Reports*, 6(1):1–10, 2016.

[ZCH+20] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[ZCJ+20] Yongjun Zhu, Chao Che, Bo Jin, Ningrui Zhang, Chang Su, and Fei Wang. Knowledge-driven drug repurposing using a comprehensive drug knowledge graph. *Health Informatics Journal*, 26(4):2737–2750, 2020.

[ZCNC18] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[ZCZ+21] Jing Zhang, Bo Chen, Lingxi Zhang, Xirui Ke, and Haipeng Ding. Neural, symbolic and neural-symbolic reasoning on knowledge graphs. *AI Open*, 2:14–35, 2021.

[ZDC+21] Wen Zhang, Shumin Deng, Mingyang Chen, Liang Wang, Qiang Chen, Feiyu Xiong, Xiangwen Liu, and Huajun Chen. Knowledge graph embedding in e-commerce applications: Attentive reasoning, explanations, and transferable rules. In *The 10th International Joint Conference on Knowledge Graphs*, pages 71–79, 2021.

[ZHH+18] Pengfei Zhu, Qi Hu, Qinghua Hu, Changqing Zhang, and Zhizhao Feng. Multi-view label embedding. *Pattern Recognition*, 84:126–135, 2018.

[ZHS+20] Yadi Zhou, Yuan Hou, Jiayu Shen, Yin Huang, William Martin, and Feixiong Cheng. Network-based drug repurposing for novel coronavirus 2019-nCoV/SARS-CoV-2. *Cell Discovery*, 6(1):1–18, 2020.

[ZHS+21] Rui Zhang, Dimitar Hristovski, Dalton Schutte, Andrej Kastrin, Marcelo Fiszman, and Halil Kilicoglu. Drug repurposing for COVID-19 via knowledge graph completion. *Journal of Biomedical Informatics*, 115:103696, 2021.

[ZLW21] Tao Zhou, Yan-Li Lee, and Guannan Wang. Experimental analyses on 2-hop-based and 3-hop-based link prediction algorithms. *Physica A: Statistical Mechanics and Its Applications*, 564:125532, 2021.

[ZLZ09] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71(4):623–630, 2009.

[ZM18] Chenyi Zhuang and Qiang Ma. Dual graph convolutional networks for graph-based semi-supervised classification. In *Proceedings of the 2018 World Wide Web Conference*, pages 499–508, 2018.

[ZQD+20] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

[ZQW+19] Yi Zhao, Meina Qiao, Haiyang Wang, Rui Zhang, Dan Wang, Ke Xu, and Qi Tan. Tdfi: Two-stage deep learning framework for friendship inference via multi-source information. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 1981–1989. IEEE, 2019.

[ZSM+20] Da Zheng, Xiang Song, Chao Ma, Zeyuan Tan, Zihao Ye, Jin Dong, Hao Xiong, Zheng Zhang, and George Karypis. Dgl-ke: Training knowledge graph embeddings at scale. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 739–748, 2020.

[ZTYL19] Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. Quaternion knowledge graph embeddings. *Advances in Neural Information Processing Systems*, 32, 2019.

[ZWLL18] Jiang Zhong, Chen Wang, Qi Li, and Qing Li. A new graph-partitioning algorithm for large-scale knowledge graph. In *International Conference on Advanced Data Mining and Applications*, pages 434–444. Springer, 2018.

[ZWT+20] Yadi Zhou, Fei Wang, Jian Tang, Ruth Nussinov, and Feixiong Cheng. Artificial intelligence in COVID-19 drug repurposing. *The Lancet Digital Health*, 2(12):e667–e676, 2020.

[ZWYX22] Qianjin Zhang, Ronggui Wang, Juan Yang, and Lixia Xue. Structural context-based knowledge graph embedding for link prediction. *Neurocomputing*, 470:109–120, 2022.

[ZXTQ19] Zhaocheng Zhu, Shizhen Xu, Jian Tang, and Meng Qu. Graphvite: A high-performance cpu-gpu hybrid system for node embedding. In *The World Wide Web Conference*, pages 2494–2504, 2019.

[ZYSC19] Yongqi Zhang, Quanming Yao, Yingxia Shao, and Lei Chen. Nscaching: simple and efficient negative sampling for knowledge graph embedding. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 614–625. IEEE, 2019.

[ZYX+22] Chaoyu Zhu, Zhihao Yang, Xiaoqiong Xia, Nan Li, Fan Zhong, and Lei Liu. Multimodal reasoning based on knowledge graph embedding for specific diseases. *Bioinformatics*, 38(8):2235–2245, 2022.

[ZYY+21] Yu Zhang, Jingming Ye, Xin Yue, Sifan Wei, Yu Wang, and Wanli Ren. Research on e-commerce recommended algorithm based on knowledge graph. In *International Conference on Business Intelligence and Information Technology*, pages 78–87. Springer, 2021.

[ZZYY14] Jianpei Zhang, Yuan Zhang, Hailu Yang, and Jing Yang. A link prediction algorithm based on socialized semi-local information. *Journal of Computational Information Systems*, 10(10):4459–4466, 2014.

[ZZZ+20] Zhao Zhang, Fuzhen Zhuang, Hengshu Zhu, Zhiping Shi, Hui Xiong, and Qing He. Relational graph neural network with hierarchical attention for knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9612–9619, 2020.

Appendix A

Additional results for the SNS method

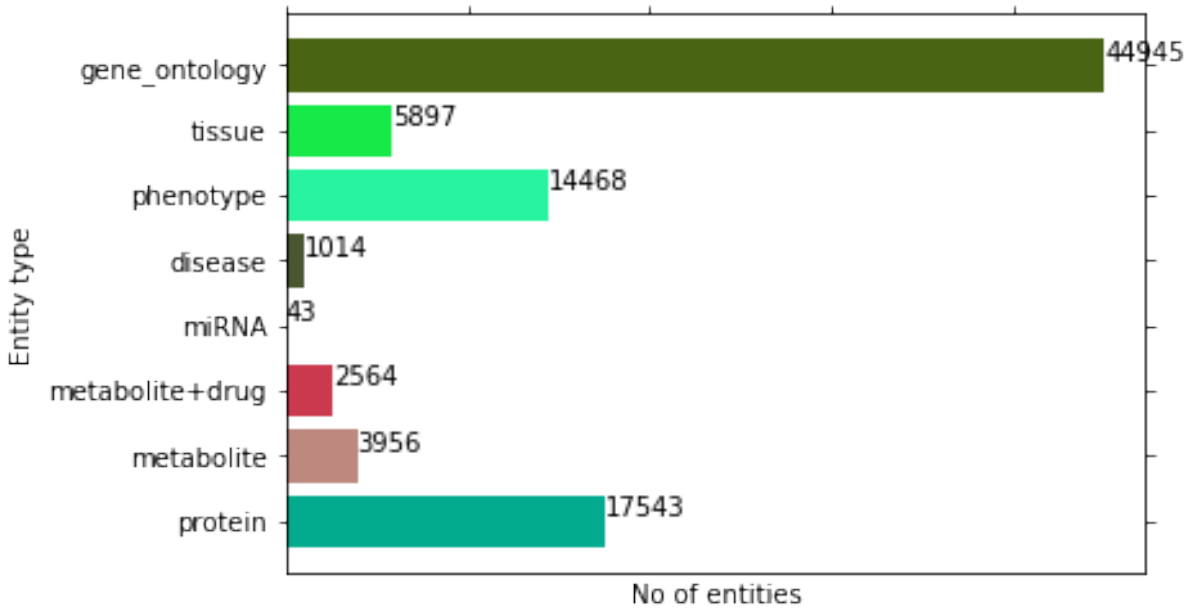


Figure A.1: Type-wise entity count in the FIGHT-HF-23R dataset

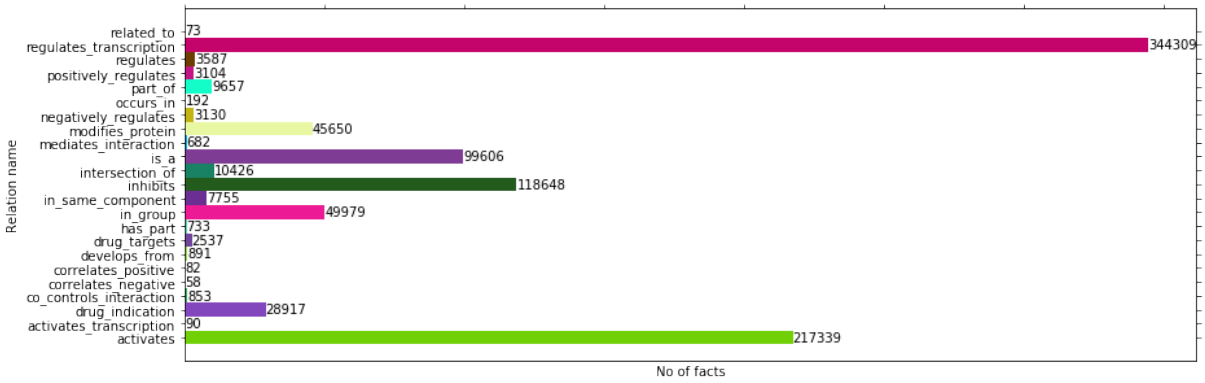


Figure A.2: Relation-wise triple count in the FIGHT-HF-23R dataset

Table A.1: Head (h) and tail (t) entity types for each relation in the FIGHT-HF-23R dataset

	Protein	Metabolite	Metabolite+drug	miRNA	Disease	Phenotype	Tissue	Gene_ontology
activates	h,t	h,t	h,t	t				
activates_transcription	h,t							
drug_indication		h	h		t	t		
co_controls_interaction	h,t	h,t	h,t					
correlates_negative	h,t			h				
correlates_positive	h,t							
develops_from							h,t	
drug_targets	t	h	h					
has_part								h,t
in_group	h,t	h,t	h,t	h,t				
in_same_component	h,t	h,t	h,t	h,t				
inhibits	h,t	h,t	h,t	h,t				
intersection_of								h,t
is_a						h,t	h,t	h,t
mediates_interaction	h,t	h,t	h,t					
modifies_protein	h,t	h,t	t					
negatively_regulates								h,t
occurs_in								h,t
part_of							h,t	h,t
positively_regulates								h,t
regulates								h,t
regulates_transcription	h,t	h,t	h,t					
related_to							h,t	

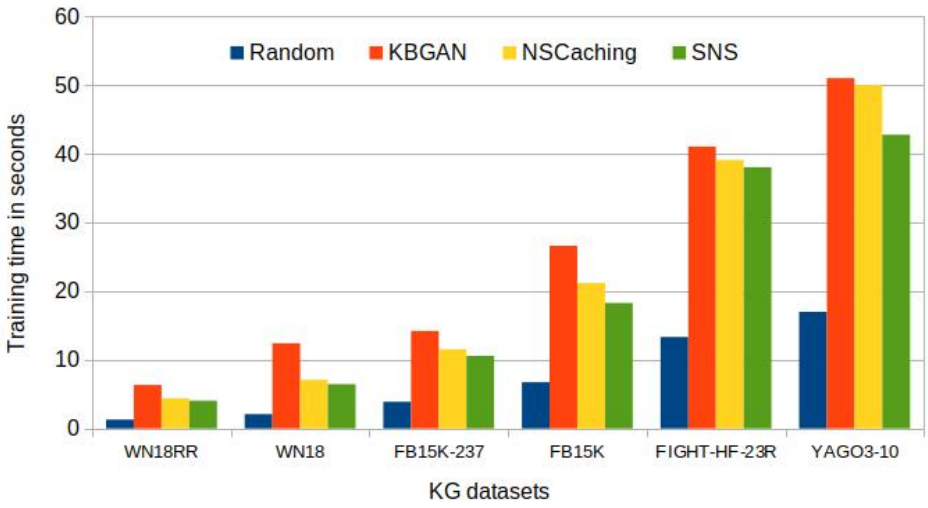
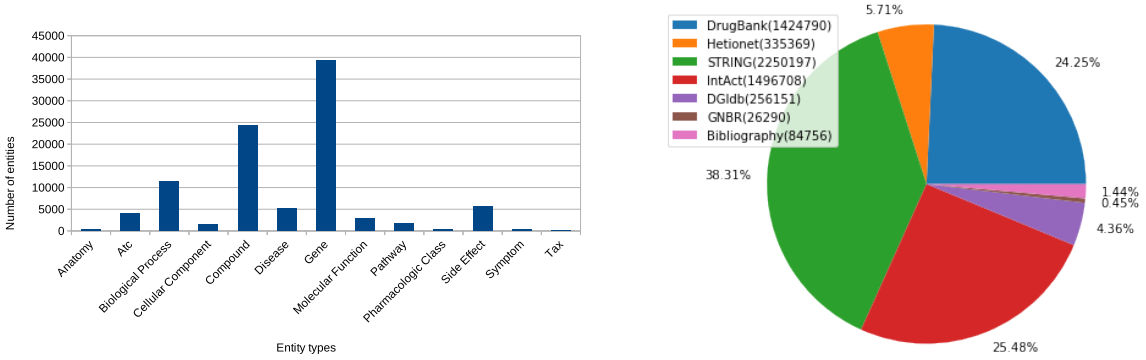


Figure A.3: Training time per epoch for the TransH model with different negative triple sampling methods.

Appendix B

Additional results for the drug repurposing study



(a) Type-wise entity count

(b) Source-wise triple count

Figure B.1: DRKG statistics: (a) Approximately 75% entities are either genes or compounds, (b) Approximately 65% of triples come from two data-source STRING and IntAct. The rest of triples come from other remaining five data-sources.

Table B.1: Data source-wise entity-pair collection

Entity-type pair	**Data Sources**						
↓	Drugbank	GNBR	Hetionet	STRING	IntAct	DGIdb	Bibliography
Gene:Gene		66,722	474,526	1,496,708	254,346		58,629
Compound:Gene	24,801	80, 803	51, 429		1,805	26,290	25,666
Disease:Gene		95,399	27,977				461
Atc:Compound	15,750						
Compound:Compound	1,379,271		6,486				
Compound:Disease	4,968	77,782	1,145				
Gene:Tax		14,663					
Gene:Biological-process			559,504				
Disease:Symptom			3,357				
Anatomy:Disease			3,602				
Disease:Disease			543				
Anatomy:Gene			726,495				
Gene:Molecular-function			97,222				
Compound:Pharma.-class			1,029				
Gene:Cellular-component			73,566				
Gene:Pathway			84,372				
Compound:Side-effect			138,944				

Table B.2: List of 27 disease entities for the COVID-19 disease. They are proteins of the SARS-CoV-2 virus, responsible for COVID-19 disease.

Disease entity	Disease entity	Disease entity
SARS-CoV-2 E	SARS-CoV-2 N	SARS-CoV-2 nsp1
SARS-CoV-2 nsp12	SARS-CoV-2 nsp14	SARS-CoV-2 nsp2
SARS-CoV-2 nsp5	SARS-CoV-2 nsp5 C145A	SARS-CoV-2 nsp7
SARS-CoV-2 nsp9	SARS-CoV-2 orf3a	SARS-CoV-2 orf6
SARS-CoV-2 orf8	SARS-CoV-2 orf9c	SARS-CoV-2 M
SARS-CoV-2 Spike	SARS-CoV-2 nsp10	SARS-CoV-2 nsp13
SARS-CoV-2 nsp15	SARS-CoV-2 nsp4	SARS-CoV-2 nsp11
SARS-CoV-2 nsp6	SARS-CoV-2 nsp8	SARS-CoV-2 orf10
SARS-CoV-2 orf3b	SARS-CoV-2 orf7a	SARS-CoV-2 orf9b

Table B.3: 31 drugs involved in clinical trials (in-trial drugs) for the COVID-19 disease

Drug name	Drug name	Drug name	Drug name
Deferoxamine	Piclidenoson	Losartan	Ibuprofen
Favipiravir	Ruxolitinib	Dexamethasone	Thalidomide
Tranexamic acid	Tocilizumab	Sarilumab	Tradipitant
Angiotensin 1-7	Oseltamivir	Baricitinib	Sargramostim
Chloroquine	Anakinra	Mavrilimumab	Azithromycin
Tetrandrine	Bevacizumab	Tofacitinib	Siltuximab
Nivolumab	Nitric Oxide	Colchicine	Remdesivir
Hydroxychloroquine	Eculizumab	Methylprednisolone	

Table B.4: Details of PDBs for the SARS-CoV-2 nsp13 protein. PDB-IDs denote identifiers of PDB structures. The ligands from PDBs are given with their PubChem database identifiers (Pubchem-IDs), abbreviated (Abbr.) and IUPAC names. The last column gives Root-Mean-Square Deviation (RMSD) values of PDBs based on ATP binding site.

	Ligand			
PDB-ID	**Pubchem-ID**	**Abbr.**	**IUPAC name**	**RMSD (Å)**
7nio	-	-	-	0,000
5rli	118569	JFM	N-(2-phenylethyl)methanesulfonamide	0.496
5rlt	225773	UVJ	3-(2-methyl-1H-benzimidazol-1-yl)propanamide	0.513
5rmk	19786643	O2A	N-methyl-1H-indole-7-carboxamide	0.519
5rlj	94347260	VW4	(2S)-2-phenylpropane-1-sulfonamide	0.520
5rmc	2806372	6SU	methyl 3-(methylsulfonylamino)benzoate	0.523
5rm7	836055	N0E	~N-(4-hydroxyphenyl)-3-phenyl-propanamide	0.533
5rmj	978238	JOV	3-chloro-N-(1-hydroxy-2-methylpropan-2-yl)benzamide	0.534
5rl9	63648807	UR7	1-(3-fluoro-4-methylphenyl)methanesulfonamide	0.537
5rl8	4693938	VVG	N-(2-fluorophenyl)ethanesulfonamide	0.541
5rm0	22882465	S7G	~N-[(3 R)-1,2,3,4-tetrahydroquinolin-3-yl]ethanamide	0.543
5rlh	2777223	K2P	2-(trifluoromethoxy)benzoic acid	0.544
5rma	694486	JHJ	N-(4-methoxyphenyl)-N'-pyridin-4-ylurea	0.544
5rm5	51114235	NUA	N-(1-ethyl-1H-pyrazol-4-yl)cyclobutanecarboxamide	0.545
5rlp	8120202	VWA	(1S)-1-(4-fluorophenyl)-N-methylethan-1-amine	0.549
5rlk	71757215	NYV	1-(propan-2-yl)-1H-imidazole-4-sulfonamide	0.550
5rlc	31390	VVM	4-amino-N-phenylbenzene-1-sulfonamide	0.558
5rme	863558	RYM	4-(benzimidazol-1-ylmethyl)benzenecarbonitrile	0.561
5rlo	43539927	UQS	N-[(2-fluorophenyl)methyl]-1H-pyrazol-4-amine	0.563
5rm6	33381566	HR5	~N-(cyclobutylmethyl)-1,5-dimethyl-pyrazole-4-carboxamide	0.570
5rm2	330448	UXG	1-(diphenylmethyl)azetidin-3-ol	0.573
5rmg	91650	MUK	4,6-dimethyl- N-phenyl-pyrimidin-2-amine	0.574
5rmm	155387694	VXG	(3S,4R)-1-acetyl-4-phenylpyrrolidine-3-carboxylic acid	0.580
5rm3	68423044	S7J	2-(trifluoromethyl)pyrimidine-5-carboxamide	0.582
5rlz	124505295	VWM	(3R)-1-acetyl-3-hydroxypiperidine-3-carboxylic acid	0.584
5rlm	71757930	VW7	N-(8-methyl-1,2,3,4-tetrahydroquinolin-5-yl)acetamide	0.588
5rm9	934518	EJQ	~N-(4-fluorophenyl)-2-pyrrolidin-1-yl-ethanamide	0.591
5rld	910270	VVY	2-phenoxy-1-(pyrrolidin-1-yl)ethan-1-one	0.599
5rly	57116692	K34	5-(1,3-thiazol-2-yl)-1H-1,2,4-triazole	0.603
7nng	43363487	UJK	1-(2-methylphenyl)-1,2,3-triazole-4-carboxylic acid	0.616
5rlr	12684717	VWD	(1R)-2-(methylsulfonyl)-1-phenylethan-1-ol	0.623
5rl6	828139	LJA	N-[3-(carbamoylamino)phenyl]acetamide	0.632
5rlg	43082989	VW1	(2S)-2-(4-cyanophenoxy)propanamide	0.642
5rmd	961974	VWY	N-ethyl-4-[(methylsulfonyl)amino]benzamide	0.656
5rm1	2760997	RY4	N-[4-(aminomethyl)phenyl]methanesulfonamide	0.699
5rlu	4291023	JG4	2-(thiophen-2-yl)-1H-imidazole	1.184
5rlf	16226828	NY7	N-(2-methoxy-5-methylphenyl)glycinamide	1.205
5rln	45792228	NZG	3-(acetylamino)-4-fluorobenzoic acid	1.230
5rmh	96462663	VX4	[(4S)-4-methylazepan-1-yl](1,3-thiazol-4-yl)methanone	1.256
5rmf	588246	NX7	(2,6-difluorophenyl)(pyrrolidin-1-yl)methanone	1.273
5rlv	16653131	VWJ	N-(propan-2-yl)-1H-benzimidazol-2-amine	1.273
5rlb	53516552	VVJ	N-cycloheptyl-N-methylmethanesulfonamide	1.283
5rl7	19261749	VVD	5-(acetylamino)-2-fluorobenzoic acid	1.288
5rmb	2763377	VWV	ethyl (1,1-dioxo-1lambda 6 ,4-thiazinan-4-yl)acetate	1.289
5rls	148031	VWG	N-hydroxyquinoline-2-carboxamide	1.304
5rlq	14196351	UVA	N-methyl-2-(methylsulfonyl)aniline	1.313
5rml	2517697	VXD	N-(3-chloro-2-methylphenyl)glycinamide	1.316
5rlw	1530178	S9S	~N-[2-(4-fluorophenyl)ethyl]methanesulfonamide	1.341
5rmi	686921	STV	~N-(1,3-benzodioxol-5-ylmethyl)ethanesulfonamide	1.349
			Average RMSD	**0.777**

Table B.5: List of residues in nsp13 structure interacting with the listed ligands (predicted ligands are highlighted in light cyan, the others correspond to known ligands). Ligands are ranked as in Table 4.4. Residue label and position are in bold when they correspond to residues delineating the ATP binding site. One-letter residue names are used.

Ligands	**E261**	**N265**	P283	G285	**G287**	**K288**	**S289**	**H290**	A313	A316	**K320**	**K323**	**E375**	D401	**Q404**	L438	G439	T440	**R442**	R443	Q537	G538	E540	K569
Diosmin	X		X	X	X	X	X	X			X						X	X	X	X		X	X	
Fosinopril				X	X	X	X		X	X	X		X						X				X	
Ergotamine				X		X	X	X			X		X						X	X		X		X
Eprosartan						X	X		X		X			X	X									
Macitentan								X			X	X							X					
Dinoprostone		X		X	X	X	X	X			X					X			X					
Risperdal						X	X	X			X		X								X	X	X	

Table B.6: Top-100 ranked *(Compound, COVID-19 target)* pairs with the *Treat* relation.

Rank	Compound	Disease-Target	Rank	Compound	Disease-Target
1	Compound::DB01234	Disease::SARS-CoV2-nsp6	51	Compound::DB01117	Disease::SARS-CoV2-orf6
2	Compound::DB00959	Disease::SARS-CoV2-nsp6	52	Compound::DB00380	Disease::SARS-CoV2-nsp5_C145A
3	Compound::DB08877	Disease::SARS-CoV2-nsp13	53	Compound::DB01177	Disease::SARS-CoV2-nsp6
4	Compound::DB01394	Disease::SARS-CoV2-nsp6	54	Compound::DB00884	Disease::SARS-CoV2-nsp13
5	Compound::DB01041	Disease::SARS-CoV2-nsp5_C145A	55	Compound::DB00496	Disease::SARS-CoV2-nsp13
6	Compound::DB00608	Disease::SARS-CoV2-nsp5_C145A	56	Compound::DB01586	Disease::SARS-CoV2-nsp6
7	Compound::DB00207	Disease::SARS-CoV2-nsp6	57	Compound::DB00490	Disease::SARS-CoV2-nsp5_C145A
8	Compound::DB00678	Disease::SARS-CoV2-nsp13	58	Compound::DB01396	Disease::SARS-CoV2-nsp5_C145A
9	Compound::DB11817	Disease::SARS-CoV2-nsp5_C145A	59	Compound::DB00305	Disease::SARS-CoV2-nsp6
10	Compound::DB01611	Disease::SARS-CoV2-nsp5_C145A	60	Compound::DB01004	Disease::SARS-CoV2-orf9c
11	Compound::DB09421	Disease::SARS-CoV2-nsp13	61	Compound::DB01212	Disease::SARS-CoV2-nsp5_C145A
12	Compound::DB06402	Disease::SARS-CoV2-nsp13	62	Compound::DB00158	Disease::SARS-CoV2-nsp6
13	Compound::DB00777	Disease::SARS-CoV2-nsp14	63	Compound::DB11915	Disease::SARS-CoV2-nsp5_C145A
14	Compound::DB00557	Disease::SARS-CoV2-nsp13	64	Compound::DB00995	Disease::SARS-CoV2-nsp5_C145A
15	Compound::DB00224	Disease::SARS-CoV2-nsp5_C145A	65	Compound::DB00477	Disease::SARS-CoV2-nsp5_C145A
16	Compound::DB00607	Disease::SARS-CoV2-nsp5_C145A	66	Compound::DB08811	Disease::SARS-CoV2-nsp13
17	Compound::DB04794	Disease::SARS-CoV2-nsp13	67	Compound::DB00486	Disease::SARS-CoV2-nsp14
18	Compound::DB05990	Disease::SARS-CoV2-nsp13	68	Compound::DB00958	Disease::SARS-CoV2-nsp6
19	Compound::DB00737	Disease::SARS-CoV2-nsp13	69	Compound::DB00743	Disease::SARS-CoV2-nsp5_C145A
20	Compound::DB00227	Disease::SARS-CoV2-nsp6	70	Compound::DB00850	Disease::SARS-CoV2-nsp13
21	Compound::DB00956	Disease::SARS-CoV2-nsp13	71	Compound::DB00933	Disease::SARS-CoV2-nsp13
22	Compound::DB00199	Disease::SARS-CoV2-nsp13	72	Compound::DB00876	Disease::SARS-CoV2-nsp13
23	Compound::DB09075	Disease::SARS-CoV2-nsp5_C145A	73	Compound::DB04841	Disease::SARS-CoV2-nsp13
24	Compound::DB01192	Disease::SARS-CoV2-nsp13	74	Compound::DB00749	Disease::SARS-CoV2-nsp5_C145A
25	Compound::DB01260	Disease::SARS-CoV2-nsp13	75	Compound::DB01183	Disease::SARS-CoV2-nsp13
26	Compound::DB01196	Disease::SARS-CoV2-orf6	76	Compound::DB11967	Disease::SARS-CoV2-nsp13
27	Compound::DB00527	Disease::SARS-CoV2-nsp13	77	Compound::DB00834	Disease::SARS-CoV2-orf9c
28	Compound::DB04552	Disease::SARS-CoV2-nsp13	78	Compound::DB01073	Disease::SARS-CoV2-nsp5_C145A
29	Compound::DB00492	Disease::SARS-CoV2-nsp13	79	Compound::DB00991	Disease::SARS-CoV2-nsp14
30	Compound::DB00993	Disease::SARS-CoV2-nsp6	80	Compound::DB00759	Disease::SARS-CoV2-nsp6
31	Compound::DB00867	Disease::SARS-CoV2-nsp13	81	Compound::DB01021	Disease::SARS-CoV2-nsp5_C145A
32	Compound::DB06233	Disease::SARS-CoV2-nsp13	82	Compound::DB00409	Disease::SARS-CoV2-nsp13
33	Compound::DB00434	Disease::SARS-CoV2-nsp14	83	Compound::DB00555	Disease::SARS-CoV2-nsp13
34	Compound::DB01032	Disease::SARS-CoV2-orf6	84	Compound::DB04844	Disease::SARS-CoV2-nsp13
35	Compound::DB00620	Disease::SARS-CoV2-orf9c	85	Compound::DB01232	Disease::SARS-CoV2-nsp6
36	Compound::DB01167	Disease::SARS-CoV2-nsp6	86	Compound::DB00857	Disease::SARS-CoV2-nsp5_C145A
37	Compound::DB00400	Disease::SARS-CoV2-nsp13	87	Compound::DB01608	Disease::SARS-CoV2-nsp14
38	Compound::DB00917	Disease::SARS-CoV2-nsp13	88	Compound::DB00218	Disease::SARS-CoV2-nsp13
39	Compound::DB00776	Disease::SARS-CoV2-nsp13	89	Compound::DB00408	Disease::SARS-CoV2-nsp14
40	Compound::DB06285	Disease::SARS-CoV2-nsp13	90	Compound::DB00800	Disease::SARS-CoV2-nsp5_C145A
41	Compound::DB01062	Disease::SARS-CoV2-nsp13	91	Compound::DB00927	Disease::SARS-CoV2-nsp13
42	Compound::DB00722	Disease::SARS-CoV2-nsp14	92	Compound::DB00585	Disease::SARS-CoV2-nsp6
43	Compound::DB04743	Disease::SARS-CoV2-nsp6	93	Compound::DB00906	Disease::SARS-CoV2-nsp13
44	Compound::DB00395	Disease::SARS-CoV2-nsp13	94	Compound::DB01009	Disease::SARS-CoV2-nsp6
45	Compound::DB01219	Disease::SARS-CoV2-nsp14	95	Compound::DB01193	Disease::SARS-CoV2-nsp5_C145A
46	Compound::DB00717	Disease::SARS-CoV2-orf9c	96	Compound::DB00495	Disease::SARS-CoV2-orf9c
47	Compound::DB00432	Disease::SARS-CoV2-nsp6	97	Compound::DB01109	Disease::SARS-CoV2-nsp6
48	Compound::DB00697	Disease::SARS-CoV2-nsp13	98	Compound::DB00773	Disease::SARS-CoV2-orf9b
49	Compound::DB08932	Disease::SARS-CoV2-nsp13	99	Compound::DB00471	Disease::SARS-CoV2-nsp6
50	Compound::DB01233	Disease::SARS-CoV2-nsp14	100	Compound::DB01406	Disease::SARS-CoV2-orf6

Table B.7: List of all docked ligands ranked according to decreasing Gold Score. The prediction values in DRKG are also indicated with the best disease target and corresponding probability score. When the best target is not nsp13, the second best target and its probability score are indicated only if it is nsp13.

PubChemID	Name	MW	In-KG?	D-rank	D-Score	Pr-target	Probability	Pr-rank	Label	Ref-PMID
5281613	Diosmin	608.5	Yes	1	79.04	nsp9	0.162	6091	Literature	33052685
55891	Fosinopril	563.7	Yes	2	78.86	nsp13	0.964	29	Predicted	This Study
644241	Nilotinib	529.5	Yes	3	76.55	nsp6/nsp13	0.860/ 0.796	101	Literature	33052685
44517748	Chromone-4c	417.4	No	4	76.17				Literature	34995115
439153	Dpnh	665.4	Yes	5	76.04	nsp9	0.146	6361	Literature	33052685
2882	Cromolyn	468.4	Yes	6	75.7	nsp13	0.619	1751	Literature	33052685
5320557	Picrasidine_N	490.5	No	7	74.55				Literature	34117992
5320555	Picrasidine_M	490.5	No	8	74.52				Literature	34117992
10531	Dihydroergotamine	583.7	Yes	9	74.13	nsp13	0.796	475	Literature	33052685
8223	Ergotamine	581.7	Yes	10	72.83	nsp13	0.805	361	Literature	33052685
24873435	Simeprevir	749.9	Yes	11	72.68	nsp6/nsp13	0.537/ 0.139	2076	Literature	32875166
16004692	Macitentan	588.3	Yes	12	71.76	nsp13	0.950	49	Predicted	This Study
5281037	Eprosartan	424.5	Yes	13	71.33	nsp13	0.878	72	Predicted	This Study
5280360	Dinoprostone	352.5	Yes	14	70.76	nsp13	0.959	38	Predicted	This Study
44187	Cefoperazone	645.7	Yes	15	70.24	nsp13	0.599	1816	Literature	33052685
185617	Scutellarin	462.4	No	16	70.19				Literature	32875166
60838	Irinotecan	586.7	Yes	17	70.11	nsp6/ nsp13	0.824/ 0.796	249	Literature	33052685
45110509	Paritaprevir	765.9	Yes	18	69.61	orf8	0.099	6990	Literature	32875166
5073	Risperdal	410.5	Yes	19	68.76	nsp6/ nsp13	0.807/ 0.807	349	Literature	33052685
107715	Ergoloid	611.7	Yes	20	68.59	nsp13	0.584	1888	Literature	33052685
11965427	Lifitegrast	615.5	Yes	21	68.53	nsp6	0.265	4002	Literature	33052685
42611257	Zelboraf	489.9	Yes	22	68.45	nsp13	0.795	743	Literature	33052685
11520894	Ridaforolimus	990.2	Yes	23	68.36	nsp13	0.963	32	Predicted	This Study
3081362	Telavancin	1755.6	Yes	24	68.06	nsp13	0.977	12	Predicted	This Study
3658	Hydroxyzine	374.9	Yes	25	67.88	nsp13	0.977	14	Predicted	
636405	Cefpiramide	612.6	Yes	26	67.26	orf9c	0.191	5531	Literature	33052685
444031	Darifenacin	426.5	Yes	27	64.62	nsp13	0.937	55	Predicted	This Study
3025	Cinchocaine	343.5	Yes	28	62.08	nsp13	0.971	27	Predicted	This Study
135413536	Emend	534.4	Yes	29	61.94	nsp6/ nsp13	0.794/ 0.761	794	Literature	33052685
4634	Oxybutynin	357.5	Yes	30	60.87	nsp13	0.955	41	Predicted	This Study
491941	Pritelivir	402.5	Yes	31	59.9	nsp9	0.002	8022	Literature	34455933
6451149	Netupitant	578.6	Yes	32	59.57	nsp5_C145A/nsp13	0.705/ 0.54	1417	Literature	33052685
941361	Flunarizine	404.5	Yes	33	59.16	nsp13	0.878	73	Predicted	This Study
33572	Ritodrine	287.35	Yes	34	59.07	nsp13	0.963	31	Predicted	This Study
489948	Epiexcelsin	414.4	No	35	58.85				Literature	34117992
4748	Perphenazine	404	Yes	36	58.77	nsp13	0.878	70	Predicted	This Study
10288191	Binimetinib	441.2	Yes	37	58.46	nsp13	0.878	76	Predicted	This Study
16678941	Lumacaftor	452.4	Yes	38	58.21	nsp15/ nsp13	0.535/ 0.07	2085	Literature	33052685
5702160	Famotidine	337.5	Yes	39	58.21	nsp13	0.869	91	Predicted	This Study
44603531	Grazoprevir	766.9	Yes	40	57.91	nsp14	0.227	4782	Literature	32875166
60648	Tiagabine	375.6	Yes	41	56.92	nsp13	0.869	93	Predicted	This Study
4034	Meclizine	390.9	Yes	42	56.71	nsp13	0.975	19	Predicted	This Study
32281	Protirelin	362.38	Yes	43	56.06	nsp13	0.982	11	Predicted	This Study
5317297	Euphorbetin	354.3	No	44	56.02				Literature	34117992
10206	Cepharanthine	606.7	No	45	54.77				Literature	33052685
4078	Mesoridazine	386.6	Yes	46	54.67	nsp13	0.878	71	Predicted	This Study
2807230	SSYA10-001	308.4	No	47	53.95				Literature	32875166
5281672	Myricetin	318.23	Yes	48	53.89	orf9b	0.018	7798	Literature	32875166
60164	Differin	412.5	Yes	49	53.77	nsp4/ nsp13	0.678/ 0.160	1521	Literature	33052685
2378	Bifonazole	310.4	Yes	50	51.46	nsp13	0.975	17	Predicted	This Study
863558	RYM	233.27	No	51	51.31				PDB	
3001028	Enjuvia	350.4	No	52	51.23				Literature	33052685
1530178	S9S	217.26	No	53	50.99				PDB	
447715	Obeticholic_acid	420.6	Yes	54	50.94	nsp13	0.975	18	Predicted	This Study
225773	UVJ	203.24	No	55	50.93				PDB	
5502	Tofisopam	382.5	Yes	56	49.15	nsp13	0.901	66	Predicted	This Study
5311066	Desonide	416.5	Yes	57	49.05	nsp13	0.971	25	Predicted	This Study
155387694	VXG	233.26	No	58	49				PDB	
6303	Cordycepin	251.24	Yes	59	48.83	nsp13	0.143	6394	Literature	34455933
6000	Tubocurarine	609.7	Yes	60	48.8	nsp13	0.796	566	Literature	33052685
54477	Remoxipride	371.27	Yes	61	48.72	nsp13	0.874	82	Predicted	This Study
152946	Moxifloxacin	401.4	Yes	62	48.65	nsp13	0.869	88	Predicted	This Study
31390	VVM	248.3	No	63	48.48				PDB	
836055	N0E	241.28	No	64	48.01				PDB	
12560	Erythromycin	733.9	Yes	65	47.75	nsp13	0.973	22	Predicted	This Study
910270	VVY	205.25	No	66	47.68				PDB	

Table B.7 continued.

PubChemID	Name	MW	In-KG?	D-rank	D-Score	Pr-target	Probability	Pr-rank	Label	Ref-PMID
43363487	UJK	203.2	No	67	47.58				PDB	
4488	Niflumic_acid	282.22	Yes	68	47.39	nsp13		28	Predicted	This Study
12304371	Isorhoeadine	383.4	No	69	45.4				Literature	34117992
2576	Carisoprodol	260.33	Yes	70	45.31	nsp13	0.950	44	Predicted	This Study
961974	VWY	242.3	No	71	45.13				PDB	
6018	Tetrabenazine	317.4	Yes	72	44.99	nsp13	0.874	84	Predicted	This Study
5284596	Naloxone	327.4	Yes	73	44.9	nsp13	0.878	75	Predicted	This Study
59053860	Bananin	327.29	No	74	44.62				Literature	34995115
5284604	Oxymorphone	301.34	Yes	75	44.42	nsp13	0.971	24	Predicted	This Study
330448	UXG	239.31	No	76	44.36				PDB	
694486	JHJ	243.26	No	77	44.23				PDB	
43082989	VW1	190.2	No	78	44.23				PDB	
16226828	NY7	194.23	No	79	43.49				PDB	
91650	MUK	199.25	No	80	43.37				PDB	
8120202	VWA	153.2	No	81	43.27				PDB	
686921	STV	243.28	No	82	43.02				PDB	
978238	JOV	227.69	No	83	42.98				PDB	
148031	VWG	188.18	No	84	42.62				PDB	
43539927	UQS	191.2	No	85	42.47				PDB	
22882465	S7G	190.24	No	86	42.43				PDB	
2760997	RY4	200.26	No	87	42.26				PDB	
51114235	NUA	193.25	No	88	41.98				PDB	
34312	Oxcarbazepine	252.27	Yes	89	41.94	nsp13	0.959	39	Predicted	This Study
71757930	VW7	204.27	No	90	41.53				PDB	
12684717	VWD	200.26	No	91	41.45				PDB	
441140	Griseofulvin	352.8	Yes	92	41.42	nsp13	0.959	37	Predicted	This Study
5284569	Hydrocodone	299.4	Yes	93	41.27	nsp13	0.973	21	Predicted	This Study
118569	JFM	199.27	No	94	40.91				PDB	
934518	EJQ	222.26	No	95	40.91				PDB	
94347260	VW4	199.27	No	96	40.75				PDB	
45792228	NZG	197.16	No	97	40.72				PDB	
71757215	NYV	189.24	No	98	40.47				PDB	
828139	LJA	193.2	No	99	40.4				PDB	
33381566	HR5	207.27	No	100	40.34				PDB	
19786643	O2A	174.2	No	101	40.25				PDB	
2806372	6SU	229.26	No	102	40				PDB	
14196351	UVA	185.25	No	103	39.98				PDB	
19261749	VVD	197.16	No	104	39.96				PDB	
588246	NX7	211.21	No	105	39.94				PDB	
4693938	VVG	203.24	No	106	39.78				PDB	
63648807	UR7	203.24	No	107	39.76				PDB	
5487	Tizanidine	253.71	Yes	108	39.39	nsp13	0.950	48	Predicted	This Study
16653131	VWJ	175.23	No	109	39.38				PDB	
96462663	VX4	224.32	No	110	39.3				PDB	
53516552	VVJ	205.32	No	111	39.2				PDB	
68423044	S7J	191.11	No	112	39.13				PDB	
5245	Risedronic_acid	283.11	Yes	113	38.25	nsp13	0.937	54	Predicted	This Study
2763377	VWV	221.28	No	114	38.05				PDB	
5280980	Clavulanic_acid	199.16	No	115	37.73				Literature	35462185
3878	Lamotrigine	256.09	Yes	116	37.63	nsp13	0.874	83	Predicted	This Study
2517697	VXD	198.65	No	117	36.98				PDB	
1738	Homovanillic_acid	182.17	No	118	36.76				Literature	35462185
124505295	VWM	187.19	No	119	36.58				PDB	
2777223	K2P	206.12	No	120	35.2				PDB	
4291023	JG4	150.2	No	121	35.15				PDB	
12035	Acetylcysteine	163.2	No	122	30.2				Literature	35462185
57116692	K34	152.18	No	123	30.12				PDB	

Appendix C

Gradient sharing in the Ring-AllReduce architecture

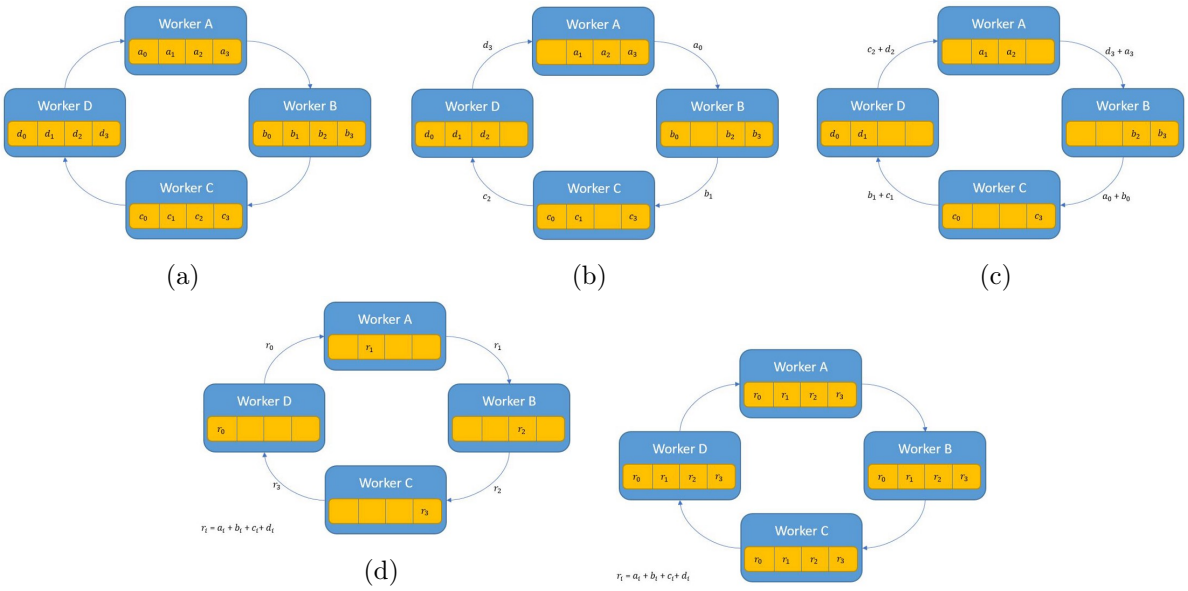


Figure C.1: Illustration of gradient sharing among four workers in a Ring-AllReduce [SDB18] architecture. The array of gradients in each worker are highlighted in yellow background. The arrows show the orientation of communication between two workers. Each worker sends its gradients to the next worker and gradients are summed for computing final gradients.

Résumé

De nombreux systèmes complexes du monde réel peuvent être représentés par des graphes, où les nœuds représentent des entités et les liens des relations entre les paires de nœuds. La prédiction de liens (LP) est l'un des problèmes les plus intéressants et les plus anciens dans le domaine de l'exploration de graphes ; elle prédit la probabilité d'un lien entre deux nœuds non connectés. Cette thèse étudie le problème LP dans les graphes simples et les graphes de connaissances (KGs). La première partie de cette thèse se concentre sur le problème LP dans les graphes simples. Dans la première étude, des approches basées sur la similarité et sur l'encastrement sont évaluées et comparées sur des graphes simples de différents domaines. L'étude a également identifié la difficulté de fixer le seuil du score de similarité pour calculer la métrique de précision des approches basées sur la similarité et a proposé une nouvelle méthode pour calculer la métrique. Les résultats ont montré la supériorité attendue des approches basées sur l'intégration. Cependant, chaque approche basée sur la similarité s'est avérée compétitive sur des graphes aux propriétés spécifiques. Nous avons pu vérifier expérimentalement que les approches basées sur la similarité sont explicables mais manquent de généralisation, tandis que les approches basées sur l'encastrement sont générales mais non explicables. La deuxième étude tente de surmonter la limitation de l'inexplicabilité des approches basées sur l'encastrement en découvrant des connexions intéressantes entre elles et les approches basées sur la similarité. La troisième étude démontre comment les approches basées sur la similarité peuvent être assemblées pour concevoir une approche LP supervisée explicable. Il est intéressant de noter que l'étude montre des performances LP élevées pour l'approche supervisée sur différents graphes, ce qui est très satisfaisant.

La deuxième partie de la thèse se concentre sur les LP dans les KGs. Un KG est représenté comme une collection de triplets RDF, (head,relation,tail) où les entités head et tail sont reliées par une relation spécifique. Le problème de LP dans un KG est formulé comme la prédiction de la tête ou de la queue manquante dans un triplet. La LP basée sur l'incorporation de KG est devenue très populaire ces dernières années, et la génération de triplets négatifs est une tâche importante dans les méthodes d'incorporation. La quatrième étude traite d'une nouvelle méthode appelée SNS pour générer des triplets négatifs de haute qualité. Nos résultats montrent une meilleure performance LP lorsque SNS est utilisé que lorsque d'autres méthodes d'échantillonnage négatif sont utilisées. La deuxième étude traite d'une nouvelle méthode d'extraction de règles neuro-symboliques et d'une stratégie d'abduction pour expliquer les LP par une approche basée sur l'intégration en utilisant les règles apprises. La troisième étude applique notre LP explicable pour développer une nouvelle approche de repositionnement des médicaments pour COVID-19. L'approche apprend un ensemble d'enchâssements d'entités et de relations dans un KG centré sur COVID-19 pour obtenir un meilleur enchâssement des éléments du graphe. Pour la première fois à notre connaissance, des méthodes de criblage virtuel sont ensuite utilisées pour évaluer les prédictions obtenues à l'aide des embeddings. L'évaluation moléculaire et les chemins explicatifs apportent de la fiabilité aux résultats de prédiction et sont de nouvelles méthodes complémentaires et réutilisables pour mieux évaluer les molécules proposées pour le repositionnement. La dernière étude propose une architecture distribuée pour l'apprentissage des KG embeddings dans des environnements distribués et parallèles. Les résultats révèlent que l'apprentissage dans l'environnement distribué proposé, par rapport à un apprentissage centralisé, réduit considérablement le temps de calcul des méthodes d'incorporation KG sans affecter les

performances des LP.

Mots-clés: graphe simple, graphe de connaissance, prédiction de lien, plongement de graphe, embedding de graphe, ensemble d'embeddings, explications, génération de triplets négatifs, repositionnement de médicaments.

Abstract

Many real-world complex systems can be well-represented with graphs, where nodes represent objects or entities and links/relations represent interactions between pairs of nodes. Link prediction (LP) is one of the most interesting and long-standing problems in the field of graph mining; it predicts the probability of a link between two unconnected nodes based on available information in the current graph. This thesis studies the LP problem in graphs. It consists of two parts: LP in simple graphs and LP knowledge graphs (KGs). In the first part, the LP problem is defined as predicting the probability of a link between a pair of nodes in a simple graph. In the first study, a few similarity-based and embedding-based LP approaches are evaluated and compared on simple graphs from various domains. he study also criticizes the traditional way of computing the precision metric of similarity-based approaches as the computation faces the difficulty of tuning the threshold for deciding the link existence based on the similarity score. We proposed a new way of computing the precision metric. The results showed the expected superiority of embedding-based approaches. Still, each of the similarity-based approaches is competitive on graphs with specific properties. We could check experimentally that similarity-based approaches are fully explainable but lack generalization due to their heuristic nature, whereas embedding-based approaches are general but not explainable. The second study tries to alleviate the unexplainability limitation of embedding-based approaches by uncovering interesting connections between them and similarity-based approaches to get an idea of what is learned in embedding-based approaches. The third study demonstrates how the similarity-based approaches can be ensembled to design an explainable supervised LP approach. Interestingly, the study shows high LP performance for the supervised approach across various graphs, which is competitive with embedding-based approaches.

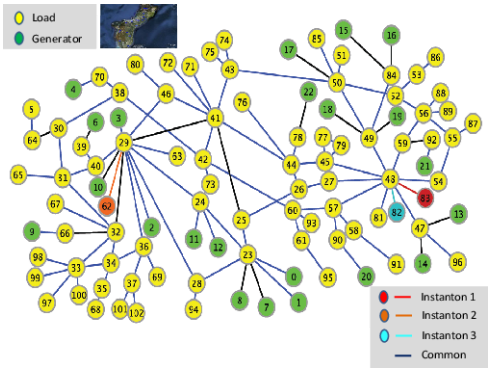
The second part of the thesis focuses on LP in KGs. A KG is represented as a collection of RDF triples, (head,relation,tail) where the head and the tail are two entities which are connected by a specific relation. The LP problem in a KG is formulated as predicting missing head or tail entities in a triple. LP approaches based on the embeddings of entities and relations of a KG have become very popular in recent years, and generating negative triples is an important task in KG embedding methods. The first study in this part discusses a new method called SNS to generate high-quality negative triples during the training of embedding methods for learning embeddings of KGs. The results we produced show better LP performance when SNS is injected into an embedding approach than when injecting state-of-the-art negative triple sampling methods. The second study in the second part discusses a new neuro-symbolic method of mining rules and an abduction strategy to explain LP by an embedding-based approach utilizing the learned rules. The third study applies the explainable LP to a COVID-19 KG to develop a new drug repurposing approach for COVID-19. The approach learns "ensemble embeddings" of entities and relations in a COVID-19 centric KG, in order to get a better latent representation of the graph elements. For the first time to our knowledge, molecular docking is then used to evaluate the predictions obtained from drug repurposing using KG embedding. Molecular evaluation and explanatory paths bring reliability to prediction results and constitute new complementary and reusable methods for assessing KG-based drug repurposing. The last study proposes a distributed architecture for learning KG embeddings in distributed and parallel settings. The results of the study that the computational time of embedding methods improves remarkably without affecting LP performance when they are trained in the proposed distributed settings than the traditional centralized settings.

Keywords: simple graph, knowledge graph, link prediction, graph embedding, graph embed-

ding, set of embeddings, explanations, generation of negative triples, drug repositioning.

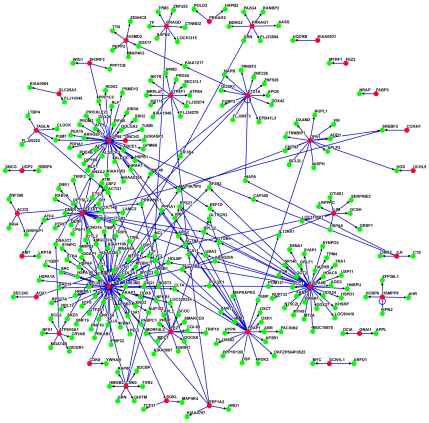
Le Résumé Étendu

Aujourd'hui, plusieurs systèmes complexes nous entourent, comme les réseaux électriques, les réseaux de transport, les réseaux de communication et les systèmes biologiques. L'étude de ces systèmes est intrinsèquement difficile en raison des dépendances, des relations ou d'autres types d'interactions entre leurs composants. Pour analyser ces réseaux, nous avons besoin de représentations structurées de ces systèmes. Les graphes sont des structures de données omniprésentes qui peuvent être utilisées pour décrire tout système complexe. Bien que le concept de graphe a été utilisé pour la première fois par le célèbre mathématicien suisse Leonhard Euler en 1735 pour résoudre le problème du pont de Königsberg, le terme "graphe" a été utilisé pour la première fois par le mathématicien anglais Sylvester en 1878 [BLW86]. Dans sa forme la plus générale, un graphe est constitué d'un ensemble de noeuds représentant des objets ou des entités et d'un ensemble de liens représentant les relations entre les paires d'entités. À titre d'illustration, un système complexe de réseau électrique peut être représenté par un graphe qui considère les dispositifs de transmission d'énergie (par exemple, les générateurs, les transformateurs, les relais) comme des noeuds et les lignes de transmission comme des liens (Figure C.2a). Les graphes offrent plus que de simples belles représentations. Ils fournissent également une base mathématique à partir de laquelle nous pouvons analyser, comprendre et extraire des connaissances sur les systèmes. Compte tenu des potentialités des graphes, plusieurs chercheurs ont étudié de nombreux graphes du monde réel, allant de graphes de réseaux sociaux simples à des graphes biologiques complexes. La Figure C.2 illustre quelques exemples de graphes. Les graphes peuvent encoder de riches informations sur les noeuds et leurs relations complexes. Les liens dans un graphe peuvent être dirigés, c'est-à-dire qu'il n'est possible de se déplacer que du premier au deuxième noeud, ou non dirigés, c'est-à-dire qu'il est également possible de se déplacer du deuxième au premier noeud. Les graphes peuvent contenir des boucles, lorsque les noeuds se relient entre eux, et des liens multiples, lorsque deux noeuds sont reliés par plusieurs liens dirigés distincts. Les graphes sans boucles, sans liens non dirigés et sans multiliens sont appelés graphes simples. Par exemple, le graphe d'amitié (Figure C.2b), où les noeuds sont des personnes, les liens sont des amitiés entre des couples de personnes ; le graphe PPI (Figure C.2c), où les nœuds sont des protéines, et les liens sont des interactions entre des paires de protéines. Les graphes peuvent contenir des informations plus descriptives sur les noeuds, comme les types, les attributs et aussi sur les liens, comme les noms des liens. Ces types de graphes sont généralement connus sous le nom de graphes de connaissances (Figure C.2d). Les directions et les noms sont deux exigences obligatoires des liens dans les graphes de connaissances (KG). Ces graphes supportent des liens multiples entre une paire de nœuds ainsi que des boucles. Par conséquent, les graphes de connaissances encodent plus d'informations sémantiques que les graphes simples. La plupart des graphes du monde réel sont de grande taille. Par exemple, le graphe HI-III PPI [KLS+19] est constitué de 18 000 protéines humaines et de 35 500 interactions protéine-protéine ; le graphe de connaissances Freebase [Goo13] contient près de 44 millions de nœuds représentant des entités de divers domaines (par exemple, des personnes, des lieux, des religions, des entreprises) et 2,4 milliards de liens représentant des relations nommées entre des paires d'entités. Cependant, ces graphes sont loin d'être complets. Le lieu de naissance et la nationalité sont manquants pour environ 78,5 % et 93,8 % des personnes, respectivement, dans le graphe de connaissances Freebase [MGW+13]. Le problème de la prédiction des liens manquants ou de l'inférence de nouveaux liens dans un graphe est communément appelé le problème de prédiction de liens [LZ11]. Les chercheurs ont développé plusieurs approches pour la prédiction de liens dans les graphes, qui calculent la plausibilité d'un lien entre deux nœuds non connectés dans un graphe. Formellement, la prédiction de liens est la tâche de prédire la probabilité d'un lien entre deux noeuds en

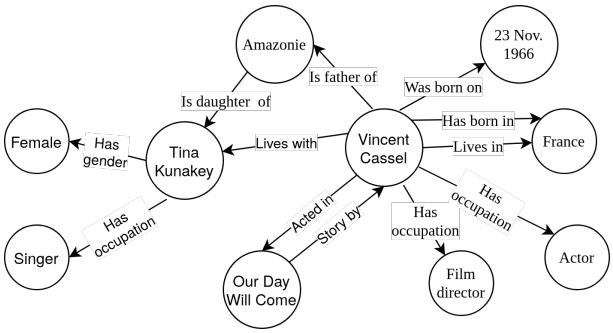


(a) Un graphique de puissance de l'île de Guam [CPS10]

(b) Un graphe social



(c) Un graphique PPI [BHC+09]



(d) Un exemple de graphe de connaissances

Figure C.2: Exemples de graphes. (a) Dans le graphe de l'énergie de l'île de Guam, les noeuds sont différents types de dispositifs d'énergie, et les liens sont des connexions physiques entre des paires de dispositifs ; (b) dans le graphe de l'amitié, un noeud représente une personne, et un lien représente l'amitié entre un couple de personnes ; (c) dans le graphe PPI, les noeuds sont des protéines humaines, et les liens représentent des interactions physiques entre des paires de protéines ; (d) dans l'exemple jouet de graphe de connaissance, les noeuds sont différents types d'entités (par ex. personne, lieu, film), et les liens représentent des relations nommées entre des paires d'entités.

se basant sur les informations topologiques et/ou attributaires disponibles d'un graphe [XPY16]. Les approches de prédiction de liens permettent de découvrir des relations complexes dans un graphe, ce qui nous aide à prendre de meilleures décisions. Par exemple, la tâche de reprogrammation de médicaments peut être formulée comme une tâche de prédiction de liens dans un graphe biologique où les liens non observés entre les nœuds de médicaments (approuvés/essayés cliniquement) et de maladies sont prédits [SLGA21, FCFP21, KS21]. Les experts peuvent ensuite procéder à la validation expérimentale de ces paires médicament-maladie prédites afin de trouver des médicaments efficaces pour la maladie en question.

Le développement d'une approche de prédiction de liens pour les graphes soulève deux défis majeurs. Le premier défi est de fournir des performances de prédiction élevées sur des graphes provenant de différents domaines. Le second défi est de fournir une explication suffisante des résultats pour apporter de la fiabilité à l'approche.

Il existe de nombreuses approches dans la littérature pour la prédiction de liens dans des graphes simples, allant de l'heuristique basée sur la similarité des noeuds aux approches plus récentes basées sur l'apprentissage de représentation, appelées approches basées sur l'intégration. Les approches basées sur l'encastrement apprennent des représentations vectorielles de faible dimension, également connues sous le nom d'encastrements, des noeuds du graphe et utilisent ensuite les représentations vectorielles pour calculer la plausibilité des liens non observés dans le graphe [CWPZ18, ZCH$^+$20, KSSB20]. Des études montrent que les approches basées sur l'encastrement offrent de meilleures performances de prédiction de liens dans des graphes de différents domaines que les approches basées sur la similarité. Cependant, une préoccupation majeure et sérieuse avec ces approches est leur manque d'explicabilité. Les approches classiques basées sur l'apprentissage supervisé pourraient constituer une solution potentielle dans ce cas, car elles sont explicables. Cependant, elles nécessitent la définition manuelle d'un bon ensemble de caractéristiques et dépendent souvent d'attributs externes des nœuds qui ne sont pas disponibles dans de nombreux graphes, tels que les graphes sociaux, les graphes de commerce électronique, en raison du problème de confidentialité des données [MBC16]. Dans ce contexte, l'état de l'art soulève deux questions : (1) Comment expliquer la prédiction de liens basée sur l'intégration ? et (2) Comment définir un bon ensemble de caractéristiques sans utiliser les attributs des nœuds pour la prédiction de liens basée sur l'apprentissage supervisé ? La prédiction de liens dans les graphes de connaissances est plus difficile que dans les graphes simples, car les graphes de connaissances contiennent des relations complexes entre les noeuds. Les approches de prédiction de liens pour les graphes de connaissances sont soit basées sur des règles logiques, soit basées sur l'intégration. Des études montrent que les approches de prédiction de liens basées sur l'intégration offrent de meilleures performances de prédiction et une meilleure évolutivité que les approches traditionnelles basées sur des règles. Cependant, les approches basées sur l'intégration ne sont pas explicables. En fait, les approches de prédiction de liens basées sur l'intégration dans les graphes de connaissances soulèvent deux nouveaux défis en plus de la performance de prédiction élevée et de l'explicabilité. Le premier est la génération d'exemples négatifs de haute qualité, car ils ne sont pas facilement disponibles. Le second est l'évolutivité des approches de prédiction de liens, car les graphes de connaissances du monde réel sont de taille énorme. Bien qu'il existe de nombreuses approches pour la prédiction de liens dans les graphes, il est remarquable que les approches existantes ne sont pas assez bonnes pour offrir des solutions satisfaisantes. Dans ce contexte, l'état de l'art de la prédiction de liens dans les graphes de connaissances appelle trois directions de recherche : (1) la génération d'exemples négatifs de haute qualité pour les approches basées sur l'intégration ; (2) l'explication de la prédiction de liens basée sur l'intégration ; et (3) la mise en oeuvre des approches de prédiction de liens basées sur l'intégration dans un environnement distribué.

Cette thèse étudie les méthodes de prédiction de liens explicables dans les graphes et leur application aux graphes de biomolécules. Concrètement, les principales contributions de ce travail de recherche sont les suivantes :

1. **Explication de la prédiction de liens basée sur le plongement dans des graphes simples**

   Les approches basées sur la similarité et celles basées sur l'intégration sont deux catégories populaires pour la prédiction de liens dans les graphes simples. Les approches de la première catégorie sont explicables mais ne sont performantes que dans les graphes présentant des caractéristiques spécifiques, tandis que les approches de la seconde catégorie offrent des performances de prédiction élevées dans la plupart des graphes mais ne sont pas explicables. Tout d'abord, nous décrivons la difficulté de calculer les scores de précision des approches

basées sur la similarité et nous présentons une stratégie pour surmonter cette difficulté. Nous comparons les approches basées sur la similarité aux approches basées sur l'intégration sur plusieurs graphes de référence avec différentes propriétés provenant de divers domaines. Au-delà de la comparaison intra et inter-catégorie des performances des approches, nous proposons une nouvelle méthode pour découvrir des connexions intéressantes entre les approches basées sur l'encastrement et celles basées sur la similarité, afin d'atténuer le problème bien connu de boîte noire des approches basées sur l'encastrement. Les résultats montrent des connexions intéressantes qui donnent des idées sur les heuristiques apprises dans les approches basées sur l'encastrement.

2. **Prédiction supervisée des liens dans les graphes simples**

   La prédiction de liens basée sur les méthodes classiques d'apprentissage automatique supervisé est moins explorée en raison de l'indisponibilité des informations sur les attributs des nœuds et des difficultés à définir un bon ensemble de caractéristiques. Cependant, la prédiction supervisée des liens pourrait être une bonne solution lorsqu'un bon ensemble de caractéristiques est disponible, car ils sont explicables comme les approches basées sur la similarité et très performants comme les approches basées sur l'intégration. Considérant le fait que différentes heuristiques basées sur la similarité se concentrent sur différents aspects structurels d'un graphe, nous définissons un riche ensemble de caractéristiques structurelles où chaque caractéristique représente une métrique de similarité. Nous avons conçu une approche de prédiction de liens par apprentissage supervisé basée sur cet ensemble de caractéristiques. Les résultats montrent que notre approche de prédiction de liens supervisée offre des performances de prédiction comparables à celles des approches récentes basées sur l'intégration. À des fins d'explication, nous calculons le score d'importance de chaque métrique/caractéristique de similarité afin d'identifier les métriques dominantes qui contribuent à générer des liens dans un graphe.

3. **Échantillonnage négatif simple (SNS) pour la prédiction de liens dans les graphes de connaissances**

   Au cours de l'apprentissage d'enchâssements efficaces, l'échantillonnage de triplets négatifs a été mis en évidence comme une étape importante, car les KGs ne contiennent que des triplets positifs. Nous présentons une nouvelle méthode efficace d'échantillonnage négatif simple (SNS) pour échantillonner des triplets négatifs de haute qualité dans les KGs. SNS est générale et injectable à n'importe quelle méthode géométrique d'intégration de KGs pour échantillonner les triples négatifs. Les résultats montrent que la méthode SNS améliore les performances de prédiction des méthodes d'intégration de KG dans de nombreux ensembles de données de graphes de connaissances et surpasse les méthodes d'échantillonnage existantes.

4. **Explication de la prédiction des liens dans les graphes de connaissances**

   L'applicabilité des approches de prédiction de liens basées sur les enchâssements aux graphes de connaissances est souvent limitée en raison de leur incapacité à expliquer les prédictions. Nous avons développé une nouvelle méthode pour apprendre des règles de haute qualité basées sur un graphe de connaissances et ses enchâssements appris. Ensuite, nous utilisons une stratégie abductive pour expliquer les prédictions par la méthode d'encastrement basée sur l'instanciation des règles apprises pour trouver des chemins explicatifs pour les prédictions. Les résultats expérimentaux montrent que notre stratégie

d'explication est capable d'expliquer de nombreux triplets prédits par une méthode de prédiction de liens basée sur l'intégration dans les graphes de connaissances.

5. **Reprogrammation de médicaments explicables et validés au niveau moléculaire pour COVID-19**

   Comme chaque méthode d'intégration de graphe de connaissances a ses propres points forts et points faibles dans l'apprentissage des intégrations, nous générons des intégrations d'ensemble de haute qualité de graphes de connaissances basées sur les intégrations apprises par des méthodes d'intégration complémentaires. Nous présentons une nouvelle approche de réadaptation des médicaments basée sur les intégrations d'ensemble d'un graphe de connaissances biologiques. L'approche formule le problème de prédiction de liens comme la prédiction de médicaments reproductibles pour une maladie. Nous appliquons cette approche à un graphe de connaissances centré sur COVID-19. Les résultats montrent que notre approche offre de meilleures performances de prédiction. Nous appliquons également notre approche de prédiction de liens explicables pour fournir des explications sur nos prédictions. Afin d'augmenter la fiabilité de nos prédictions, nous intégrons également une évaluation moléculaire pour valider davantage nos prédictions. Il s'agit de la première tentative de combiner une approche basée sur l'intégration d'un graphe de connaissances avec l'approche établie de criblage virtuel pour la réadaptation des médicaments.

6. **Une plateforme distribuée pour l'apprentissage de plongement de graphes de connaissances avec Ray**

   Les méthodes traditionnelles de plongement de graphes de connaissances apprennent les plongements de manière centralisée (une seule machine). Mais la plupart des graphes de connaissances du monde réel ont une taille énorme, et une seule machine n'est pas suffisante pour stocker de tels graphes de connaissances et en apprendre les incorporations. Nous proposons un nouveau cadre distribué pour l'apprentissage d'imbrications de graphes de connaissances d'une manière entièrement distribuée et parallèle. Nous utilisons l'environnement Ray pour implémenter notre cadre. Les résultats montrent que notre plateforme distribuée pour l'apprentissage des méthodes de plongement est adaptable aux graphes de connaissances à très grande échelle et à la taille du cluster utilisé tout en préservant les performances de prédiction des liens.

Bien que ce travail de thèse fournisse des résultats satisfaisants pour la prédiction de liens explicables dans les graphes simples et les graphes de connaissances, nous identifions quelques limitations ainsi que des perspectives intéressantes à explorer (afin d'améliorer l'impact de notre travail) : (1) réglage fin des hyperparamètres, (2) vérification de la robustesse des modèles d'encastrement sur les KG de grande taille, (3) étude de la transférabilité des modèles d'intégration KG appris, (4) étude de plongement de graphe de connaissances multivues, (5) étude d'une nouvelle méthode de partitionnement des graphes de connaissances.