Università degli Studi di Milano
FACOLTÀ DI SCIENZE E TECNOLOGIE

DIPARTIMENTO DI INFORMATICA
GIOVANNI DEGLI ANTONI

CORSO DI LAUREA MAGISTRALE IN
INFORMATICA

ON THE EVALUATION OF HETEROGENEOUS LINK PREDICTION METHODS ON AN RNA-CENTERED KNOWLEDGE GRAPH

Supervisor: Prof. Marco Mesiti
Co-supervisor: Dott. Emanuele Cavalleri

Candidate:
Mayank Pratap Singh
Student ID: 14383A

Academic Year 2023-2024

Questo lavoro è dedicato ai miei genitori

"What I cannot create, I do not understand"

– Richard Feynman

"It's not only powerful,
but it's also inadequate"

– Miller Puckette

Acknowledgments

I would like to express my heartfelt gratitude to my supervisor, **Prof. Marco Mesiti**, whose unwavering support and guidance were instrumental throughout every stage of my thesis. His expertise, insightful feedback, and dedication significantly shaped the outcome of this work. He provided me tremendous help in guiding the research work with brilliant ideas and open critics. I greatly appreciate all the efforts he made to make me understand the scientific and computational background of the research work.

I am also indebted to the entire team at **AnacletoLab** for their collaborative spirit, valuable discussions, and the stimulating research environment they provided throughout this journey.

Special thanks go to **Dott. Emanuele Cavalleri**, my co-supervisor, for his constructive criticism and valuable suggestions, which greatly contributed to improving the quality of my thesis.

Lastly, I extend my heartfelt gratitude to my family, whose unwavering support and motivation have been the cornerstone of my academic journey. Special appreciation goes to my sister, **Rupanshi Bhadouria**, who has been a pillar of strength throughout this process. Her encouragement has been a driving force behind my perseverance and dedication.

The completion of this thesis stands as a testament to the collective efforts of all these individuals, for which I am sincerely grateful.

Contents

Chapter 1

Introduction

The "RNA world" has emerged as a crucial area of study in modern biology and the swift progress in RNA-based technologies has greatly transformed biomedical research, offering new perspectives on fundamental biological processes and leading to important discoveries in gene expression, disease mechanisms, and therapeutic treatments and it also open new avenues for the development of targeted therapeutics, tailored to individual biomolecular characteristics.

In order to address this challenge, RNA-KG was developed, which is a comprehensive knowledge graph that aggregates biological knowledge about RNAs from multiple public databases. It integrates functional relationships between RNAs, genes, proteins, and chemicals along with ontologically grounded biomedical concepts.

A central focus of this research is the application of link prediction techniques to RNA-KG and to provide a detailed comparison of different models used on an RNA-centric knowledge graph. Our focus is on extracting node embeddings and assessing them using metrics such as precision, recall, and accuracy. Furthermore, link prediction performance is also assessed for node type classification. These evaluations highlight the efficiency of each model in predicting node types and uncovering meaningful RNA-related interactions.

Our findings offer insights into model performance for RNA-KG link prediction and enhance its utility as a resource for biomedical research, enabling the discovery of novel RNA-related interactions and contributing to the development of personalized medicine.

The thesis is divided into five chapters. The first chapter introduces RNA-KG and outlines the research objectives, motivation, and contributions of the thesis. The second chapter focuses on the construction of RNA-KG, including data integration, ontological alignment, and generation of RNA-specific views. The third chapter delves into graph representation learning techniques for homogeneous and heterogeneous graphs and also compares models such as DistMult, ComplEx, GCN, and R-GCN for link prediction and

node classification. Chapter four describes the experimental setup, evaluates the embedding models on RNA-KG, and analyzes their performance using different metrics. Finally, the concluding chapter summarizes key findings, limitations, and proposes future directions to enhance RNA-KG and its applications in biomedical research.

Chapter 2

RNA-KG: A Knowledge Graph tailored for RNA Molecules

The study of RNA has emerged as a crucial frontier for the study of fundamental biological processes and human diseases and is paving the way for the development of new drugs tailored to each patient's biomolecular characteristics. RNA molecules, including both coding and non-coding RNAs, play diverse roles in cellular function, and these molecules can lead to a significant breakthrough in treatment of cancer, genetic disorders, neurodegenerative conditions, cardiovascular and infections diseases. The recent success of RNA-based technologies, such as mRNA vaccines, highlights their therapeutic potential.

However, despite the vast amount of scientific data on coding and non-coding RNAs, it remains scattered across multiples public databases, and a centralized, uniform, and semantically consistent representation of the "RNA world" is still lacking. The extraction and integration of information from different data sources for conducting knowledge discovery activity requires a lot of effort from researchers. To address these issues, knowledge graphs (KGs) have emerged as a powerful solution for organizing interrelated information across various domains, enabling the integration of heterogeneous data to uncover complex interdependencies and hidden relationships. RNA-KG is a KG that centralizes biological knowledge about RNAs.

2.1 RNA

RNA (ribonucleic acid) is a polymeric molecule essential for numerous biological processes, particularly in translating genetic information from DNA into proteins, which are necessary for various cellular functions. As illustrated in Figure 1, RNA consists of a sugar-phosphate backbone [1] and nucleobases (adenine, cytosine, guanine, and uracil) [2]. These nucleobases pair up to form codons, three-base sequences that code for specific amino acids during protein synthesis.

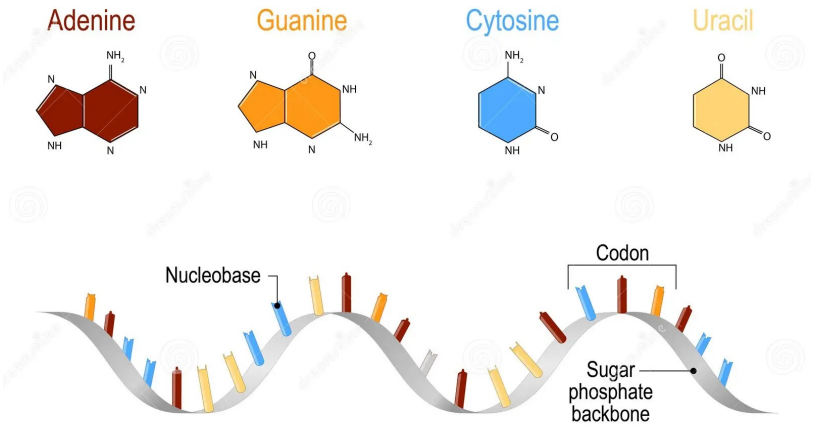


Figure 1: Structure of RNA.

RNA molecules play a fundamental role in cell biology, performing a wide range of functions either *i*) directly by regulating gene expression [3], exhibiting enzymatic activity [4], through the modification or regulation of other RNAs or other bio-molecules [5], or *ii*) indirectly by being translated into proteins [6].

Figure 2 illustrates the main classes of RNAs. **Messenger RNA (mRNA)** [7], a coding RNA, carries genetic information from DNA to ribosomes for protein synthesis. In contrast, non-coding RNAs include **ribosomal RNA (rRNA)** [8], which forms part of the ribosome, and **transfer RNA (tRNA)** [9], which delivers amino acids during translation. Other non-coding RNAs include **MicroRNAs (miRNAs)** [10], which regulate gene expression, and **small nuclear RNAs (snRNAs)**, which are involved in RNA splicing [11]. **Small nucleolar RNAs (snoRNAs)** guide chemical modifications of rRNA [12], while **long non-coding RNAs (lncRNAs)** regulate transcription and chromatin organization [13]. Lastly, **catalytic RNAs (ribozymes)** function as enzymatically active molecules in biochemical reactions [14].

2.2 Knowledge Graphs (KGs)

There is no standard definition for the concept of Knowledge Graph (KG).

From a structural perspective, a KG can be represented as a graph (N, E, N_T, E_T), where N is the set of typed nodes representing real-world entities (the available types are contained in N_T). The set E represents the typed edges between nodes, i.e. $E \subseteq N \times E_T \times N$, where E_T represents the predicate that can exist among entities in the considered domain. A triple $(s, p, o) \in E$ represents the existence of the relationship/predicate p between a subject s and an object o. Figure 3 provides an example of KG in the biomedical domain [15].

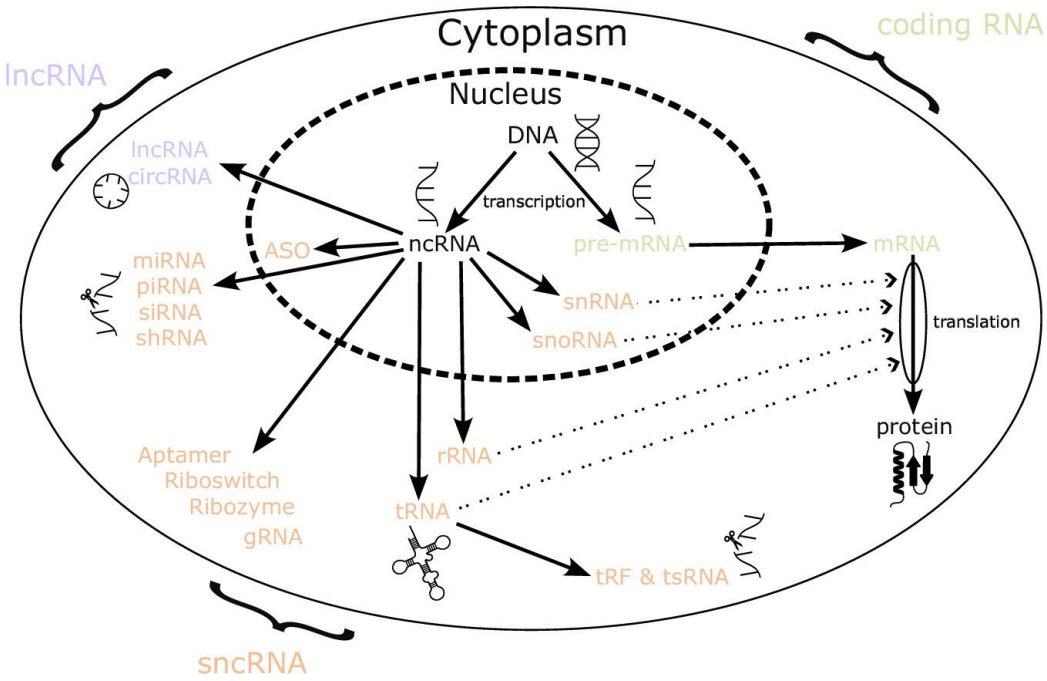


Figure 2: Schematic representation of the RNA network within a cell.

Edges and nodes are typed according to their semantics: green nodes represent drugs, the yellow node is a disease, and brown nodes represent genes. Properties are not displayed for the sake of visualization but can be added to both edges and nodes.

In the context of the knowledge representation, a KG can be considered a pair $\langle T, A \rangle$, where T represents the TBox and A represents the ABox [16]. The TBox defines the taxonomy of the specific domain the KG will be about, considering classes, properties/relationships, and assertions that are generally accepted within that domain. The ABox contains the attributes and roles of instances of a class, and assertions about the membership of the instance to classes of the TBox. Figure 4 provides an example of modelling knowledge. (a) The TBox includes classes (i.e., "Gene", "DNA sequence", and "Cell nucleus"), properties (i.e., "located in" and "is a"), and the assertions between classes (i.e., "Gene is a DNA sequence" and "Gene located in Cell nucleus"). (b) The ABox includes instances of classes (i.e., "Endothelin receptor type B") represented in the TBox and assertions about those instances (i.e., "Endothelin receptor type B, instance of, Gene" and "Endothelin receptor type B, causes, ABCD syndrome").

Both definitions are widely accepted, but one may be preferred over the other depending on the context. For example, the former is more commonly used in link prediction tasks, while the latter is more common when building or reasoning about a KG.

Figure 3: Example of biomedical KG.

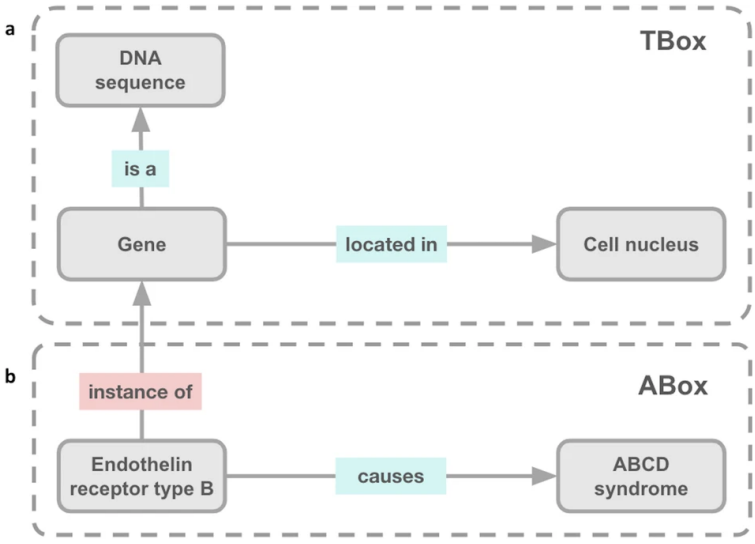


Figure 4: Example of knowledge modelling within a KG.

2.3 RNA-KG construction

RNA-KG [17] is a knowledge graph specifically designed to represent and integrate biological knowledge about RNA molecules, including both coding and non-coding RNAs. RNA-KG includes triples extracted from over 60 public databases, providing a centralized and semantically consistent framework for understanding the interactions and relationships involving RNA, genes, proteins, and other biomedical concepts. It contains 673,825 nodes and 12,692,212 edges, representing various biological entities and their interactions. The Knowledge graph can be queried using SPARQL [18]. The construction of RNA-KG followed four phases that can be organized in the workflow reported in Figure 5.

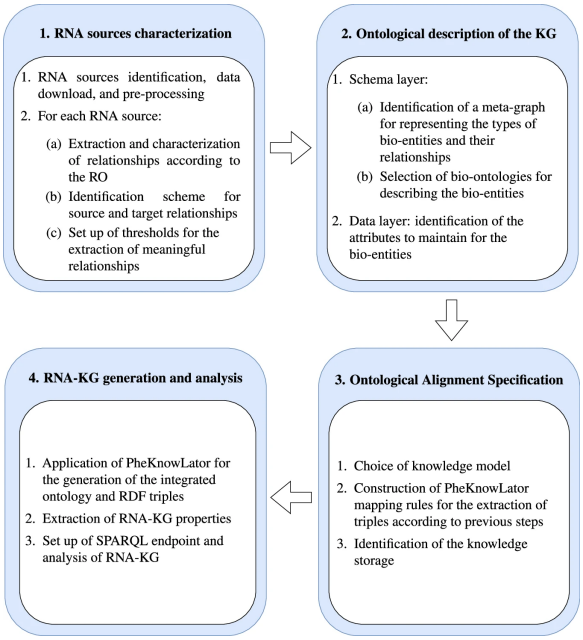


Figure 5: Workflow for the construction of RNA-KG.

Starting from the relationships stored in the considered data sources, a schema (named `RNA-KG meta-graph` [19]) that outlines the relationships among various bio-entities was created. Figure 6 depicts the proposed meta-graph. It provides both direct and inverse relationships that are considered to guarantee bi-directional navigation of the generated KG. To simplify the visualization of the meta-graph, most of the non-RNA bio-entities that are known to play an important role in studying the biology (and supporting the discovery) of novel RNA drugs were omitted. Then, the PheKnowLator [16] tool was used for aligning identifiers to eleven well-reputed bio-ontologies and generating the KG.

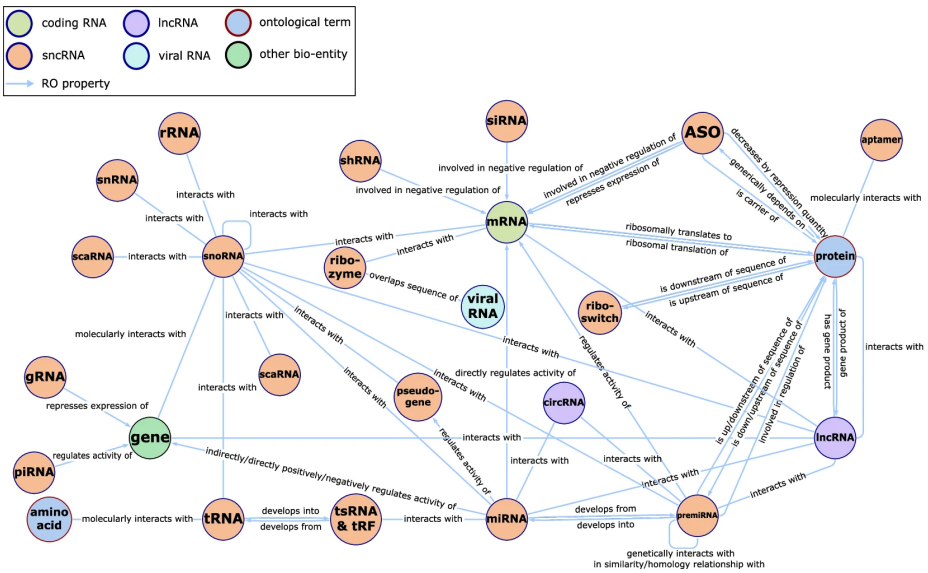


Figure 6: RNA-KG metagraph.

2.3.1 RNA sources characterization

Data was gathered from more than 60 public databases dealing with RNA sequences and annotations developed by well-reputed organizations, published in top journals in the last 10 years, periodically updated, and containing significant amounts of RNA molecules and relevant relationships with other types of molecules and bio-entities.

The collected data was transformed into a common format (tab-separated values - TSV files), using Pandas [20] DataFrames. This step helped to standardize the data and remove any syntactic inconsistencies.

For the construction of RNA-KG eleven biomedical ontologies were employed to establish common semantics across the data sources. Table 1 lists the ontologies considered for this purpose. These ontologies were chosen because their terms and hierarchical structures are widely accepted by the scientific community for clearly describing biological classes and entities such as diseases, phenotypes, chemicals, biological processes, proteins, and their interrelationships.

To ensure the reliability of the relationships included in RNA-KG, only meaningful relationships were considered e.g., by setting up specific criteria, such as p-values or False Discovery Rate (FDR) thresholds, or experimental scores defined in the data sources.

Finally, the Relation Ontology (RO) is used for characterizing relationships. Table 2 reports the main relationships of the considered data sources according to the RO ontology. For each relation, the table reports the RO identifier, the corresponding meaning, and, whenever available, the inverse relation. The relation names that are exploited for unidirectional relationships are marked with the * symbol. The hierarchical organization

Name	Abbr.	Description
Human Phenotype Ontology [21]	HPO	Terms representing medically relevant phenotypes and disease-phenotype annotations.
Gene Ontology [22]	GO	Terms representing attributes of gene products in all organisms. Cellular component, molecular function, and biological process domains are covered.
Monarch Merged Disease Ontology	Mondo	Terms representing human diseases.
Vaccine Ontology [23]	VO	Terms in the domain of vaccines and vaccination.
Chemical Entities of Biological Interest [24]	ChEBI	Structured classification of molecular entities of biological interest focusing on "small" chemical compounds.
Uber-anatomy Ontology [25]	Uberon	Terms representing body parts, organs, and tissues in a variety of animal species, with a focus on vertebrates.
Cell Line Ontology [26]	CLO	Terms representing publicly available cell lines.
PROtein Ontology [27]	PRO	Terms representing protein-related entities (including specific modified forms, orthologous isoforms, and protein complexes).
Sequence Ontology [28]	SO	Terms representing features and properties of nucleic acid used in biological sequence annotation.
Pathway Ontology [29]	PW	Terms for annotating gene products to pathways.
Relation Ontology [30]	RO	Terms and properties representing relationships used across a wide variety of biological ontologies.

Table 1: Main biomedical ontologies used for RNA-KG construction (* modified to include only human and viral proteins).

of concepts in RO allows the expression of different kinds of relationships at different granularities (e.g., the general property interacts with can be substituted with more specific properties such as molecularly interacts with or genetically interacts with). Figure 7 summarizes the available relations involving RNA molecules and bio-entities (i.e., gene, protein, chemical, and disease), with miRNA-lncRNA interactions being the most common. The Relation Ontology (RO) is used for characterizing relationships. We were able to retrieve around 150 million distinct relationships of this type from public RNA-based data sources. In terms of cardinality, they are followed by lncRNA-mRNA interactions (~28 million) and miRNA-mRNA/miRNA-gene interactions (~6 million each). Finally, a meta-graph was constructed to represent all possible interactions derived from the considered data sources.

Relation ID	Name	Inverse Relation ID	Inverse Name
RO:0000056	participates in	RO:0000057	has participant
RO:0000079	function of	RO:0000085	has function
RO:0011013	indirectly positively regulates activity of		
RO:0001015	location of	RO:0001025	located in
RO:0011016	indirectly negatively regulates activity of		
RO:0002202	develops from	RO:0002203	develops into
RO:0002204	gene product of	RO:0002205	has gene product
RO:0002245	over-expressed in		
RO:0002246	under-expressed in		
RO:0002260	has biological role		
RO:0002263	acts upstream of		
RO:0002264	acts upstream of or within		
RO:0002291	ubiquitously expressed in	RO:0002293	ubiquitously expresses
RO:0002302	is treated by substance	RO:0002606	is substance that treats
RO:0002314	characteristic of part of		
RO:0002325	colocalizes with*		
RO:0002326	contributes to		
RO:0002327	enables	RO:0002333	enabled by
RO:0002331	involved in		
RO:0002387	has potential to develop into		
RO:0002428	involved in regulation of		
RO:0002430	involved in negative regulation of		
RO:0002432	is active in		
RO:0002434	interacts with*		
RO:0002435	genetically interacts with*		
RO:0002436	molecularly interacts with*		
RO:0002448	directly regulates activity of		
RO:0002449	directly negatively regulates activity of		
RO:0002450	directly positively regulates activity of		
RO:0002479	has part that occurs in		
RO:0002526	overlaps sequence of*		
RO:0002528	is upstream of sequence of	RO:0002529	is downstream of sequence of
RO:0002559	causally influenced by	RO:0002566	causally influences
RO:0003002	represses expression of		
RO:0003302	causes or contributes to condition		
RO:0004033	acts upstream of or within, negative effect		
RO:0004035	acts upstream of, negative effect		
RO:0010001	generically depends on	RO:0010002	is carrier of
RO:0011002	regulates activity of		
RO:0011007	decreases by repression quantity of		
BFO:0000050	part of	BFO:0000051	has part
RO:HOM0000000	in similarity relationship with*		
RO:HOM0000001	in homology relationship with*		

Table 2: Main relations among bio-entities involving RNA with the RO identifier (* symmetric relationship).

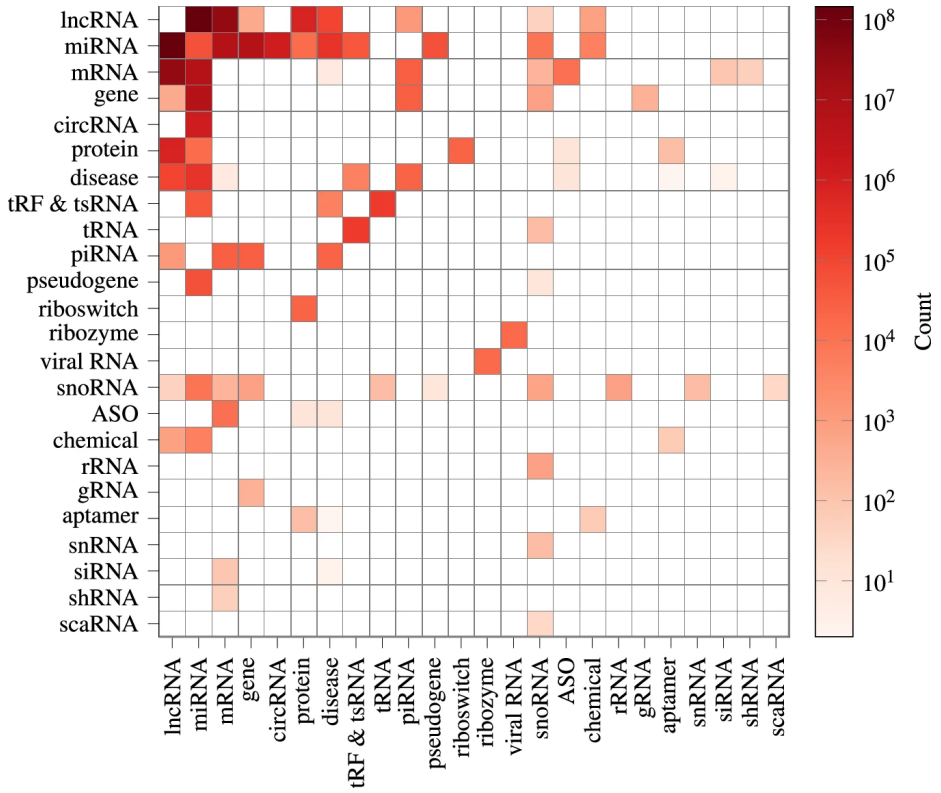


Figure 7: Number of relationships involving RNA molecules and relevant bio entities (gene, protein, chemical, and disease) within the considered RNA sources. Colors represent the ranges of relationships in log scale, as reported in the legend.

2.3.2 Ontological description of the KG

This step takes care of building a schema for the different bio-entities available in the graph. The schema (*RNA-KG meta-graph*) specifies all the classes of bio-entities and all the relationships that can exist between them, building an ontological schema based on the characterization of the various RNA sources.

A set of basic properties for each class was also defined: identifiers, node/edge types, and source provenance. This set was kept limited to avoid a size explosion of the final graph.

2.3.3 Ontological Alignment Specification

RDF triples were selected as representation model due to their flexibility and uniformity, which support an ontologically-grounded KG for various analyses and reasoning.

To align entities and relationships of the considered data sources to the schema defined

in the previous step, the PheKnowLator (Phenotype Knowledge Translator [16]) ecosystem was adopted. PheKnowLator is a tool that generates biomedical KGs based upon two knowledge models: (*i*) *Class-based* approach, where each entity is a subclass of an ontology class, or (*ii*) *instance-based* approach, where each entity is declared an instance of an ontology class. PheKnowLator was chosen because it supports both approaches and for its simplicity in defining look-up tables tailored to the biomedical domain. Even if RNA-KG is made available in all the supported knowledge models, in this thesis we considered the instance-based model that includes inverse relations because it is the most suitable to be processed by different kinds of ML algorithms for node and link prediction. Then, specific mapping rules were derived based on the characteristics of each RNA source and so that PheKnowLator could extract triples that respected the chosen ontology identifier.

PheKnowLator also cleans and merges ontologies that identify nodes and edges within the KG domain. This step ensures to semantically describe the full structure of RNA-KG and to be compliant with the *meta-graph* that was defined in the previous step of the building procedure. Merged ontology was then integrated in the KG to include hierarchical biologically meaningful relations among concepts.

Figure 8 shows a small toy-example subgraph extracted from RNA-KG according to the proposed set-up. We can notice the presence of inverse relationships (`located in` and its inverse `location of`), and the presence of ontology `subClassOf` edges (e.g., miRNA molecules are specified as `subClassOf` the SO term `miRNA`).

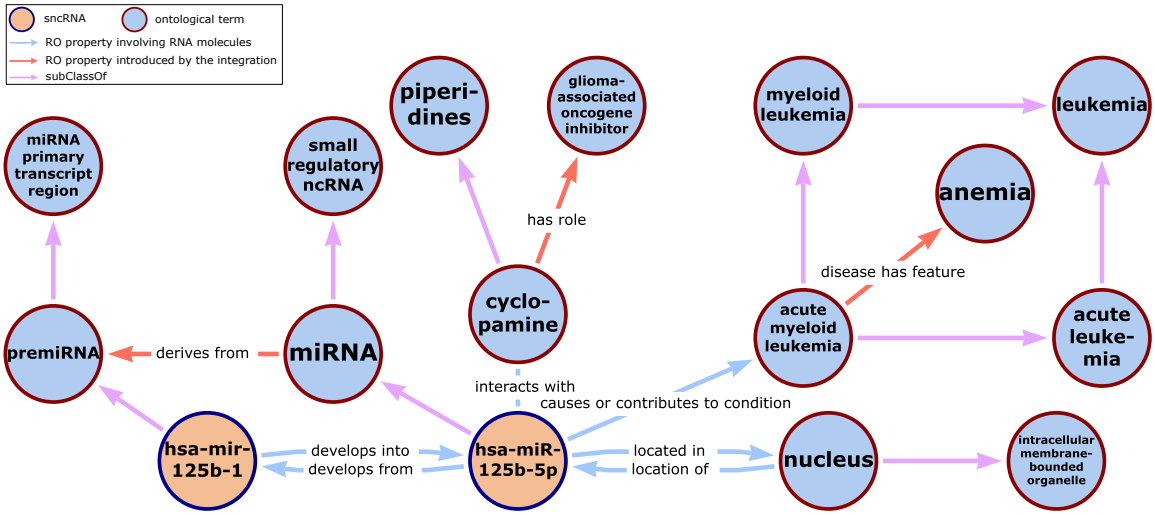


Figure 8: Example of a RNA-KG subgraph realized according to the instance-based, inverse relations, semantically abstracted (OWL-NETS without harmonization) parameters.

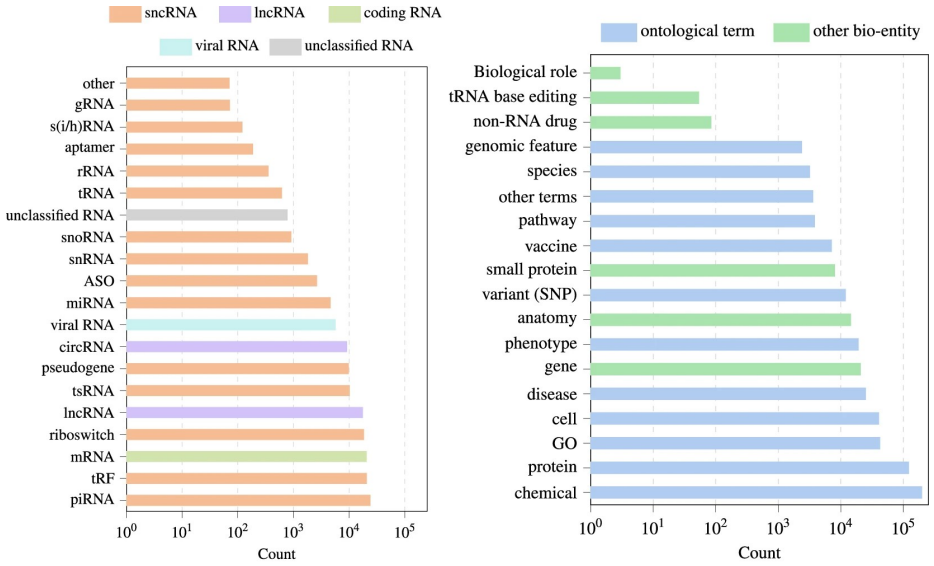
2.3.4 RNA-KG generation and analysis

PheKnowLator mapping rules were applied to pre-processed data to generate a KG compliant with the meta-graph. The GRAPE[31] library was used to evaluate the KG, facilitating the detection of topological features and anomalies, and supporting advanced graph embedding techniques. Additionally, a Blazegraph endpoint was created for open access to RNA-KG. SPARQL queries enable extraction of specific graph portions for various analyses. The full RNA-KG is available for download at `http://RNA-KG.anacleto.di.unimi.it`.

Node distribution. Figure 9a presents the distribution of nodes based on the primary types of RNA molecules, categorizing them into sncRNA, mRNA, viral RNA, and lncRNA. Among these, miRNA, mRNA, and lncRNA are the most prevalent in RNA-KG due to their extensive study and well-established classification. Many RNA sources are categorized as lncRNA and miRNA, while mRNA is commonly involved in relationships with various other ncRNAs. Figure 9b further elaborates on the distribution of ontological terms.

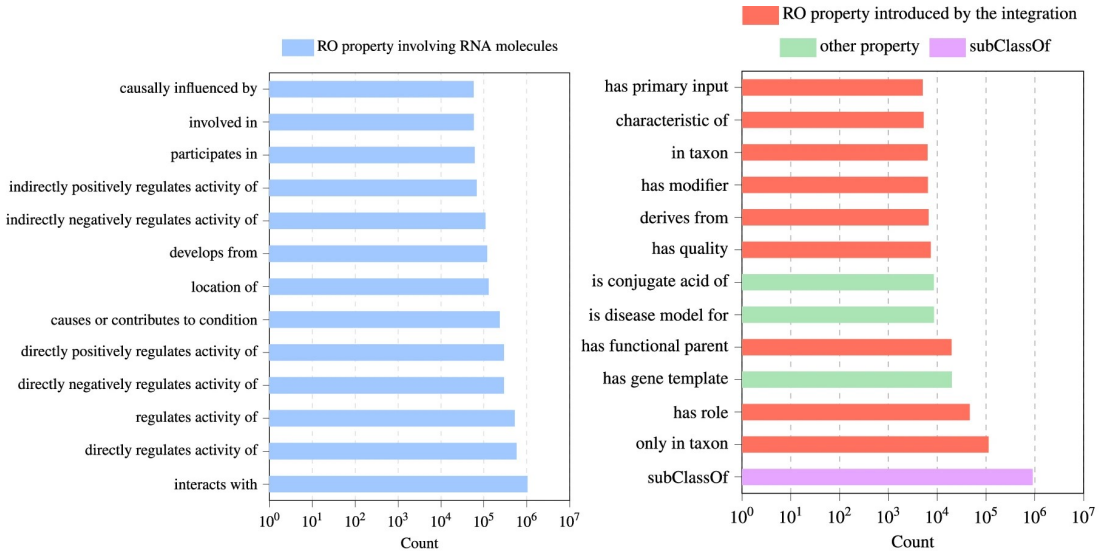
Edge distribution. Figure 10a illustrates the distribution of edge types associated with RNA molecules. As expected from the meta-graph structure, "interacts with" is the most frequent edge type, as it represents interactions for which the original data source did not provide specific semantics. The significant presence of "regulates activity of" edges reflects the abundance of miRNA molecules in RNA-KG, which regulate the activity of entities such as genes, pseudogenes, and mRNA. Figure 10b displays the distribution of the remaining edges, where the prevalence of "subClassOf" edges is notable due to the integration of bio-ontologies, with each RNA molecule being a subclass of a relevant SO class (e.g., SO_0000276 for miRNA molecules).

RNA-KG statistics. In the study of RNA-KG, it was found that the average degree of the undirected graph version of RNA-KG is relatively small (25.23), and the diameter is also modest (33), indicating properties typical of scale-free networks. Figure 11.a shows the degree distribution, which suggested a heavy-tailed distribution [32]. These properties are usually associated with scale-free networks, or more generally to heavy-tailed degree distributions, which is a common structure in real-world complex systems. This motivates the computation of the empirical *complementary cumulative distribution function* (CCDF) for the degree, reported in Figure 11.b. A linear trend in this plot is usually associated with a power law distribution. The theoretical power law obtained for the degrees is shown in Figure 11.b together with other common heavy-tailed distributions. The truncated power law was the best fit found. The closeness centrality distribution is shown in Figure 11.c, and it shows a bimodal behaviour, which is explained by the existence of a well-connected core, usually present in heavy-tail degree distribution networks [17].



(a) Distribution of RNA molecules. **(b)** Distribution of other bio-entities.

Figure 9: Node distribution according to node types.



(a) Distribution of edges involving RNA molecules. Only direct edges with more than 10,000 occurrences are reported.

(b) Distribution of other edges in RNA-KG. Only direct edges with more than 5,000 occurrences are reported.

Figure 10: Edge distribution according to edge types.

t-SNE representation. Figure 12 illustrates the t-SNE visualization of node and edge embeddings in RNA-KG, generated using the GRAPE implementation of Node2Vec with continuous bag of words (CBOW) and a walk length of 5. Figure 12a shows that node

Table 3: Basic topological properties of RNA-KG

Graph parameter	
Number of nodes	673,825
Number of directed edges	12,692,212
Max out degree	26,939
Max in degree	116,363
Number of edges[1]	8,501,595
Max degree[1]	116,368
Min degree[1]	1
Mean degree[1]	25.23
Diameter[1]	33
Upper bound Treewidth[1]	10,611
Mean closeness centrality[1]	2.92×10^{-7}

[1] Values calculated on the undirected version of the KG.

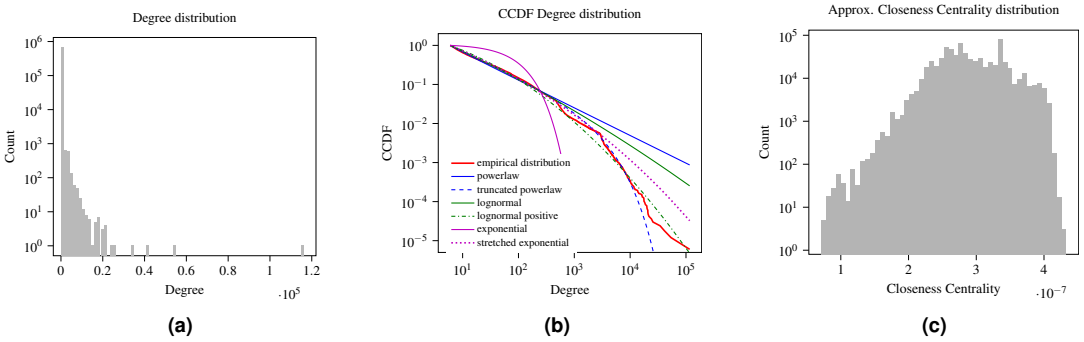
embeddings effectively group nodes of the same type based on their functional similarities. Figure 12b displays edge embeddings, revealing that while most edges are well-clustered, certain relations like "interacts with" and various forms of "regulates activity of" overlap significantly.

2.4 RNA-KG views

An *RNA-KG view* is an RNA-KG subgraph tailored for a specific edge prediction task. Views might be integrated with PheKnowLator's Human disease benchmark KG [33, 34] subgraphs of interest for including other relevant relationships such as *gene-disease*, *disease-disease*, *gene-gene*. According to PheKnowLator ecosystem strategy, we build different views integrating appropriate biomedical ontologies.

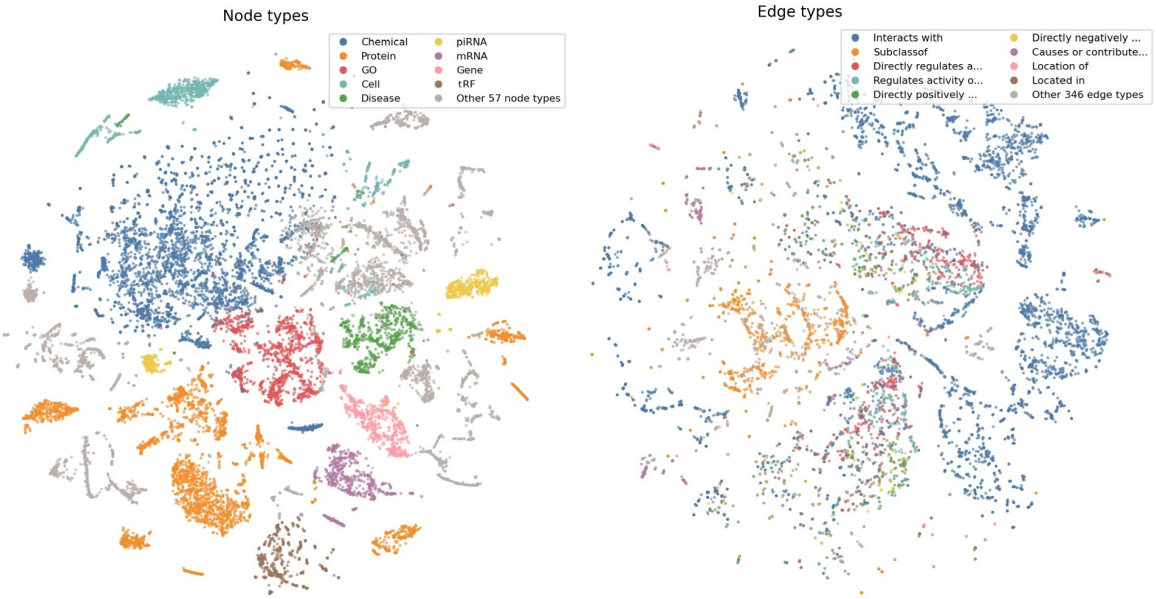
The objective is to provide a series of basic prediction tasks involving different kinds of triples that are relevant in the biomedical domain using GRL methods for heterogeneous graphs. The idea is to systematically apply link prediction methods on RNA-KG by defining a set of link predictive tasks to which systematically apply RW-based methods for heterogeneous graphs and GNN methods for heterogeneous graphs.

Table 4 shows a schematic representation of each view that we considered for conducting experiments. An identifier (`Name` column) is associated with each view, along with the types of nodes that are linked within the graph (`Node types` column). In the remainder, we will refer to these views as View *n*, e.g., View 1. Nodes and edges within views



(a) (b) (c)

Figure 11: (a) Node degree distribution (semi-log). (b) Complementary cumulative distribution (CCDF) for the node degree. (c) Approximated closeness centrality distribution.



(a) Node types embedding. **(b)** Edge types embedding.

Figure 12: Node and Edge Embeddings.

are identified according to biomedical ontologies in `Integrated ontologies` column. `Relevant edges` shows the edge types we included in the test sets. We integrated these ontologies in the views according to PheKnowLator strategy described in [16]. Detailed statistics for each view are presented in Figures 13–15.

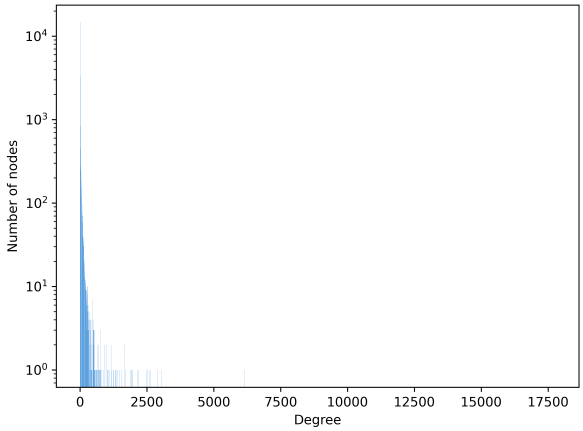
Degree distribution plots and cardinalities for node and edge types are reported when they are relevant to the downstream analysis. The degree distribution plot for each view confirms the same behaviour of the original graph they were generated from. A similar

observation holds for the CCDF degree and closeness centrality distribution plots (not reported here for the sake of readability).

View name	Node types	Relevant edges	Integrated ontologies
View 1	miRNA phenotype disease gene	miRNA-disease miRNA-phenotype miRNA-gene	Mondo HPO RO
View 4	miRNA phenotype disease gene	miRNA-disease miRNA-phenotype miRNA-gene	Mondo HPO SO RO
View 5	miRNA phenotype disease gene pseudogene lncRNA pathway protein GO term	miRNA-disease miRNA-phenotype miRNA-gene miRNA-GO lncRNA-disease lncRNA-phenotype lncRNA-GO protein-GO	GO SO PRO PW Mondo HPO RO

Table 4: Overall description of RNA-KG views.

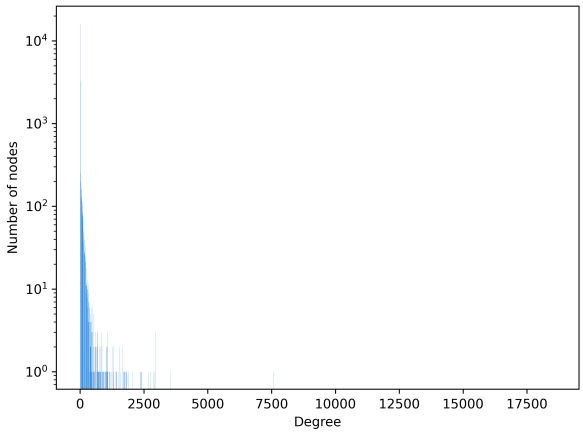
Figure 13: Node degree distribution and counts for nodes and edges in View 1.



Node	Count
Disease	23972
Phenotype	18488
Gene	18341
miRNA	3384

Source	Target	Count	Edge type(s)
miRNA	Gene	700917	Involved in regulation of
miRNA	Disease	103575	Causes or contributes to condition
miRNA	Phenotype	35690	Causes or contributes to condition
Gene	Phenotype	24562	Causes or contributes to condition
Gene	Disease	12642	Causes or contributes to condition

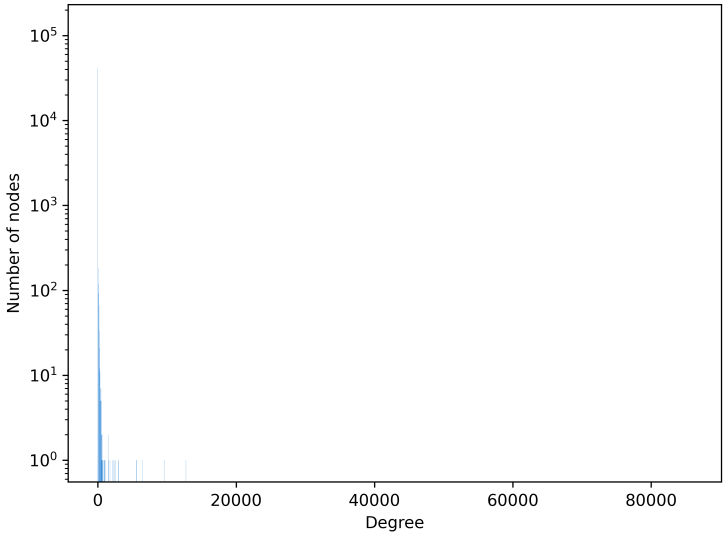
Figure 14: Node degree distribution and counts for nodes and edges in View 4.



Node	Count
Disease	24203
Gene	19517
Phenotype	19152
miRNA	4565
Genomic feature	2391

Source	Target	Count	Edge type(s)
miRNA	Gene	1333269	Involved in regulation of
miRNA	Disease	105211	Causes or contributes to condition
miRNA	Phenotype	36752	Causes or contributes to condition
Gene	Phenotype	24519	Causes or contributes to condition
Gene	Genomic feature	23322	Subclassof
Gene	Disease	14422	Causes or contributes to condition
miRNA	miRNA	5750	Develops from Develops into
miRNA	Genomic feature	4565	Subclassof

Figure 15: Node degree distribution and counts for nodes and edges in View 5.



Node	Count
Protein	217561
GO	42537
Disease	24073
Chemical	23922
Gene	20376
Phenotype	18713
lncRNA	16641
Pseudogene	5156
miRNA	4800
Pathway	3850

Source	Target	Count	Edge type(s)
miRNA	Gene	1439378	Directly positively regulates activity of Regulates activity of Directly negatively regulates activity of Indirectly positively regulates activity of Indirectly negatively regulates activity of
lncRNA	Protein	198287	Interacts with
miRNA	Disease	113450	Causes or contributes to condition
miRNA	Protein	81095	Interacts with Located in Is downstream of sequence of Involved in regulation of Is upstream of sequence of
miRNA	Pseudogene	60011	Regulates activity of
miRNA	GO	58472	Interacts with Causes or contributes to condition Participates in ...
miRNA	Phenotype	39220	Causes or contributes to condition
lncRNA	GO	36476	Causes or contributes to condition Participates in ...
lncRNA	Disease	26463	Causes or contributes to condition Ubiquitously expressed in Under-expressed in Over-expressed in
miRNA	lncRNA	18484	Interacts with
miRNA	Pathway	17450	Participates in
miRNA	miRNA	7326	Interacts with Develops into Develops from

Chapter 3

Graph Representation Learning Techniques on Heterogeneous Graphs

There are many instances where we use a group of points connected either by lines or by arrows to represent various situations; the points might represent people, places or atoms, and the lines or arrows can represent relationships, pipelines or chemical bonds. Such diagrams are common across disciplines and are known by different names: sociograms (psychology), circuit diagram (physics), communication networks, family trees, etc. D. König first proposed the term "graph"[35] as a universal label for all these representations.

Graphs are highly versatile data structures used to represent networks of entities and the relationships between them. These networks can be vast and span various domains, such as social media, financial systems, and biological networks.

Since most data in real-world applications come in the form of graphs, research in graph representation learning [36] has gained significant attention in recent years. In this chapter, we will be discuss about heterogeneous graphs, as well as the methods of graph representation learning applied to it.

3.1 Homogeneous graph representation learning

Homogeneous Graphs are the type of graphs containing all the nodes and the edges of the same type depicting the uniformity in the graph and in it's structure. for an instance we can take Protein-Protein Interaction Networks where all the nodes are Proteins and the edges represent the interaction between them. (Figure 16) represents a homogeneous graph where nodes represent person and the edges represent money transfer. Some examples of homogeneous graphs are:

1. Social Networks: Nodes represent people, and edges represent friendships or social connections.

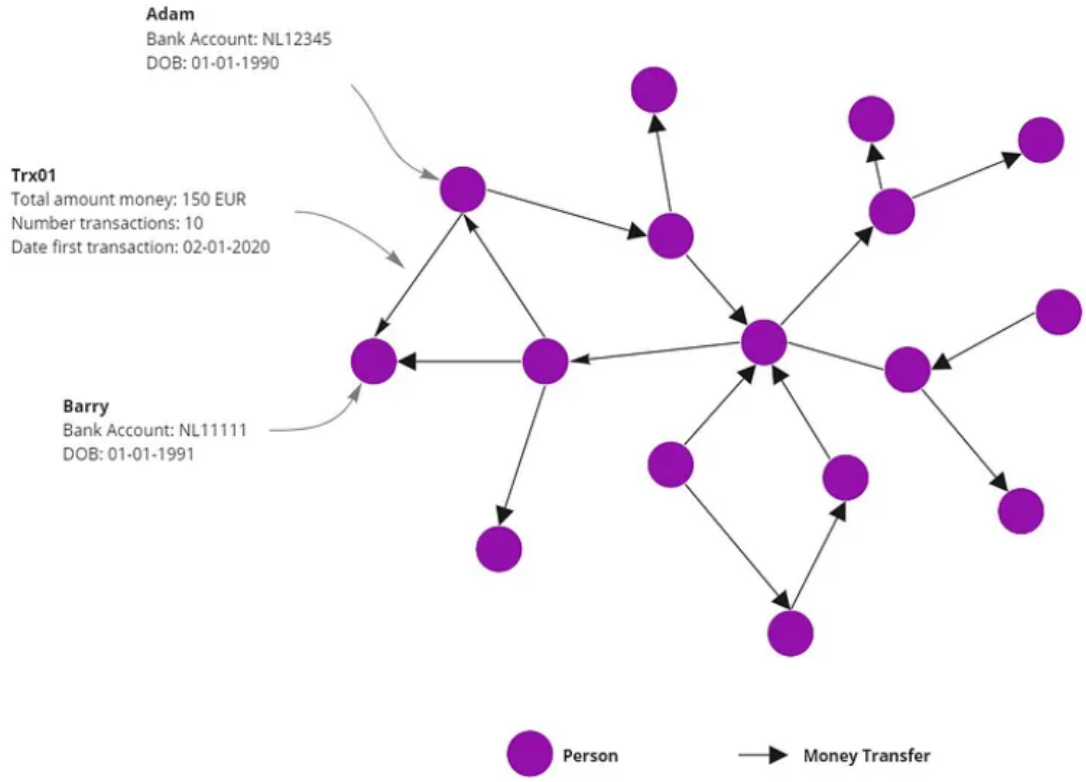


Figure 16: Homogeneous Graph.[37]

2. Citation Networks: Nodes represent research papers, and edges represent citations between papers.

3. Web Graphs: Nodes represent web pages, and edges represent hyperlinks between them.

Formally, a homogeneous graph G = (V, E) is defined by a set of nodes V and a set of edges E between these nodes. We denote an edge going from node $u \in V$ to node $v \in V$ as $(u, v) \in E$. Here, all vertices (V) are of the same type and all edges (E) represent the same type of relationships.

To effectively use graphs in different downstream applications, it is important to represent them in an efficient manner. Figure 17 shows that graph can be simply represented using the adjacency matrix [38] which is a square matrix whose elements indicate whether pairs of vertices are adjacent or not in the graph, or using the extracted features of the graph. However, the dimensionality of the adjacency matrix is often very high for big

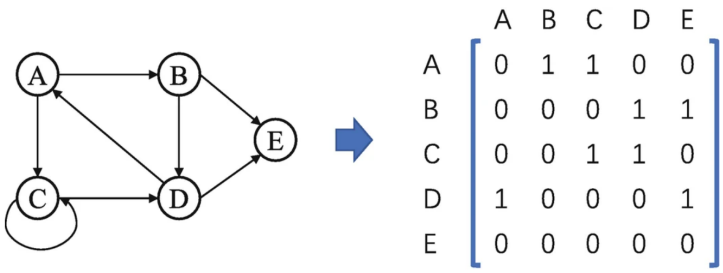


Figure 17: The adjacency matrix representation for a directed graph

graphs, and the feature extraction based methods are time consuming and may not represent all the necessary information in the graphs.

3.1.1 Graph Representation Learning

Graph representation learning (GRL)[39] is a powerful approach that aims to translate graph-structured data into low-dimensional, dense vector embeddings that capture the essential structure and features of nodes, edges, or entire graphs. The core idea of GRL is to learn these embeddings from the input graph data and then apply them to downstream tasks like node classification, link prediction, graph classification, community detection, and visualization. By representing nodes in a reduced vector space, GRL allows for efficient processing of complex graphs with fewer computational constraints. Techniques for GRL fall into two main categories: traditional graph embedding methods[40] and more advanced approaches like graph neural networks (GNNs)[41], which have gained significant attention in recent years. These methods can handle both static and dynamic graphs.

In Figure 23, a comprehensive overview of Graph Representation Learning is presented, which highlights the two primary approaches: Graph Embedding and Graph Neural Networks (GNNs). The figure categorizes various graph embedding techniques, including homogeneous[42], heterogeneous[43], and attributed embeddings, showcasing methods such as Node2Vec[44], DeepWalk[45], and meta-path-based embeddings. Additionally, it outlines different GNN architectures, including Graph Convolutional Networks (GCNs)[46], Graph Recurrent Networks, and Graph Autoencoders[47], each with their respective methods. This visual map provides a clear distinction between traditional graph embeddings and more recent neural-based approaches, facilitating a better understanding of their applications in graph-based data analysis

An Overview of Graph Embedding and Its Types

To utilize graphs effectively in downstream machine learning and data mining tasks, it is necessary to represent graph elements, such as nodes and edges, through numerical

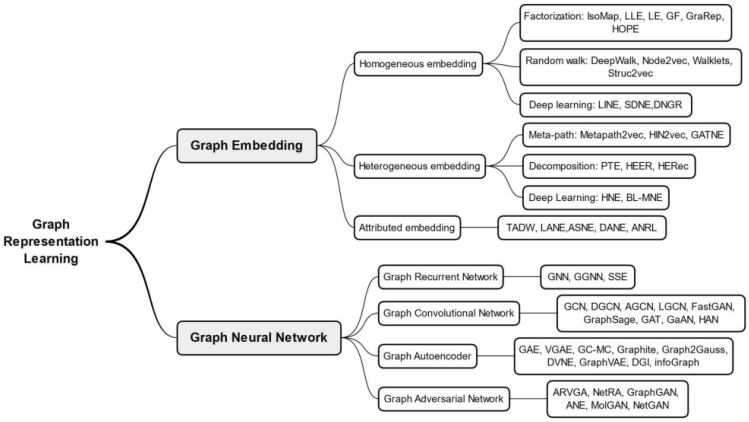


Figure 18: Overview of Graph Representation Learning.

features. One common approach is to use the adjacency matrix, but for large graphs, this becomes inefficient as the size of the graph grows like mentioned previously. A more practical method is to represent the graph and its components through feature sets, particularly for individual nodes. For instance, in anomaly detection, nodes with dense neighborhoods are often potential anomalies. Including features like in-degree and out-degree in node representations can improve the detection accuracy of such anomalies, as anomalous nodes often exhibit higher degrees. However, identifying relevant features across different applications and capturing the graph's overall structure can be challenging and time-intensive when done manually. To address this, the graph embedding methods[48] have been proposed, which study the issue of automatically generating representation vectors for the graphs. These techniques frame graph representation learning as a machine learning problem, leveraging the graph's structure and properties to generate embedding vectors. Graph embedding methods encompass node, edge, and subgraph embedding approaches which are defined as follows:

1. Node embedding: Given a graph $G = (V, E)$, where V represents the set of nodes and E represents the set of edges, node embedding involves learning a mapping function $f : v_i \rightarrow \mathbb{R}^d$. This function transforms each graph node v_i into a low-dimensional vector of dimension d, where d is much smaller than the total number of nodes $|V|$ ($d \ll |V|$). The goal is to maintain the relationships and similarities between nodes in the embedding space. As shown in Figure 19 and Figure 20, these embeddings capture the structural and feature information of the nodes and their relationships with other nodes.

2. Edge embedding: In a graph $G = (V, E)$, edge embedding converts each edge into a low-dimensional vector of dimension d, where d is much smaller than the total number

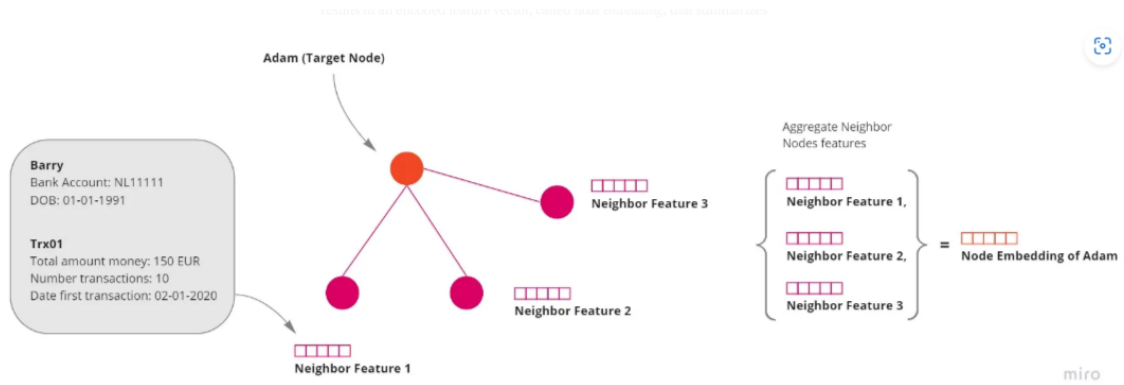


Figure 19: Generating node Embeddings.

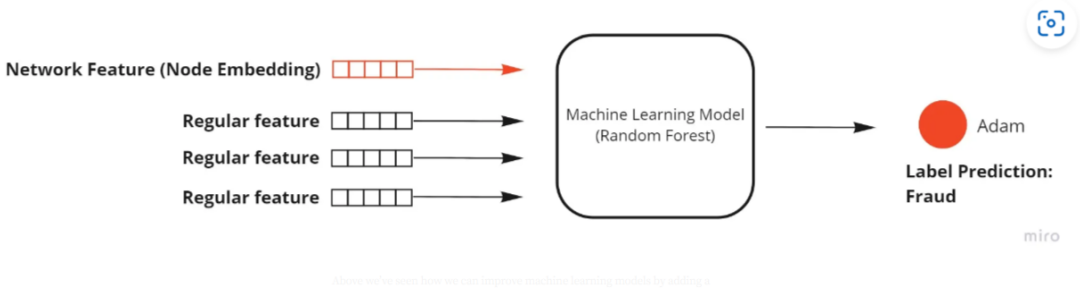


Figure 20: Link Prediction.

of nodes $|V|$ ($d \ll |V|$). The goal is to maintain the relationships and similarities between edges in the embedding space.

3. Subgraph embedding: Given a graph $G = (V, E)$, subgraph embedding methods generate a low-dimensional vector d for a subgraph within G, ensuring that the vector dimension d is much smaller than the total number of nodes $|V|$ ($d \ll |V|$), while preserving the similarities between subgraphs in the embedding space.

Use cases of Graph Embeddings

1. Node Classification: It is the task of assigning a label to each node in a test dataset. These labels represent categories or classes that the nodes belong to. The process leverages both the features of individual nodes (such as their properties or attributes) and the structure of the graph (i.e., the relationships and connections between nodes). The idea is that nodes connected or similar to each other are likely to share the same label. For

instance, in RNA-KG, nodes might represent different RNA molecules, and the classification task could involve predicting whether a particular RNA molecule is associated with a specific biological process or disease.

2. Anomaly Detection: In domains like cybersecurity or fraud detection, graph embeddings can help identify abnormal patterns. For instance, in financial transaction networks, unusual patterns in transaction behavior can indicate fraudulent activity. By embedding the transaction graph, models can easily spot anomalies that deviate from the norm.

3. Community Detection: refers to the process of identifying clusters or groups of nodes within a graph that are more densely connected to each other than to the rest of the graph. Community detection algorithms analyze the network's structure to find groups of nodes that have a higher probability of being connected with each other. These communities might represent natural divisions within the network, such as different functional modules in a biological network or social circles in a social network. In RNA-KG, this could mean identifying groups of RNA molecules that work together in the same biological pathway or process.

4. Recommender Systems: Use information about users and items to suggest items or connections that the user might be interested in. In the context of a graph, this involves recommending new connections or items based on the graph's structure and node features. Recommendation systems can be implemented using link prediction techniques, where the system predicts potential links between users and items. For example, in a social network, the system might recommend new friends (user-user links) or new content (user-item links) based on existing connections. In RNA-KG, it could suggest potential research areas or experimental targets by predicting new links between known molecules and diseases.

3.1.2 Graph neural network (GNN)

Graph Neural Networks (GNNs) (Figure21)[49] are a type of neural network that is designed to operate directly on graph-structured data. While traditional neural networks work on regular grid data like images (2D grids of pixels) or text (sequences of words), GNNs generalize neural networks to arbitrary graph structures. They are highly influenced by Convolutional Neural Networks (CNNs) and graph embedding. GNNs are used in predicting nodes, edges, and graph-based tasks.

Graph Neural Networks (GNNs) are highly versatile and find application across various tasks. In (Figure22), we can see different applications of GNN, which include node classification, where labels for nodes are predicted based on neighboring node features;

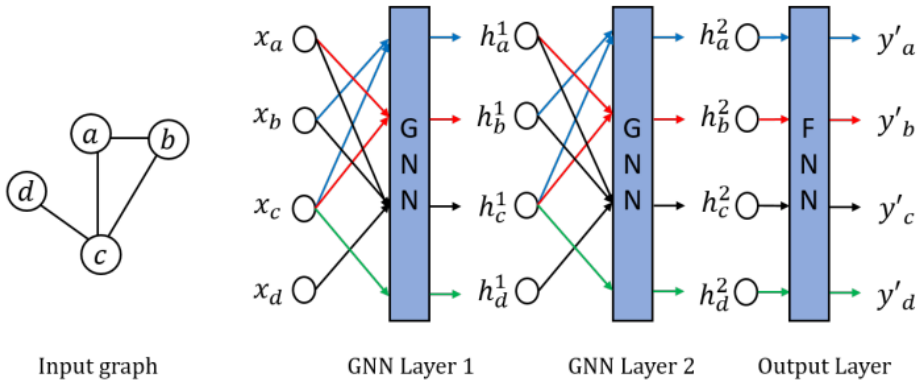


Figure 21: A general supervised framework for training Graph Neural Network (GNN) layers is outlined below. Two GNN layers are applied to an input graph in order to compute the node representation vectors for its nodes. The colors on arrows indicate the neighbors of a target node that are aggregated to generate the target node's representation. Let x_a be the feature vector of node a, and h_a^1 and h_a^2 represent the node embedding vectors for node a after applying the first and second GNN layers, respectively. The embeddings generated from these layers are then used for a node classification task. The predicted label for node a is denoted as y'_a.

link prediction, which identifies relationships between nodes; and graph classification, which categorizes graphs based on structural properties. Other important use cases include community detection, graph embedding, and graph generation, all of which leverage the underlying graph structure to extract meaningful insights or generate new graph data.

Some commonly used GNN models are Graph Convolutional Network (GCN), Graph Attention Network (GAT), Graph Isomorphism Network (GIN), Temporal Graph Networks (TGNs), Relational Graph Convolutional Network (R-GCN) and HGCN (Hyperbolic Graph Convolutional Network.

Categories of GNN-Based Methods

There are three main categories of GNN-based methods:

1. Static GNN: Static GNN based graph embedding methods are suitable for graph representation learning on static graphs, which do not change over time. It is commonly used for Node classification, graph classification, and link prediction in static datasets.

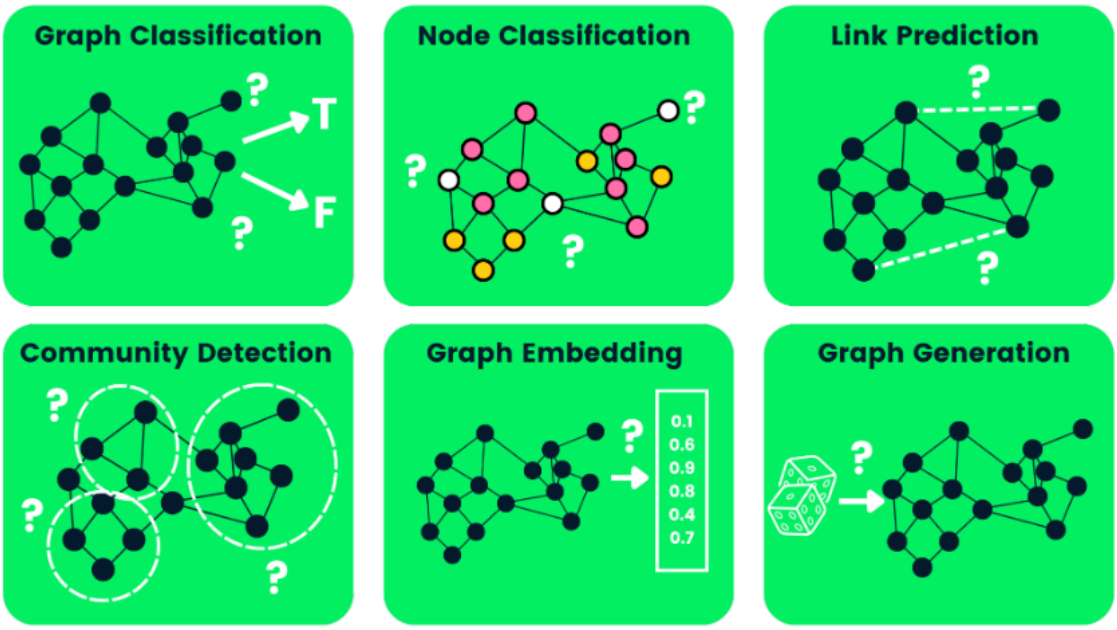


Figure 22: Use cases of GNN.[50]

2. Spatiotemporal GNN: Spatial-temporal GNNs are a category of GNN that capture both the spatial and temporal properties of a graph. They leverage temporal dependencies and spatial structures to make predictions. There are wide applications for STGNNs such as traffic flow forecasting, epidemic forecasting and sleep stage classification. For instance, in traffic prediction, the future traffic in a road is predicted considering the traffic congestion of its connected roads in previous times.

3. Dynamic GNN: Dynamic GNNs are specifically designed for dynamic graphs where nodes and edges can be added or removed over time. They focus on maintaining up-to-date representations of the graph as it evolves. It is used in Real-time social network analysis, fraud detection, and recommendation systems that need to adapt to changing user behavior.

In Figure 23, the advantages and disadvantages of traditional graph embedding methods compared to Graph Neural Networks (GNNs) are highlighted and then in Figure 24, we illustrate two key approaches—graph embedding and graph neural networks (GNNs)—for handling graph data based on topology and attributes, along with their respective outcomes.

Graph Representation Learning can be applied to various types of graphs depending on the structure and properties of the data. GRL techniques are flexible and can work on different graph structures such homogeneous graphs, heterogeneous graphs, directed

Category	Advantages	Disadvantages
Traditional	Higher expressive power, scalable in some categories	Not generalizable to unseen nodes, not considering node/edge attributes easily
GNN-based	Generalize to unseen nodes, consider node/edge attributes, can do both task-specific and node similarity based training	Expressive power, scalability, over-smoothing, over-squashing, homophily assumption and catastrophic forgetting.

Figure 23: Comparison of traditional and GNN-based graph representation learning.

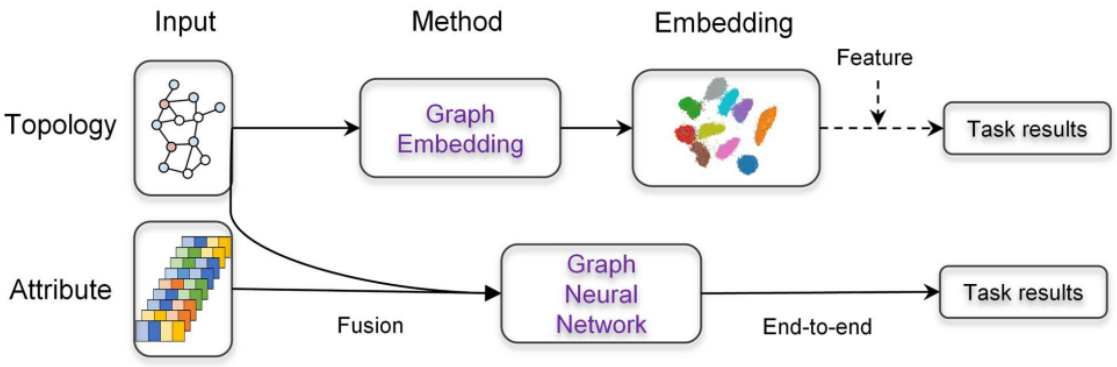


Figure 24: Workflow diagram for processing graph data using Graph embedding and GNN.

graphs, undirected graphs etc.

3.1.3 Approaches for Homogeneous Graphs:

Traditional Methods: Traditional methods for generating graph embeddings aim to extract meaningful low-dimensional representations of nodes by leveraging the graph's structural properties. These methods primarily focus on two main approaches: matrix factorization and random walk-based methods.

1. Matrix Factorization: It is a mathematical technique used to decompose a large matrix into a product of smaller matrices, typically to extract meaningful lower-dimensional representations or latent factors. In the context of graphs, matrix factorization is often applied to matrices that represent the structure or properties of the graph, such as the adjacency matrix and the Locally linear embedding.

a. Adjacency Matrix/Laplacian Matrix Factorization: This involves factorizing the adjacency matrix or graph Laplacian matrix to obtain lower-dimensional node embeddings. The idea is that the factorization captures the key structural properties of the graph.

b. Locally Linear Embedding (LLE): A non-linear method that preserves the local structure of the graph. LLE creates node embeddings by reconstructing each node as a

linear combination of its neighbors. It captures both the graph topology and the local relationships between nodes.

2. Random Walk-based Methods: These are graph embedding techniques that generate node representations by simulating random walks on a graph. These methods capture the structural and relational properties of the graph by exploring paths and sequences of nodes, focusing on the co-occurrence of nodes during the walk. Two random walk-based methods are DeepWalk and Node2Vec.

a. DeepWalk: This method generates node embeddings by performing short random walks on the graph to create node sequences, which are then fed into a language model like Word2Vec to generate embeddings. The key idea is to treat nodes that frequently co-occur in the random walks as more similar. It is simple and effective, and it captures both local and global graph structures. However, It may struggle with large graphs due to memory and computational constraints.

b. Node2Vec: An extension of DeepWalk that introduces a biased random walk. Node2Vec allows more control over the random walk process by tuning parameters for DFS-like (depth-first) or BFS-like (breadth-first) exploration of the graph.It can capture both structural roles (nodes with similar connections) and homophily (nodes that are close in the graph).

Deep Learning-Based Approaches: Neural methods have become the most prominent approaches for learning node embeddings in homogeneous graphs. These methods, particularly graph neural networks (GNNs), rely on the notion of message passing or neighborhood aggregation to capture node relationships and graph structure. Some of the approaches are:

1. Graph Convolutional Networks (GCNs): GCNs extend convolutional operations from grid-like data (e.g., images) to graph-structured data. Each node's representation is updated by aggregating features from its neighbors. It is effective for semi-supervised learning tasks, such as node classification, where a subset of nodes is labeled. However, it typically require the entire graph to be loaded into memory, which can be a bottleneck for very large graphs.

2. Graph Attention Networks (GATs): GATs improve upon GCNs by introducing an attention mechanism. Rather than treating all neighbors equally, GATs assign different weights to neighboring nodes based on their importance to the target node. It is able to focus on the most relevant neighbors for each node, improving accuracy in tasks like node classification.

3. Graph Isomorphism Networks (GINs): GINs are a special type of GNN that can distinguish between non-isomorphic graphs. They use a specific aggregation function (a sum over node features) that allows them to learn powerful representations capable of distinguishing graph structures. It is expressive enough to differentiate complex graph

structures, unlike traditional GCNs.

3.2 Heterogeneous Graph representation learning

Formally, a heterogeneous graph (HG)[51], also referred to as a **heterogeneous information network**, is defined as a graph $G = (V, E)$, where V represents the set of nodes, and E represents the set of links or edges. Each node $v \in V$ and each link $e \in E$ is associated with a mapping function $\phi(v) : V \rightarrow A$ and $\phi(e) : E \rightarrow R$, respectively. Here, A denotes the types of nodes, and R denotes the types of links, with the condition that $|A| + |R| > 2$, indicating the presence of more than one type of node or link in the graph.

The network schema for a heterogeneous graph is defined as $S = (A, R)$, which serves as a meta-template, characterizing the relationships between node types and link types within the graph [52].

These graphs (Figure 25)[53] contain multiple types of nodes and/or edges. These graphs capture complex relationships between different entities, making the structure more intricate compared to homogeneous graphs. Examples: Knowledge Graphs: Nodes represent different entities (e.g., people, organizations, locations), and edges represent various relationships (e.g., works_{at}, lives_{in}. E-commerce Graphs: Nodes can represent users, products, and reviews, while edges represent user-product interactions or product co-purchases. Biological Networks: Nodes represent different biological entities (e.g., genes, proteins), and edges represent various interactions (e.g., protein-protein interactions, gene regulation).

3.2.1 Challenges in Heterogeneous Graphs

Heterogeneous graphs are more complex than homogeneous graphs. The increased complexity of heterogeneous graphs introduces additional challenges:

1. Heterogeneous Node and Edge Types: Different types of nodes and edges may contain diverse information that cannot be captured uniformly. For example, the relationship between authors and papers is different from the relationship between papers and venues. Capturing the semantic differences between various types of nodes and edges is critical to learning meaningful representations.

2. Complex Interactions between Entities: In heterogeneous graphs, interactions between entities (nodes and edges) are more complex due to the multiple types of relationships. Capturing the dependencies between different entity types is challenging, as these interactions are not uniform. Modeling these multi-relational interactions is difficult since they require specialized techniques that account for the varying nature of relationships between different entities.

3. High Dimensionality of Multi-Type Features: Heterogeneous graphs often have features of different types and dimensions for different node categories. This results

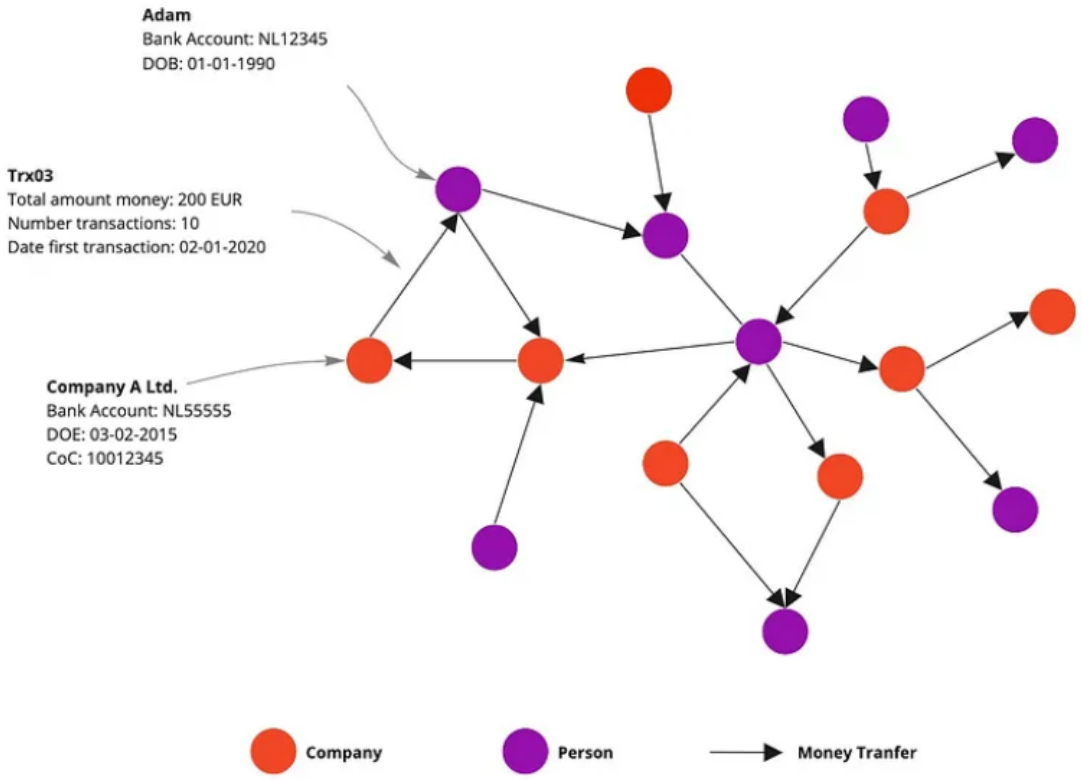


Figure 25: Heterogeneous Graph.[37]

in high-dimensional, diverse feature sets that are difficult to integrate. Efficiently handling and integrating high-dimensional features from different types of nodes and edges is challenging due to the varied semantics and scales of the data.

3.2.2 Approaches for Heterogeneous Graphs

Heterogeneous graphs are more complex because they contain multiple types of nodes and edges. The challenge in heterogeneous graphs is to effectively capture the diverse relationships between different types of entities.

1. Meta-Path Based Approaches: **a. Meta-paths** are sequences of nodes and edge types that capture specific relationships between different node types. These methods extend the random walk-based approaches used for homogeneous graphs.

b. HIN2Vec: This method is a direct extension of random walks to heterogeneous graphs. Instead of random walks that treat all nodes equally, HIN2Vec[54] uses meta-paths to guide the walks through specific types of nodes and edges. The meta-paths encode meaningful relationships between node types.

c. Metapath2Vec: Similar to DeepWalk, Metapath2Vec[55] uses random walks guided by meta-paths. The walks generate sequences of nodes, which are then used to train embeddings through a skip-gram model (like Word2Vec[56]). Meta-paths allow the model to capture the rich semantics of the graph by encoding specific node types and relationships.

2. Relational Graph Convolutional Networks (RGCN): RGCN[57] extends the basic GCN to heterogeneous graphs by learning separate weight matrices for each type of edge (or relation). For example, in a knowledge graph, RGCN would have different weights for edges representing "employed by" versus "located in." It allows the model to explicitly account for the differences between edge types, making it well-suited for knowledge graph embeddings and tasks like link prediction. However, the need for separate weight matrices for each relation increases the model complexity, limiting scalability for graphs with many relations.

3. HetGNN (Heterogeneous Graph Neural Networks): HetGNN[58] is designed to handle complex, heterogeneous graphs by sampling heterogeneous neighbors for each node and applying different aggregation functions for different node types. The model combines node content features with topological information to create node embeddings. It has been used in various domains, such as recommendation systems and knowledge graph embeddings.

3.3 Models used for Link Prediction and Node Classification in RNA-KG

3.3.1 DistMult

DistMult[59] is a simple yet effective model for embedding entities and relations in knowledge graphs. It is based on the Bilinear model[60] where each relation is represented by a diagonal matrix rather than a full matrix.

The **score function** in DistMult for a given triple (h, r, t) (head, relation, tail) is defined as:

$$f(h, r, t) = h^T \text{diag}(r) t = \sum_{i=0}^{d} h_i \cdot r_i \cdot t_i$$

This score captures pairwise interactions between only the components of **h** and **t** along the same dimension and thus can only deal with symmetric relations.

Advantages of Distmult

1. Simplicity: The model is simple and computationally efficient due to its use of diagonal matrices.
2.Symmetry: DistMult is particularly suitable for symmetric relations since it uses a diagonal relation matrix, which means that the relation between two entities is the same in both directions. For example, if h is related to t via relation r, then t is also related to h via the same relation. It performs well in scenarios where the relationships are symmetric, such as "is a friend of" or "is similar to."
3. 1-to-N/N-to-1:It can also handle 1-to-N / N-to-1 relationships, since multiple vectors can exist that have the same dot product score with relation applied to the head or tail.

Limitations of Distmult

1. Limited Expressiveness: DistMult cannot effectively capture asymmetric or more complex relationships since it relies on diagonal matrices. Due to the commutative property of multiplication, DistMult's scoring will always give the same score to (h,r,t) and (t,r,h), so these cannot be differentiated in score.
2. No Complex Interaction: It does not consider higher-order interactions between entities, which may lead to less accurate predictions in certain scenarios.

3.3.2 ComplEx

ComplEx[61] extends the DistMult model by introducing complex-valued embeddings to better capture asymmetric relationships and more complex interactions between entities. In ComplEx, each entity and relation is represented as a complex vector. This allows for capturing both real and imaginary parts, enabling the model to represent relationships in a richer way. Its main contribution is to embed KGs in complex space.
The **score function** for ComplEx is defined as follows:

$$f(h,r,t) = \mathrm{Re}\left(h^T \mathrm{diag}(r)\bar{t}\right) = \mathrm{Re}\left(\sum_{i=1}^{d} h_i r_i \bar{t}_i\right)$$

where $r, h, t \in \mathbb{C}^d$, and $\bar{t}_i$ represents the complex conjugate of t.

By using this scoring function, triples that have asymmetric relations can obtain different scores.

Advantages of ComplEx

1. Expressiveness: ComplEx can capture a wider variety of relationships, including both symmetric and asymmetric ones, due to its complex-valued embeddings.
2. Asymmetry: The use of complex numbers allows ComplEx to effectively model asymmetric relations, making it more versatile than DistMult. For example, it can represent relations like "is a parent of," where the direction of the relationship matters.
3. Higher Performance: It often yields better performance in link prediction and other knowledge graph tasks compared to simpler models like DistMult.

Limitations of ComplEx

1. Complexity: The introduction of complex embeddings increases the model's complexity, both in terms of computational resources and implementation.
2. Interpretability: The use of complex numbers can make the model's output less interpretable compared to real-valued embeddings.

Figure 26 provides a summary of different score functions $f(h, r, t)$[62], evaluating them on symmetry, antisymmetry, inversion, and composition properties. RotatE stands out by supporting all properties, including composition, making it versatile for complex relational patterns.

Model	Score Function	Symmetry	Antisymmetry	Inversion	Composition
SE	$-\|\|\boldsymbol{W}_{r,1}\mathbf{h} - \boldsymbol{W}_{r,2}\mathbf{t}\|\|$	✗	✗	✗	✗
TransE	$-\|\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|\|$	✗	✓	✓	✓
TransX	$-\|\|g_{r,1}(\mathbf{h}) + \mathbf{r} - g_{r,2}(\mathbf{t})\|\|$	✓	✓	✗	✗
DistMult	$\langle \mathbf{h}, \mathbf{r}, \mathbf{t} \rangle$	✓	✗	✗	✗
ComplEx	$\mathrm{Re}(\langle \mathbf{h}, \mathbf{r}, \bar{\mathbf{t}} \rangle)$	✓	✓	✓	✗
RotatE	$-\|\|\mathbf{h} \circ \mathbf{r} - \mathbf{t}\|\|$	✓	✓	✓	✓

Figure 26: The pattern modeling and inference abilities of several models.

3.3.3 Graph Convolutional Network (GCN)

GCNs[63] are a type of neural network designed to operate on graph-structured data. GCNs are designed to handle irregular graph data, where entities (nodes) are connected by relationships (edges). GCNs work by aggregating information from a node's neighbors to update the node's representation, which can then be used for various tasks such as node classification, link prediction, or graph classification.

Working of GCNs

In a GCN, the update for each node is computed by aggregating the features from its neighbors and passing this through a transformation layer. The process can be summarized as follows:
1. Aggregation: Each node gathers information from its neighboring nodes, typically by summing or averaging their features.
2. Transformation: The aggregated features are multiplied by a learnable weight matrix, transforming the features into a new space.
3. Non-linearity: A non-linear activation function, such as ReLU, is applied to introduce non-linearity.
Stacking multiple layers of this operation allows GCNs to capture information from increasingly distant nodes, enabling them to model both local and global graph structure.

The update rule[64] for a single GCN layer can be expressed as:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

Where:

- $H^{(l)}$ is the matrix of node features at layer l,
- $\tilde{A}$ is the adjacency matrix with added self-loops,
- $\tilde{D}$ is the degree matrix corresponding to $\tilde{A}$,
- $W^{(l)}$ is the layer's learnable weight matrix,
- σ is a non-linear activation function.

Advantages of GCNs

1. Efficient Neighborhood Aggregation: GCNs capture information from a node's local neighborhood, making them effective for tasks like node classification and link prediction.
2. Parameter Efficiency: Shared weights across the graph reduce the number of parameters, improving generalization and reducing overfitting.
3. Scalability: GCNs can process large graphs by stacking layers, enabling them to capture global structure efficiently.

Limitations of GCNs

1. Over-smoothing: Stacking too many layers causes node embeddings to become indistinguishable, reducing model performance.

2. Limited Expressiveness: GCNs primarily capture local information, struggling with complex or long-range dependencies.
3. Graph Quality Dependence: GCN performance is sensitive to noisy or incomplete graph structures.

3.3.4 RGCN

RGCN[65] stands for **Relational Graph Convolutional Network**, which is an extension of the Graph Convolutional Network (GCN) designed to handle *multi-relational graphs*. In such graphs, the edges can have different types, representing different kinds of relationships between nodes. RGCN[66] was introduced to model and learn from graphs where entities (nodes) are connected by multiple types of edges (relations), which is common in knowledge graphs, social networks, and other graph-structured data.

In an RGCN, the graph's edges are labeled with specific relations, indicating different types of interactions between nodes. For example, in a knowledge graph, a node might represent a person, and edges could represent various relationships like "friend of" or "works at." In standard GCNs, there's only one type of edge, so a single weight matrix is used for all nodes. In RGCNs, each relation type has its own weight matrix, allowing the network to learn how different relations affect the node representations differently.

Similar to GCNs, RGCNs also work with node features (attributes associated with each node). However, they also incorporate the types of relations when performing the convolution operation over the graph.

The **forward propagation** for an RGCN layer is generally described as:

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

Where:

- $h_i^{(l+1)}$ is the hidden state of node v_i in the layer $l+1$ of nueral network,
- R is the set of relations,
- $\mathcal{N}_i^r$ denotes the set of neighbor indices of node i connected by relation r,
- $W_r^{(l)}$ is the weight matrix corresponding to relation r at layer l,
- $W_0^{(l)}$ is a shared weight matrix applied to the node's own features (self-loop),
- $c_{i,r}$ is a problem specific normalization constant that can either be learned or chosen in advance,

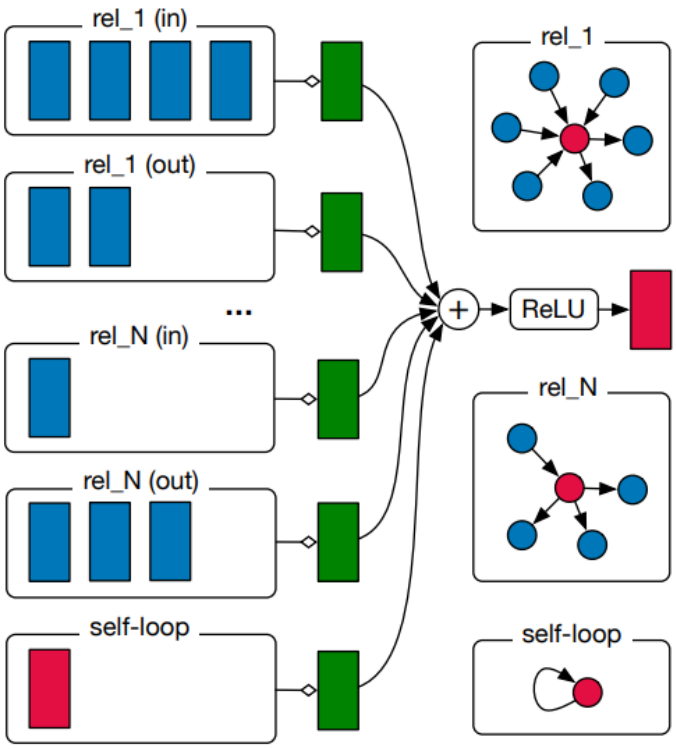


Figure 27: Diagram for computing the update of a single graph node/entity (red) in R-GCN model.

- σ is the activation function (like ReLU).

In Figure27, the update computation for a single graph node (in red) within the R-GCN model is illustrated. Activations, represented as d-dimensional vectors, are collected from neighboring nodes (in dark blue) and transformed separately for each relation type, accounting for both incoming and outgoing edges. The resulting representation (in green) is then accumulated in a (normalized) sum and passed through an activation function, such as ReLU. This update process is applied to each node in parallel, using shared parameters across the entire graph.

Advantages of RGCN:

1. Modeling Complex Relations: RGCNs are particularly effective at handling heterogeneous graphs where entities are connected by different types of edges.
2. Scalability: By focusing on local neighborhoods and relation-specific transformations, RGCNs are scalable to large graphs.

Limitations of RGCN

1. Overfitting: Due to the large number of parameters (one set of weights per relation), RGCNs can be prone to overfitting, especially when there are many relation types but limited data.
2. Computational Complexity: More relations require more weight matrices, which can increase computational demands.

Chapter 4

Experimental Comparison of Link Prediction Methods on RNA-KG

In this chapter, we delve into the experimental evaluation of various link prediction methods applied to the RNA-centric knowledge graph (RNA-KG). Building on the graph's complex structure and biological insights, we systematically compare the performance of different embedding and graph neural network-based models. The experimental setup focuses on node embedding extraction, edge prediction, and classification tasks, using models such as DistMult, ComplEx, and GNN variants like RGCN. By leveraging these techniques, we aim to measure predictive accuracy across multiple views of RNA-KG, providing a comprehensive understanding of their effectiveness in capturing biological relationships and supporting biomedical discovery. This evaluation lays the groundwork for identifying the most robust approaches to uncovering novel insights within RNA-related data.

4.1 Experimental Set Up

4.1.1 Server Specifications

All the experiments were run on the Google Colab Pro environment, utilizing a NVIDIA T4 GPU with ≈1.5 GHz of processing speed and 15 GB of GPU RAM. The environment also provides 12.7 GB of system RAM and 235.7 GB of disk space.

4.1.2 Embedding Model and Parameters

This chapter use multiple embedding models, i.e DistMult, ComplEx, GCN and RGCN on RNA-KG views. Each model was evaluated on both node classification and link prediction tasks. After generating the node embeddings, these embeddings were normalized using

a `StandardScaler` to ensure that features were on the same scale. A StandardScaler is a data preprocessing technique used in machine learning to normalize features by removing the mean and scaling them to unit variance. It ensures that the data has a mean of 0 and a standard deviation of 1. The data was then split into training and testing sets in an 80/20% ratio. Node types were encoded using both integer and one-hot encoding, allowing for a consistent and standardized input representation across models.

For the node classification task, a Random Forest Classifier was trained using the generated node embeddings to predict their respective labels. The classifier's performance was assessed based on the accuracy score, and classification report. Similarly, for link prediction, embeddings representing pairs of nodes were used to predict the existence of edges between them. This allowed for the evaluation of how well the embeddings captured structural relationships within the graph.

.

4.1.3 Experimental Parameters

All experiments shared several common parameters, such as the use of random seed initialization (python seed 42) to ensure reproducibility of results. The training and testing datasets were split using an 80/20% ratio for all models. This split ensured that a majority of the data was used for training while reserving a smaller portion for testing and validation purposes.

The dimensionality of the embeddings was fixed at 64 across the RGCN, GCN, DistMult, and ComplEx models, providing a uniform baseline for evaluating performance. Both RGCN and GCN utilized the Adam optimizer [67] with a learning rate of 0.01, while DistMult and ComplEx used learning rates of 0.001, which enabled a fair comparison across the different graph representation learning approaches. Regularization techniques such as dropout were applied consistently across models to prevent overfitting.

4.1.4 Node type prediction

Tables 5 and 6 present the results of node classification tasks for predicting node types (miRNA, disease, and gene) using four models: RGCN, GCN, DistMult, and ComplEx. The evaluation metrics include accuracy, precision, recall, and F1-score.

In table 5, RGCN consistently achieves the best performance across all views (View 1, View 4, View 5) and node types, demonstrating near-perfect metrics and superior ability to capture graph relationships. GCN performs well, particularly in View 1, but its performance slightly declines in Views 4 and 5, especially for gene nodes. DistMult and ComplEx exhibit lower overall performance, with DistMult struggling in precision and F1-scores for miRNA and gene nodes, and ComplEx showing limitations in handling complex relationships, especially for these node types.

In table 6, RGCN maintains its superior performance across all subgraph evaluations, achieving near-perfect metrics. GCN performs consistently well but shows reduced recall for disease and gene nodes in Views 4 and 5, indicating sensitivity to subgraph structures. DistMult remains less effective, with particular struggles in precision for disease nodes in View 4. ComplEx performs well for disease nodes but has lower recall for miRNA and gene nodes, reflecting its limitations with diverse relationships.

In general, RGCN consistently outperforms other models in both full graph and subgraph evaluations, making it the most suitable choice for node type prediction tasks. GCN also shows strong performance but exhibits some variability across views.

Table 5: Node Type Prediction Evaluation of Different Views

Models/Views	Accuracy	Precision			F1-Score			Recall		
		miRNA	Disease	Gene	miRNA	Disease	Gene	miRNA	Disease	Gene
RGCN										
View 1	0.95	1.00	0.98	0.99	1.00	0.98	0.99	1.00	0.98	0.99
View 4	0.95	0.99	0.97	0.99	0.99	0.97	0.99	1.00	0.98	0.99
View 5	0.96	0.99	0.97	0.99	0.99	0.98	0.99	0.98	0.98	0.99
GCN										
View 1	0.95	0.99	0.95	0.99	0.99	0.96	0.99	0.99	0.96	0.99
View 4	0.92	0.98	0.97	0.96	0.99	0.96	0.97	0.99	0.96	0.98
View 5	0.99	0.99	0.94	0.97	0.99	0.94	0.98	0.99	0.95	0.99
DistMult										
View 1	0.62	0.95	0.62	0.98	0.94	0.66	0.98	0.93	0.70	0.98
View 4	0.63	0.97	0.62	0.96	0.95	0.68	0.97	0.92	0.76	0.98
View 5	0.66	0.97	0.63	0.96	0.94	0.69	0.97	0.92	0.77	0.98
ComplEx										
View 1	0.63	0.99	0.53	1.00	0.99	0.60	0.99	0.99	0.70	0.99
View 4	0.62	0.99	0.53	0.98	0.99	0.61	0.98	0.98	0.72	0.98
View 5	0.62	1.00	0.53	0.98	0.99	0.61	0.98	0.98	0.72	0.99

4.1.5 Comparison of the Models Trained on RNA-KG views

This section presents a comparative evaluation of the four models—RGCN, GCN, DistMult, and ComplEx—focusing on their performance and the quality of node embeddings. Figures 28–30 show the plots comparing the various models for the node prediction task. The goal is to highlight how each model captures the structural and relational aspects of the graph, impacting link prediction and node classification tasks.

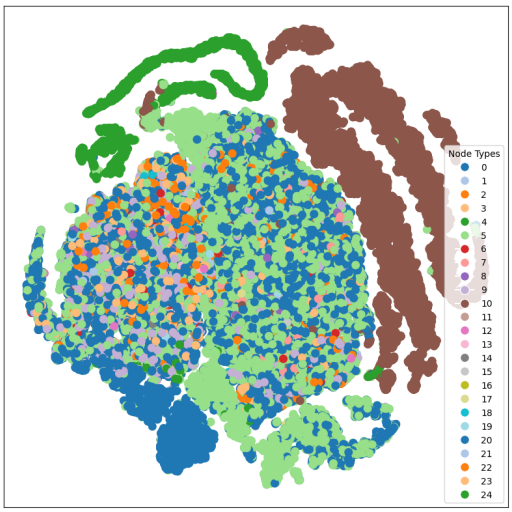
We employed t-SNE to visualize the embeddings because it reduces their dimensionality therefore enabling visualization in a two-dimensional space. This method helped us assess how well the different models cluster and separate node types, as well as their

Table 6: Node Type Prediction Evaluation of Subgraphs of Different Views

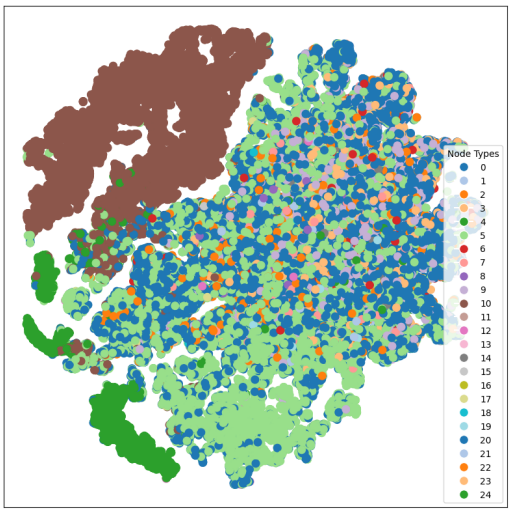
Models/Views	Accuracy	Precision			Recall		
		miRNA	Disease	Gene	miRNA	Disease	Gene
RGCN							
View 1	0.99	1.00	0.99	0.99	1.00	0.99	0.99
View 4	0.99	1.00	0.99	0.88	1.00	0.99	0.99
View 5	0.99	1.00	0.99	0.96	0.99	0.99	0.99
GCN							
View 1	0.99	0.99	0.99	0.99	0.99	0.99	0.99
View 4	0.98	0.99	0.99	0.97	0.99	0.98	0.99
View 5	0.99	0.99	0.99	0.98	0.99	0.98	0.99
DistMult							
View 1	0.97	0.96	0.98	0.97	0.91	0.98	0.97
View 4	0.97	0.96	0.98	0.96	0.91	0.98	0.97
View 5	0.97	0.98	0.98	0.96	0.92	0.98	0.98
ComplEx							
View 1	0.99	0.99	0.98	0.99	0.98	0.99	0.98
View 4	0.98	0.99	0.98	0.98	0.97	0.98	0.98
View 5	0.98	0.99	0.98	0.98	0.97	0.99	0.98

effectiveness in capturing graph structures. By comparing these visualizations, we can evaluate each model's ability to represent entities and relationships, providing insights into their strengths and weaknesses for downstream tasks like node classification and link prediction.

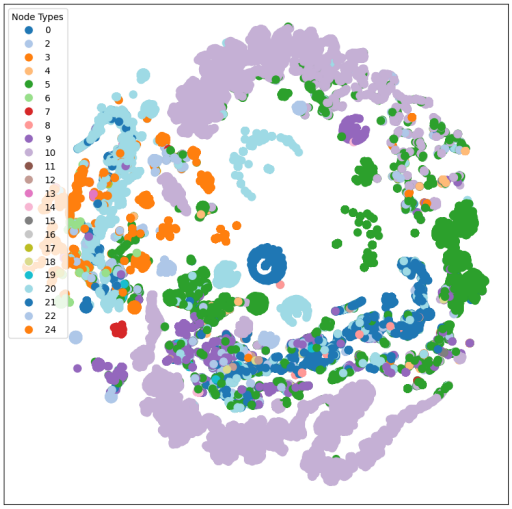
We conclude GNN based models such as GCN and RGCN have better performance than ComplEx and DistMult.



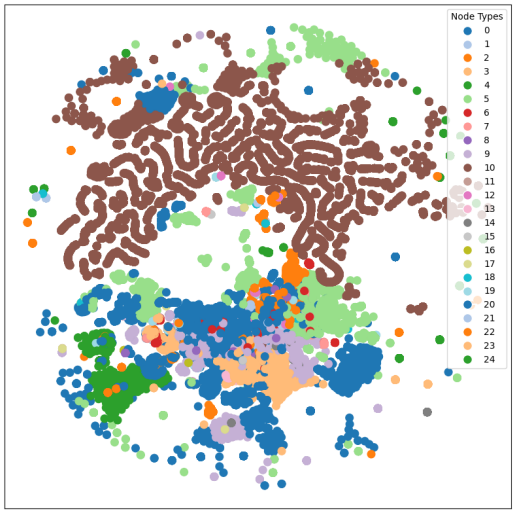
(a) ComplEx



(b) DistMult

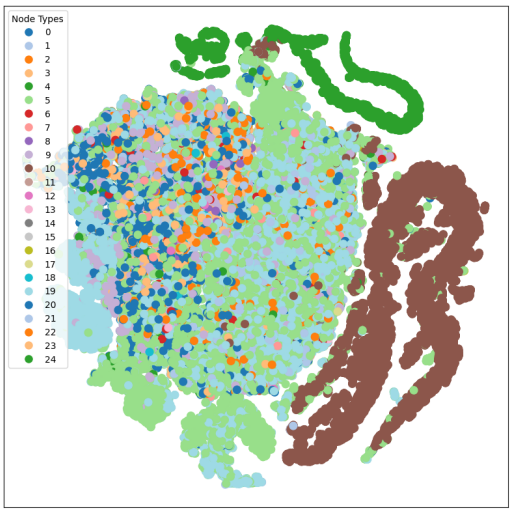


(c) GCN

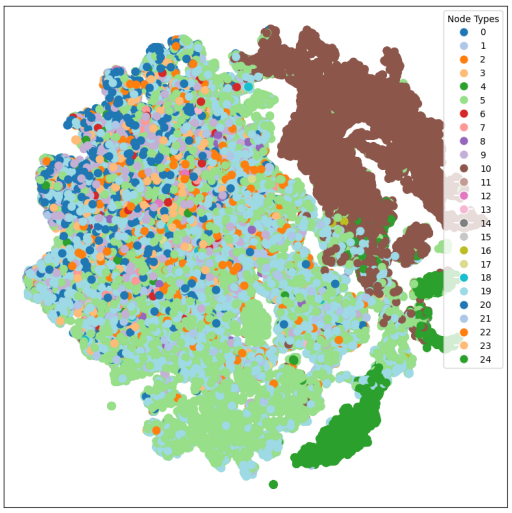


(d) RGCN

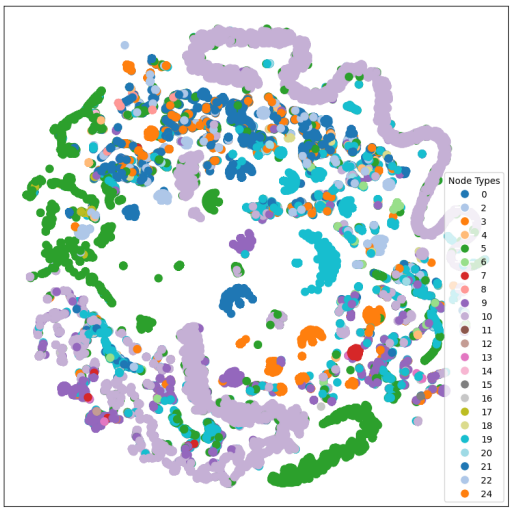
Figure 28: Comparison of Different Embeddings on VIEW1



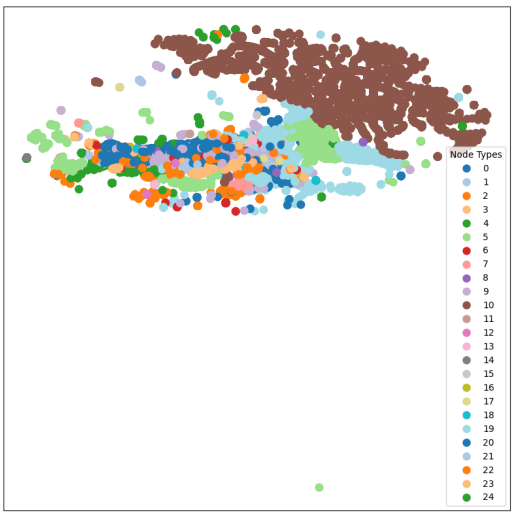
(a) ComplEx



(b) DistMult

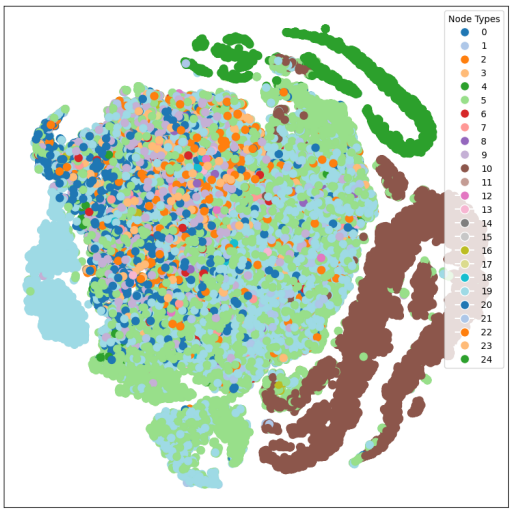


(c) GCN

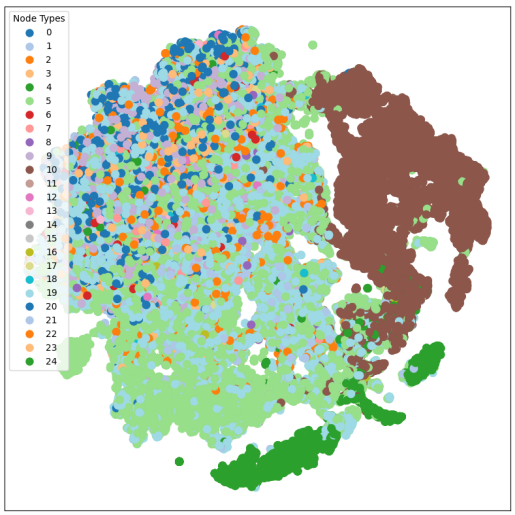


(d) RGCN

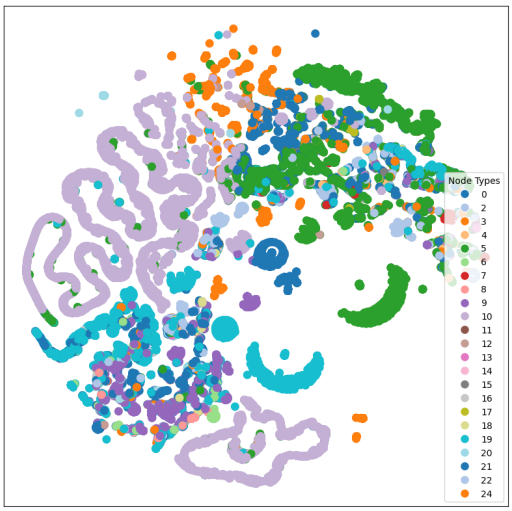
Figure 29: Comparison of Different Embeddings on VIEW4



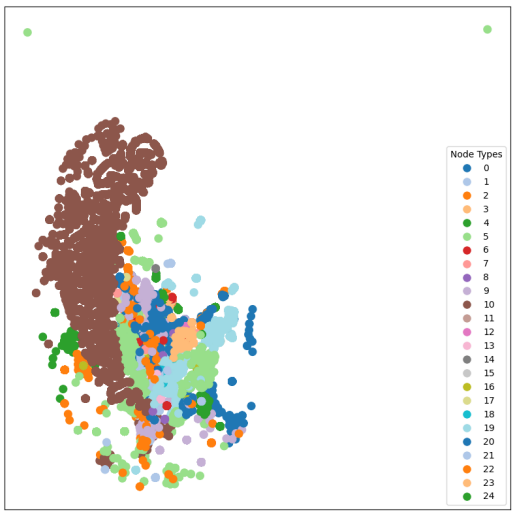
(a) ComplEx



(b) DistMult



(c) GCN



(d) RGCN

Figure 30: Comparison of Different Embeddings on VIEW5

4.2 Generic Edge prediction

Generic edge prediction focuses on identifying potential links between entities in a knowledge graph, independent of the specific edge types. In the context of RNA-KG, this involves predicting interactions or relationships among biological entities like RNAs, genes, proteins, and diseases. This task is critical for uncovering novel connections that can drive

biomedical discoveries, particularly in understanding complex biological networks. The section evaluates various methods and models to determine their efficiency and accuracy in performing edge prediction tasks across heterogeneous and multi-relational data.

4.2.1 Embedding Model and Parameters

The embeddings produced by DistMult, ComplEx, RGCN, and GCN models are then utilized as inputs for edge prediction. All experiments follow a 5-fold cross-validation setup, with an 80/20 split for training and testing, and a fixed random seed to ensure reproducibility (python seed = 42).

4.2.2 Edge Prediction Models and Parameters

The key parameters used for the edge prediction task with Random Forest and Decision Tree algorithms are summarized in Tables 7 and 8.

Table 7: Parameters for the decision tree model used for the edge prediction task.

Parameter	Value
edge_embedding_methods	Concatenate
use_scale_free_distribution	True
training_unbalanced_rate	1
criterion	Gini
splitter	Best
max_depth	10
min_samples_split	2
min_samples_leaf	1
max_features	sqrt
min_impurity_decrease	0.0
ccp_alpha	0.0
random_seed	42
train_size	0.8

4.2.3 Experimental Parameters

Random Forest and Decision Tree models were employed for classifying plausible links not covered by RNA-KG and discard improbable ones. Two node embeddings are concatenated to represent relationships between the two nodes.

Table 8: Parameters for the random forest model used for the edge prediction task.

Parameter	Value
edge_embedding_methods	Concatenate
use_scale_free_distribution	True
training_unbalanced_rate	1
max_depth	10
max_features	sqrt
n_estimators	1000
n_jobs	-1
random_seed	42
train_size	0.8

Random Forest and Decision Tree models were configured with the following parameters: the edge embedding method was set to "Concatenate," (The process of combining feature vectors from two nodes connected by an edge in a graph. Specifically, for an edge between nodes u and v, their individual feature vectors $\mathbf{h}_u$ and $\mathbf{h}_v$ are concatenated into a single feature vector $[\mathbf{h}_u||\mathbf{h}_v]$, where $[||]$ denotes concatenation.) and a scale-free distribution was used. The training unbalanced rate was set to 1, and the model was trained with a maximum depth of 10 and sqrt as the maximum number of features considered for splitting. A total of 1000 estimators were used, with parallel processing enabled for efficiency. The training set size was 80% of the data, and a fixed random seed (python seed = 42) ensured reproducibility.

Decision Trees offered better interpretability but tended to overfit. In contrast, Random Forest demonstrated improved robustness by combining predictions from multiple trees, thereby reducing variance and delivering slightly higher accuracy along with balanced metrics such as informedness and likelihood ratios. Despite its advantages, Random Forest incurred higher computational costs and reduced interpretability compared to the simpler Decision Tree model, which was faster to train and easier to understand. Overall, Random Forest provided superior generalization and predictive reliability, albeit with increased resource demands.

4.2.4 Embedding/GNN-like Methods Comparison

The random forest train and test metrics along with decision tree train and test metrics for different views in mentioned in tables 9– 20 and the evaluation of edge prediction using random forest and decision tree on different views across different models is shown in figures 31– 36 .

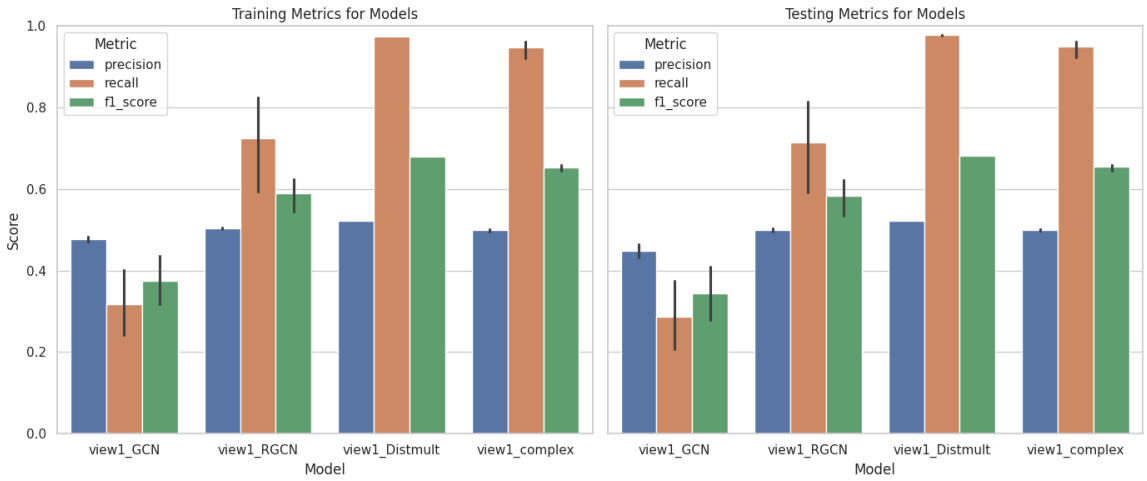


Figure 31: Evaluation of Random forest edge prediction on View1 across different models.

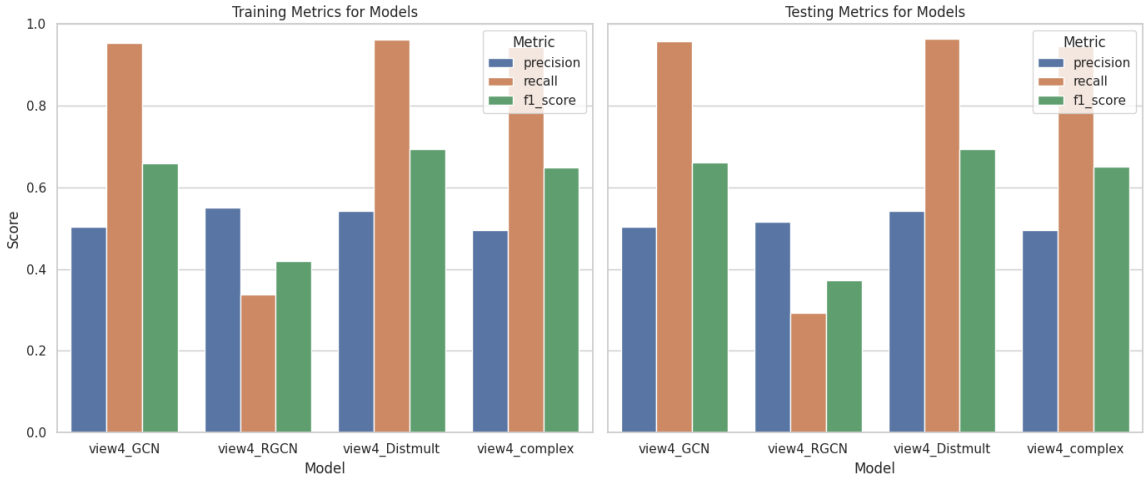


Figure 32: Evaluation of Random forest edge prediction on View4 across different models.

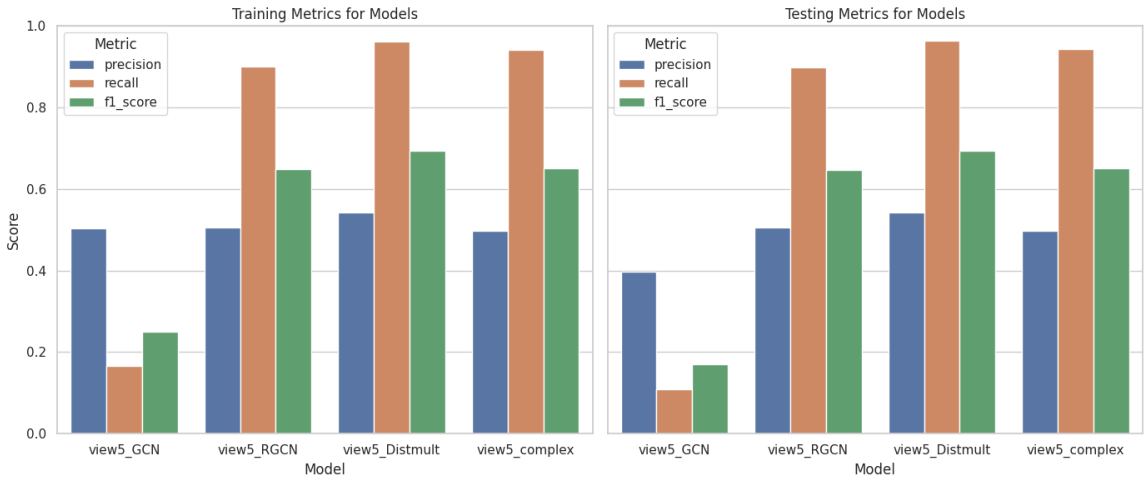


Figure 33: Evaluation of Random forest edge prediction on View5 across different models.

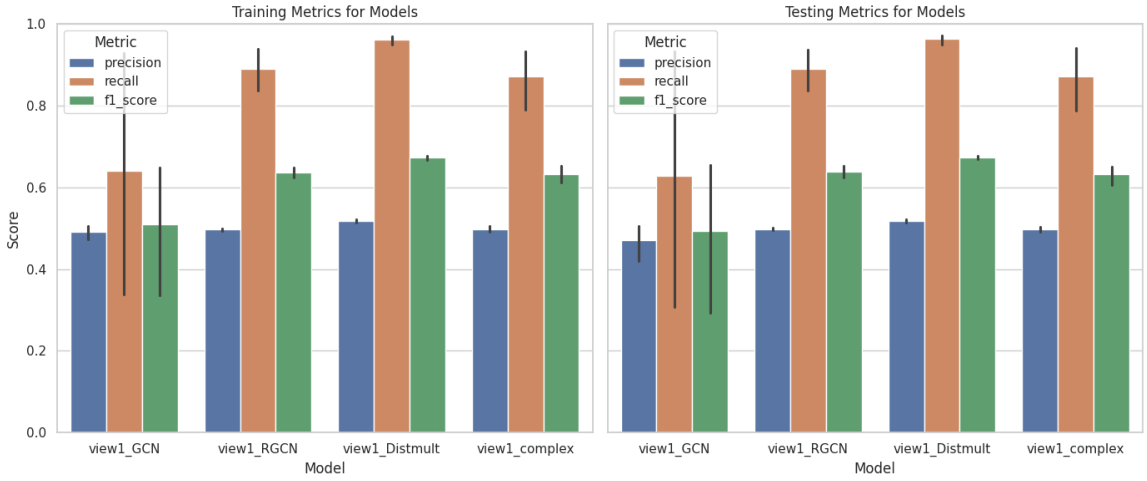


Figure 34: Evaluation of decision tree edge prediction on View1 across different models.

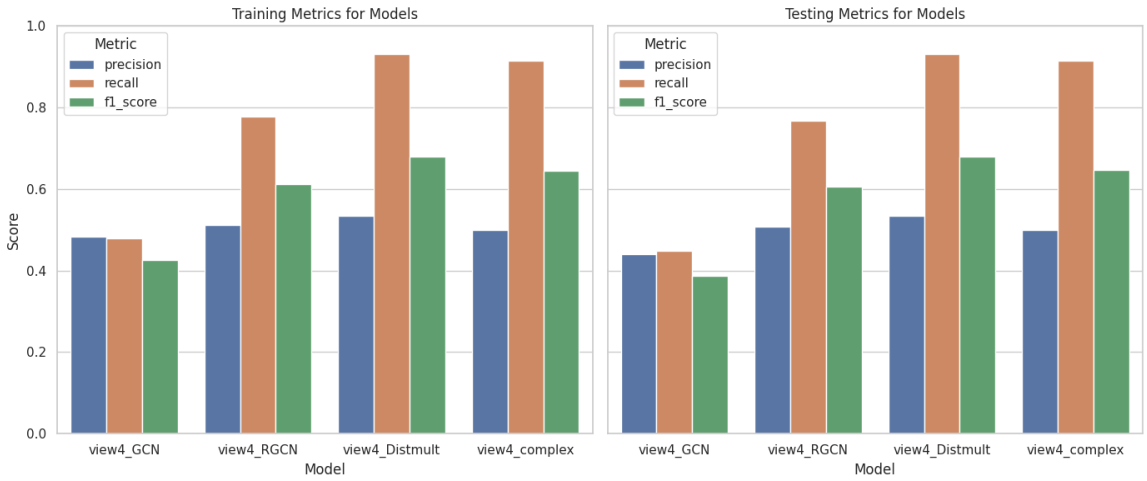


Figure 35: Evaluation of decision tree edge prediction on View4 across different models.

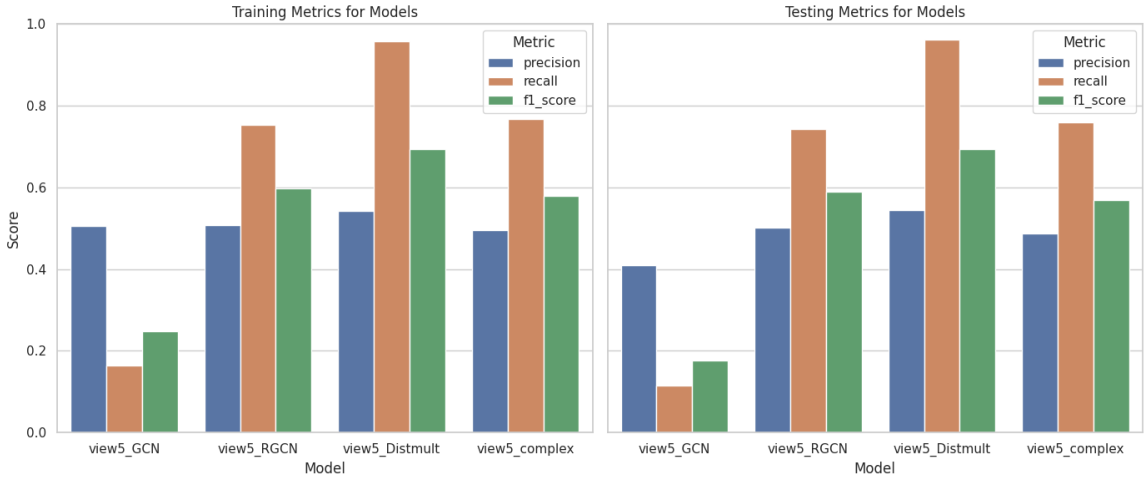


Figure 36: Evaluation of decision tree edge prediction on View5 across different models.

Table 9: Random Forest Train Metrics for View 1

Model	Precision	Recall	F1 Score
RGCN	0.50	0.72	0.59
GCN	0.48	0.32	0.37
DistMult	0.52	0.97	0.68
Complex	0.50	0.95	0.65

Table 10: Random Forest Test Metrics for View 1

Model	Precision	Recall	F1 Score
RGCN	0.50	0.71	0.58
GCN	0.45	0.29	0.34
DistMult	0.52	0.98	0.68
Complex	0.50	0.95	0.65

Table 11: Random Forest Train Metrics for View 4

Model	Precision	Recall	F1 Score
RGCN	0.55	0.34	0.42
GCN	0.50	0.95	0.66
DistMult	0.54	0.96	0.69
Complex	0.50	0.94	0.65

Table 12: Random Forest Test Metrics for View 4

Model	Precision	Recall	F1 Score
RGCN	0.52	0.29	0.37
GCN	0.50	0.96	0.66
DistMult	0.54	0.96	0.69
Complex	0.50	0.95	0.65

Table 13: Random Forest Train Metrics for View 5

Model	Precision	Recall	F1 Score
RGCN	0.51	0.90	0.65
GCN	0.50	0.17	0.25
DistMult	0.54	0.96	0.69
Complex	0.50	0.94	0.65

Table 14: Random Forest Test Metrics for View 5

Model	Precision	Recall	F1 Score
RGCN	0.51	0.90	0.65
GCN	0.40	0.11	0.17
DistMult	0.54	0.96	0.69
Complex	0.50	0.94	0.65

Table 15: Decision Tree Train Metrics for View 1

Model	Precision	Recall	F1 Score
RGCN	0.50	0.89	0.64
GCN	0.49	0.64	0.51
DistMult	0.52	0.96	0.67
Complex	0.50	0.87	0.65

Table 16: Decision Tree Test Metrics for View 1

Model	Precision	Recall	F1 Score
RGCN	0.50	0.89	0.64
GCN	0.47	0.64	0.49
DistMult	0.52	0.96	0.67
Complex	0.50	0.87	0.63

Table 17: Decision Tree Train Metrics for View 4

Model	Precision	Recall	F1 Score
RGCN	0.51	0.78	0.61
GCN	0.48	0.48	0.43
DistMult	0.53	0.93	0.68
Complex	0.50	0.91	0.65

Table 18: Decision Tree Test Metrics for View 4

Model	Precision	Recall	F1 Score
RGCN	0.51	0.77	0.61
GCN	0.44	0.45	0.39
DistMult	0.53	0.93	0.68
Complex	0.50	0.92	0.65

Table 19: Decision Tree Train Metrics for View 5

Model	Precision	Recall	F1 Score
RGCN	0.51	0.75	0.60
GCN	0.50	0.16	0.25
DistMult	0.54	0.96	0.69
Complex	0.50	0.77	0.58

Table 20: Decision Tree Test Metrics for View 5

Model	Precision	Recall	F1 Score
RGCN	0.50	0.74	0.59
GCN	0.41	0.11	0.18
DistMult	0.54	0.96	0.69
Complex	0.49	0.76	0.57

Chapter 5

Conclusions and Future Work

This thesis investigated the performance of two neural network-based models, **Graph Convolutional Networks (GCN)** and **Relational Graph Convolutional Networks (RGCN)**, alongside two non-neural models, **DistMult** and **ComplEx**, in link prediction tasks on an RNA-centered Knowledge Graph (**RNA-KG**). The goal was to evaluate their effectiveness in capturing complex relationships within heterogeneous biological data.

Among the neural models, **RGCN** outperformed **GCN** in scenarios involving diverse and multi-relational data. RGCN's ability to handle different types of relationships with separate weight matrices enabled it to capture relational complexity more effectively, making it suitable for the RNA-KG structure. **GCN**, while efficient and straightforward, struggled with multi-relational scenarios, limiting its effectiveness compared to RGCN.

For the non-neural models, **ComplEx** demonstrated better performance than **DistMult**, particularly in handling asymmetric relationships. By leveraging complex-valued embeddings, ComplEx provided a richer representation of interactions within RNA-KG. **DistMult**, while computationally simpler and effective for symmetric relationships, lacked the expressiveness needed for capturing more nuanced and directional interactions.

Overall, **RGCN** emerged as the best-performing model across all categories, showcasing its robustness in modeling heterogeneous and multi-relational data. Its ability to model complex interactions made it particularly effective in extracting meaningful insights from RNA-related data. **ComplEx** also proved to be a strong contender, especially when computational efficiency was prioritized, but it was less effective in capturing the depth of relational patterns compared to RGCN.

These findings underscore the trade-offs between neural and non-neural approaches: while neural models such as **RGCN** excel in modeling relational intricacies and provide better overall accuracy, non-neural models like **ComplEx** offer computational simplicity and are well-suited for resource-constrained scenarios.

Future research could explore several promising directions. One potential avenue is

the optimization of **RGCN** for large-scale graphs, addressing its computational and memory constraints to enhance scalability. Another intriguing possibility is the development of hybrid approaches that combine the strengths of neural models like **RGCN** with the efficiency of non-neural models such as **ComplEx**. Additionally, incorporating advanced techniques like attention mechanisms or contrastive learning could further refine the performance of graph-based models on RNA-KG. These advancements would not only improve link prediction tasks but also expand the utility of RNA-KG in biomedical research, paving the way for new discoveries in RNA biology and personalized medicine.

Bibliography

[1] Horacio G. Pontis. Chapter 14 - case study: Sugar phosphates. In Horacio G. Pontis, editor, *Methods for Analysis of Carbohydrate Metabolism in Photosynthetic Organisms*, pages 191–203. Academic Press, Boston, 2017.

[2] IUPAC. nucleotide bases. *Pure and Applied Chemistry*, 2019.

[3] Lin Liu, Zhao Li, Chang Liu, Dong Zou, Qianpeng Li, Changrui Feng, Wei Jing, Sicheng Luo, Zhang Zhang, and Lina Ma. LncRNAWiki 2.0: a knowledgebase of human long non-coding RNAs with enhanced curation model and database system. *Nucleic Acids Research*, 50(D1):D190–D195, 2022.

[4] Agnese Loda and Edith Heard. Xist rna in action: Past, present, and future. *PLoS genetics*, 15(9):e1008333, 2019.

[5] Chandrasekhar Kanduri. Kcnq1ot1: a chromatin regulatory rna. *Seminars in Cell & Developmental Biology*, 22(4):343–350, June 2011.

[6] Matthias K. Vorländer, Belén Pacheco-Fiallos, and Clemens Plaschka. Structural basis of mrna maturation: Time to put it together. *Current Opinion in Structural Biology*, 75:102431, August 2022.

[7] Jeff Ross. mrna stability in mammalian cells. *Microbiological reviews*, 59(3):423–450, 1995.

[8] Robin R Gutell, Nils Larsen, and Carl R Woese. Lessons from an evolving rrna: 16s and 23s rrna structures from a comparative perspective. *Microbiological reviews*, 58(1):10–26, 1994.

[9] Eric M Phizicky and Anita K Hopper. trna biology charges to the front. *Genes & development*, 24(17):1832–1860, 2010.

[10] Yimei Cai, Xiaomin Yu, Songnian Hu, and Jun Yu. A brief review on the mechanisms of mirna regulation. *Genomics, Proteomics and Bioinformatics*, 7(4):147–154, 2009.

[11] Eric Batsché and Maya Ameyar-Zazoua. The influence of argonaute proteins on alternative rna splicing. *Wiley Interdisciplinary Reviews: RNA*, 6(1):141–156, 2015.

[12] U Thomas Meier. The daunting task of modifying ribosomal rna. *RNA*, 28(12):1555–1557, 2022.

[13] Gudrun Böhmdorfer and Andrzej T Wierzbicki. Control of chromatin structure by long noncoding rna. *Trends in cell biology*, 25(10):623–632, 2015.

[14] Jennifer A Doudna and Jon R Lorsch. Ribozyme catalysis: not different, just worse. *Nature structural & molecular biology*, 12(5):395–402, 2005.

[15] Nan Li, Zhihao Yang, Ling Luo, Lei Wang, Yin Zhang, Hongfei Lin, and Jian Wang. Kghc: a knowledge graph for hepatocellular carcinoma. *BMC Medical Informatics and Decision Making*, 20(S3), July 2020.

[16] TJ Callahan et al. An open source knowledge graph ecosystem for the life sciences. *Scientific Data*, 11, 2024.

[17] E. Cavalleri and et al. Rna-kg: An ontology-based knowledge graph for representing interactions involving rna molecules, 2023.

[18] Eric Prud'hommeaux and Andy Seaborne. Sparql query language for rdf - w3c recommendation, 2018. Available at `https://www.w3.org/TR/rdf-sparql-query/`.

[19] Emanuele Cavalleri, Sara Bonfitto, Alberto Cabri, Jessica Gliozzo, Paolo Perlasca, Mauricio Soto-Gomez, Gabriella Trucco, Elena Casiraghi, Giorgio Valentini, and Marco Mesiti. A meta-graph for the construction of an rna-centered knowledge graph. In Ignacio Rojas, Olga Valenzuela, Fernando Rojas Ruiz, Luis Javier Herrera, and Francisco Ortuño, editors, *Bioinformatics and Biomedical Engineering*, pages 165–180, Cham, 2023. Springer Nature Switzerland.

[20] Wes McKinney and PD Team. Pandas-powerful python data analysis toolkit. *Pandas—Powerful Python Data Analysis Toolkit*, 1625, 2015.

[21] P. N. Robinson and et al. The human phenotype ontology: A tool for annotating and analyzing human hereditary disease. *The American Journal of Human Genetics*, 83(5):610–615, 2008.

[22] M. Ashburner and et al. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

[23] Y. He and et al. Vo: Vaccine ontology. Nature Precedings, 2009.

[24] K. Degtyarenko and et al. Chebi: a database and ontology for chemical entities of biological interest. *Nucleic Acids Research*, 36:D344–D350, 2007.

[25] C. J. Mungall, C. Torniai, G. V. Gkoutos, S. E. Lewis, and M. A. Haendel. Uberon, an integrative multi-species anatomy ontology. *Genome Biology*, 13:R5, 2012.

[26] S. Sarntivijai and et al. Clo: The cell line ontology. *Journal of Biomedical Semantics*, 5:37, 2014.

[27] D. A. Natale and et al. The protein ontology: a structured representation of protein forms and complexes. *Nucleic Acids Research*, 39:D539–D545, 2010.

[28] K. Eilbeck and et al. The sequence ontology: a tool for the unification of genome annotations. *Genome Biology*, 6, 2005.

[29] V. Petri and et al. The pathway ontology - updates and applications. *Journal of Biomedical Semantics*, 5:7, 2014.

[30] Chris Mungall et al. oborel/obo-relations: 2023-08-18 release. Zenodo, 2023.

[31] L. Cappelletti, T. Fontana, E. Casiraghi, V. Ravanmehr, T.J. Callahan, C. Cano, M.P. Joachimiak, C.J. Mungall, P.N. Robinson, J. Reese, and G. Valentini. Grape for fast and scalable graph processing and random walk-based embedding. `https://github.com/AnacletoLAB/grape`, 2023. Accessed: 2024-11-21.

[32] Phillip Bonacich and Philip Lu. *12. Scale-Free Networks*, pages 117–136. Princeton University Press, Princeton, 2012.

[33] Tiffany Callahan. Pheknowlator: A knowledge graph generation tool. `https://github.com/callahantiff/PheKnowLator`, 2024. Accessed: 2024-11-16.

[34] Python Knowledge Toolkit (PKT-KG). pkt-kg: A python library for knowledge graph operations. `https://pypi.org/project/pkt-kg/`, 2024. Accessed: 2024-11-16.

[35] Claude Berge. *The theory of graphs*. Courier Corporation, 2001.

[36] Fenxiao Chen, Yun-Cheng Wang, Bin Wang, and C.-C. Jay Kuo. Graph representation learning: a survey. *APSIPA Transactions on Signal and Information Processing*, 9:e15, 2020.

[37] Towards Data Science. Introduction to machine learning with graphs, 2023. Accessed: 2024-09-25.

[38] C. H. Ding, X. He, H. Zha, M. Gu, and H. D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 107–114. IEEE, 2001.

[39] William L Hamilton. *Graph representation learning*. Morgan & Claypool Publishers, 2020.

[40] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, 2018.

[41] 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.

[42] M Shimrat. Embedding in homogeneous spaces. *The Quarterly Journal of Mathematics*, 5(1):304–311, 1954.

[43] Xiao Wang, Deyu Bo, Chuan Shi, Shaohua Fan, Yanfang Ye, and S Yu Philip. A survey on heterogeneous graph embedding: methods, techniques, applications and sources. *IEEE Transactions on Big Data*, 9(2):415–436, 2022.

[44] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[45] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[46] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):1–23, 2019.

[47] Guillaume Salha, Romain Hennequin, and Michalis Vazirgiannis. Keep it simple: Graph autoencoders without graph convolutional networks. *arXiv preprint arXiv:1910.00942*, 2019.

[48] John Doe and Jane Smith. Example title of the paper. *arXiv preprint arXiv:2304.05055*, 2023.

[49] Zhenyu Yang, Jian Xie, Feng Liu, Xiang Wang, Qi Zhang, and Jian Guo. A unified framework for link prediction with graph neural networks. *arXiv preprint arXiv:2204.01855*, 2022.

[50] DataCamp. Comprehensive introduction to graph neural networks (gnns), 2024. Accessed: 2024-09-25.

[51] Xiao Wang, Deyu Bo, Chuan Shi, Shaohua Fan, Yanfang Ye, and Philip S. Yu. A survey on heterogeneous graph embedding: Methods, techniques, applications and sources. *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[52] Chuxu Zhang, Mengqi Zhang, Chao Huang, and Xia Hu. Heterogeneous graph neural network, 2021. Accessed: 2024-09-25.

[53] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Serrano, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc., 2013.

[54] Tao-yang Fu, Wang-Chien Lee, and Zhen Lei. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1797–1806, 2017.

[55] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 135–144, 2017.

[56] Kenneth Ward Church. Word2vec. *Natural Language Engineering*, 23(1):155–162, 2017.

[57] Thiviyan Thanapalasingam, Lucas van Berkel, Peter Bloem, and Paul Groth. Relational graph convolutional networks: a closer look. *PeerJ Computer Science*, 8:e1073, 2022.

[58] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 793–803, 2019.

[59] B. Yang, W. T. Yih, X. He, J. Gao, and L. Deng. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations (ICLR)*, 2014.

[60] M. Nickel, V. Tresp, and H. P. Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 11, pages 809–816, 2011.

[61] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, and G. Bouchard. Complex embeddings for simple link prediction. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.

[62] 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 (ICLR)*, 2019.

[63] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153*, 2019.

[64] Thomas Kipf. Graph convolutional networks. `https://tkipf.github.io/graph-convolutional-networks/`, 2016. Accessed: 2024-09-27.

[65] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. *arXiv preprint arXiv:1703.06103*, 2017. arXiv:1703.06103.

[66] Thiviyan Thanapalasingam, Lucas van Berkel, Peter Bloem, and Paul Groth. Relational graph convolutional networks: A closer look. *arXiv preprint arXiv:2107.10015*, 2021. arXiv:2107.10015.

[67] GeeksforGeeks. Adam optimizer. `https://www.geeksforgeeks.org/adam-optimizer/`, 2023. Accessed: 2024-11-21.

Project developed at AnacletoLab
http://anacletolab.di.unimi.it