Università degli Studi di Milano
FACOLTÀ DI SCIENZE E TECNOLOGIE

DIPARTIMENTO DI INFORMATICA
GIOVANNI DEGLI ANTONI

CORSO DI LAUREA MAGISTRALE IN
INFORMATICA

A COMPARISON OF HETEROGENEOUS LINK PREDICTION METHODS ON AN RNA-CENTERED KNOWLEDGE GRAPH

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

Candidate:
Khushboo Chauhan
Student ID: 14380A

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 deepest gratitude to my supervisor, Prof. Marco Mesiti, for his unwavering support and guidance throughout every stage of this thesis. His expertise, insightful feedback, and open-minded approach played a crucial role in shaping this work. His brilliant ideas and constructive critiques helped me navigate both the scientific and computational aspects of the research, making even the most complex concepts more approachable. I am also incredibly thankful to the entire team at Anacleto Lab for fostering such a collaborative and inspiring environment, where valuable discussions and fresh perspectives enriched my research experience.

A heartfelt thank you to Dr. Emanuele Cavalleri, my co-supervisor, whose constructive criticism and thoughtful suggestions greatly enhanced the quality of this thesis. His keen eye for detail and encouragement pushed me to refine my work and reach new heights.

I owe a special thanks to my family, whose unwavering support and motivation have been my foundation throughout this journey. My heartfelt appreciation goes to my siblings, Rupanshi and Tanya, whose encouragement and belief in me kept me grounded and focused. Their strength and constant reassurance were invaluable. This thesis is as much a result of their support as it is of my effort, and for that, I am profoundly grateful.

Contents

Chapter 1

Introduction

The rapid advancements in RNA-based technologies have profoundly revolutionized biomedical research, enabling significant breakthroughs in understanding gene expression, disease mechanisms, and therapeutic interventions. Central to these advancements is the ability to predict interactions within RNA-centered knowledge graphs, which integrate diverse biological entities and their relationships. The rich and varied information across RNA-KG's nodes and edges suggests the use of methods specifically developed for heterogeneous graphs [1]. Link prediction in such heterogeneous networks is crucial for uncovering novel biological insights and facilitating drug discovery.

Traditional link prediction methods, while effective in homogeneous networks, often fall short in capturing the complex, multi-relational nature of RNA-centered knowledge graphs. The multi-dimensional relationships between biological entities present unique challenges that require sophisticated computational approaches. Recent developments in machine learning, particularly in embedding extraction techniques such as TRANSE, RotatE etc, offer promising solutions by leveraging the rich structural and semantic information inherent in these graphs.

This study aims to provide a comprehensive comparison of various heterogeneous link prediction methods applied to an RNA-centered knowledge graph. We focus on node embedding extraction techniques using TRANSE, Rotate, and RDF2Vec, and evaluate them based on precision accuracy and recall .Further, evaluating link prediction accuracy, for node type classification. These evaluations not only measure the effectiveness of each embedding extraction method but also assess their potential in predicting node types, providing a multifaceted view of their performance.

The thesis is organized in five chapters. The first chapter introduces the motivation and objectives of the thesis. Chapter 2 RNA-KG delves into the construction and the characteristics of the RNA-KG , also explaining the data sources, ontologies and methodologies used to create the RNA-KG. Chapter 3 Graph representation learning techniques focuses on the various models and methods for homogeneous and heterogeneous graphs and why

the methods used in heterogeneous differs from homogeneous. Chapter 4 presents the experimental setup and the results comparing the different models in node classification and for edge prediction. Here we discuss the non-neural methods and a neural network based model (GATCONV) and discuss there performance in the task of edge prediction. The final chapter summarizes the findings of the study and the future research.

Chapter 2

RNA-KG: A Knowledge Graph tailored for RNA Molecules

The study of RNA holds significant promise for understanding biological processes and developing novel therapeutics, as demonstrated by the success of mRNA vaccines for COVID-19 [2]. Non-coding RNAs (ncRNAs) encompass a wide range of RNA species [3], and the vast data generated by genomics labs showcases diverse interactions between ncRNAs and other bio-entities (e.g., genes, proteins, and diseases). Integrating this data can drive knowledge discovery and the development of RNA-based drugs.

However, extracting and organizing this information requires significant effort. Knowledge Graphs (KG [4]) were taken into use as an effective way to organize interconnected knowledge across various domains. KGs facilitate the integration of heterogeneous information extracted from diverse data sources with the aim of highlighting complex interdependencies and uncovering hidden relationships. They can be modeled using property graphs (e.g., Neo4j [5]) or RDF [6], each with its own advantages [7]. A KG structured with an ontology consists of a schema (TBox) and factual data (ABox), supporting reasoning with languages like OWL [8] and SPARQL [9]. RNA-KG, built using RDF triples from over 60 public sources, serves as a foundation for studying RNA molecules and their interactions. It leverages a meta-graph framework grounded according to the Relation Ontology (RO [10]), which ensures common semantics for the different relationships that can be extracted from the sources. Section 1 of this chapters describes about the preliminary considerations like RNA molecule, biomedical KGs and ontologies before diving into RNA-KG. Next section describes the steps considered for the creation of the RNA-KG and the last section informs about the generated RNA-KG and its key data .

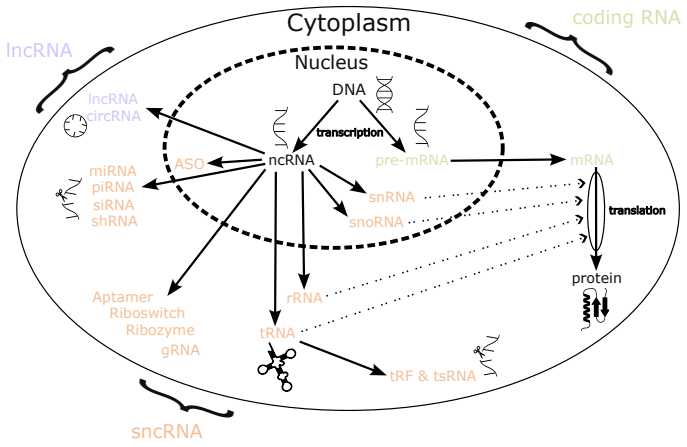


Figure 1: Schematic representation of the RNA network within a cell

2.1 Preliminary Considerations for RNA-KG Development

2.1.1 RNA Molecules

RNA molecules play a crucial role in cell biology and hold the potential for novel therapeutics that could revolutionize the treatment and prevention of human diseases [11]. RNAs possess several biological functions: they regulate gene expression [12], exhibits enzymatic activity [13], modifies or regulates other RNAs and biomolecules [14], and are translated into proteins by ribosomes [15].

Eukaryotic messenger RNA **(mRNA)** undergoes processing to achieve its mature, protein-encoding form, which is then translated into amino acids by ribosomes [16]. **Non-coding RNAs (ncRNAs)**, which are not translated into proteins, are divided into two groups: **long non-coding RNAs (lncRNAs)** and **small non-coding RNAs (sncRNAs)**. **lncRNAs** are critical in disease regulation, affecting processes such as **competitive endogenous RNA (ceRNA) regulation**, transcription, and epigenetics [17]. They modulate chromatin, nuclear bodies, and mRNA stability, and can regulate transcription factors or co-regulators [18]. lncRNAs, such as circular RNAs **(circRNAs)**, also influence splicing and DNA methylation via interactions with enzymes like DNMTs and TETs, playing roles in both normal physiological processes (e.g., Xist in X-chromosome inactivation [19]) and diseases, including cancer [20]. Figure 1 represents a clear schemantic representation of the RNA within a cell representing the main classes in RNAs.

Figure 2: Example of biomedical KG.

2.1.2 Biomedical knowledge graphs

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 2 provides an example of KG in the biomedical domain [21]. 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 [22]. 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 3 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

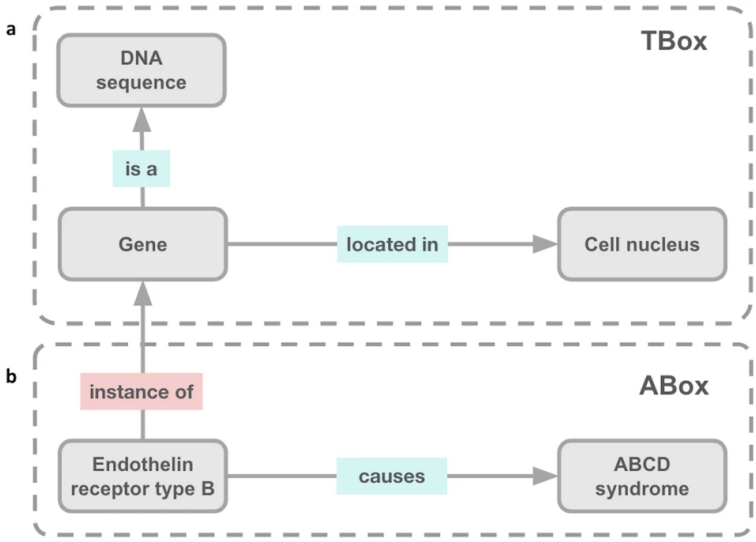


Figure 3: Example of knowledge modelling within a KG.

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.

2.1.3 Biomedical ontologies

In computer and information science, the term "ontology" refers to hierarchical representations of concepts within a specific domain, such as biomedicine, that are based on relationships. Ontologies are utilized in the semantic web to establish a shared vocabulary that facilitates reuse and reapplication across computer systems, enhancing both efficiency and flexibility, as well as supporting natural language processing. The ontologies considered during RNA-KG construction are show in Table 1. These term and hierarchical structures are commonly accepted by scientific community to unequivocally describe biological classes and entities such as diseases, phenotypes, chemicals, biological processes, proteins, and relations between them.

Name	Abbr.	Description
Human Phenotype Ontology [23]	HPO	Terms representing medically relevant phenotypes and disease-phenotype annotations.
Gene Ontology [24]	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 [25]	VO	Terms in the domain of vaccines and vaccination.
Chemical Entities of Biological Interest [26]	ChEBI	Structured classification of molecular entities of biological interest focusing on "small" chemical compounds.
Uber-anatomy Ontology [27]	Uberon	Terms representing body parts, organs, and tissues in a variety of animal species, with a focus on vertebrates.
Cell Line Ontology [28]	CLO	Terms representing publicly available cell lines.
PROtein Ontology [29]	PRO	Terms representing protein-related entities (including specific modified forms, orthologous isoforms, and protein complexes).
Sequence Ontology [30]	SO	Terms representing features and properties of nucleic acid used in biological sequence annotation.
Pathway Ontology [31]	PW	Terms for annotating gene products to pathways.
Relation Ontology [10]	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).

2.2 The creation of RNA-KG

RNA-KG [32] is the first ontology-based KG specifically designed to represent both coding and non-coding RNA molecules, along with their interactions with other biomolecular data, pathways, abnormal phenotypes, and diseases. Figure 4 shows the workflow considered for the creation of RNA-KG.

RNA sources characterization

The first phase involves identifying relevant RNA data sources, downloading the data, and performing necessary pre-processing steps. An extensive research was carried out which resulted in identification of more than 60 public repositories dealing with RNA sequences and annotations developed by well-established organizations, published in top journals,

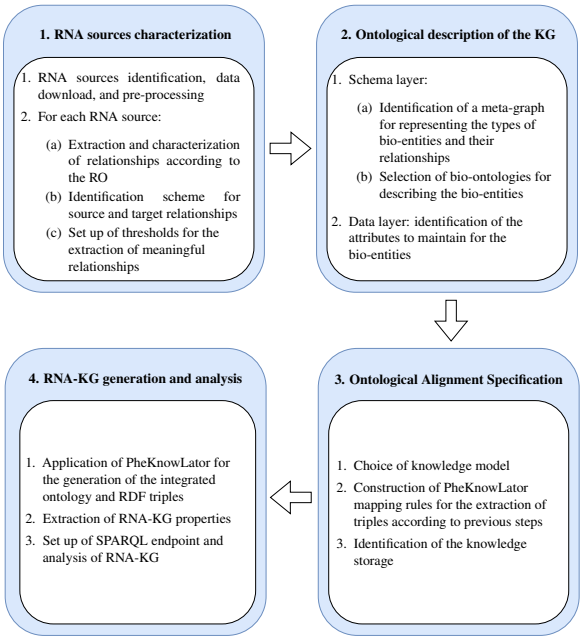


Figure 4: Workflow of construction of RNA-KG

and regularly updated. The data extracted was transformed into a common (TSV files) format and the inconsistencies were removed. Most of the sources used flat file formats, with only a few offering APIs or RDF/SPARQL access. The Relation Ontology (RO) was applied to standardize relationships between RNA molecules and bio-entities, enabling consistent descriptions of interactions at different levels of granularity.

Figure 6 shows the cardinality of relationships in RNA-KG, including miRNA-lncRNA interactions (150 million distinct interactions), followed by lncRNA-mRNA (~28 million), miRNA-mRNA/miRNA-gene (~6 million each), and protein-lncRNA (~800,000). RNA aptamer--disease interactions were rare due to the limited availability of RNA aptamer drugs. The integration of nRNA data from various sources revealed challenges related to mapping and duplication. Some data entries were omitted when mapping to a reference ontology was not feasible, and issues of duplicate entries, due to different identifiers for the same molecule, were noted.

Ontological description of KG

In the second phase, the bio-entities and relationships that needed to be represented in the knowledge graph were identified, focusing on the schema layer. This included determining the types of bio-entities to manage and the kinds of relationships that exist between them. Specific instances and properties (data layer) essential for constructing the KG were also highlighted. These include fundamental properties such as identifiers,

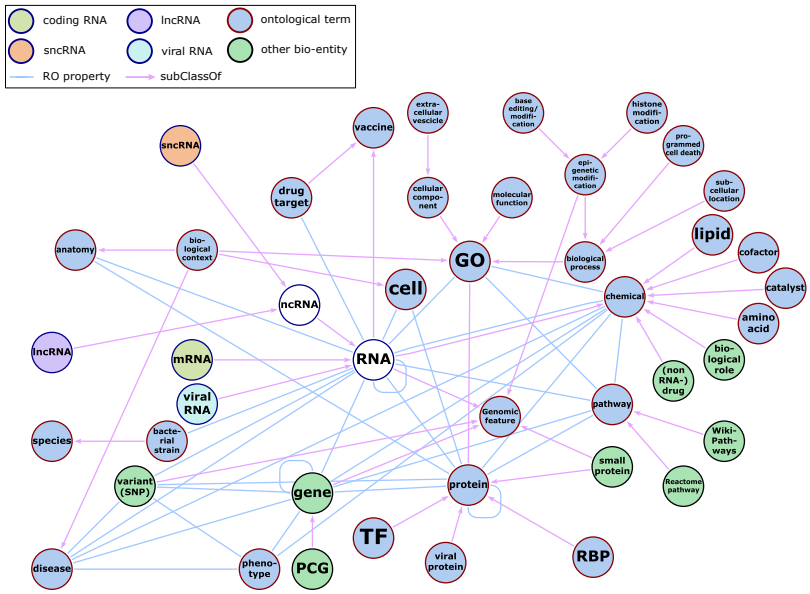


Figure 5: The complete conceptual RNA-KG meta-graph. RNA nodes are summarized into a few general types (e.g., ncRNA and mRNA) to simplify the visualization.

node types, and source provenance, which are crucial for avoiding excessive growth in KG size while ensuring a robust structure. Future updates may extend this to include class properties to enrich the data layer.

Building on the insights from the previous phase, a meta-graph was constructed to represent all the bio-entities and relationships [33]. The meta-graph provides bi-directional navigation by including both direct and inverse relationships. After identifying the key entities and relationships, the essential properties to retain for each entity were determined, focusing initially on fundamental characteristics like identifiers and provenance.

The general relationships *interacts with* available in RO with the meaning "A relationship that holds between two entities in which the processes executed by the two entities are causally connected" have been specified in the most specific relationships *molecularly interacts with* in the classification to represent the situation in which the two partners are molecular entities that directly physically interact with each other (e.g., via a stable binding interaction or a brief interaction during which one modifies the other). Figure 5 shows an abstract representation of meta-graph by clustering in a single RNA nodes connected with various bio-entities based on insights extracted from RNA sources and literature.

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).

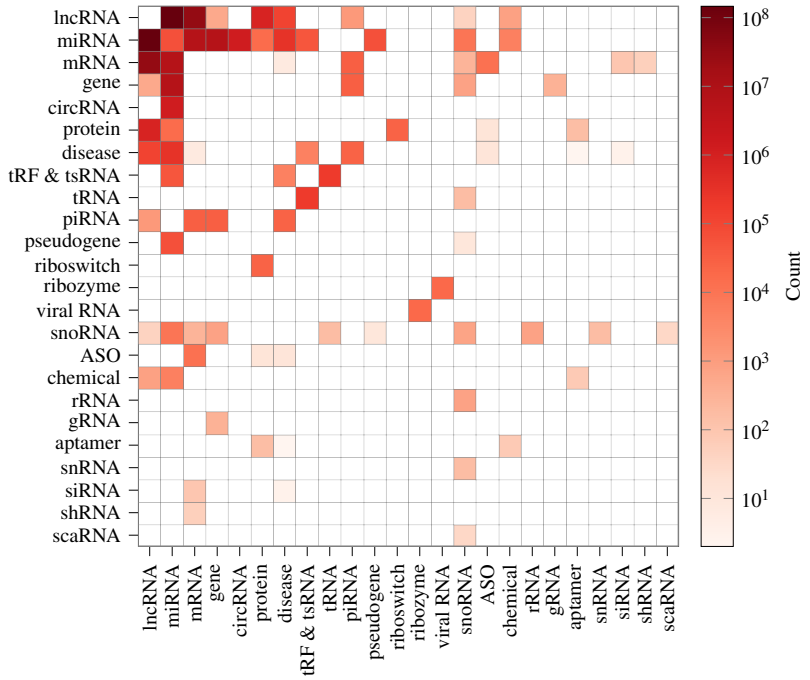


Figure 6: Number of relationships involving RNA molecules and relevant bio-entities (gene, protein, chemical, and disease) within the considered RNA sources. Colors represent the ranges of relationships in log scale, as reported in the legend.

Ontological alignment specification

To align entities and relationships to the schema defined in the previous step, the PheKnowLator (Phenotype Knowledge Translator [22]) ecosystem was adopted. PheKnowLator efficiently represents bio-entities and their relationships. It also simplifies the identification of columns containing molecular identifiers and the specification of their relationships according to the RO ontology. RNA-KG can be exported in various models using PheKnowLator, depending on the type of analysis. The instance-based, inverse relation, semantically abstracted (OWL-NETS [34]) configuration is considered the most appropriate version of RNA-KG to be processed by machine learning algorithms, especially for node and link prediction methods.

OWL-NETS (Ontology Web Language-Networks) abstracts biomedical knowledge into a simplified network format that preserves only biologically meaningful concepts and relationships, while filtering out irrelevant or overly complex details. The instance-based inverse relation model further enhances this by ensuring that inverse relationships (e.g., `located in` and its inverse `location of`) are automatically included, thus enabling more robust reasoning and bidirectional data exploration (Figure 8).

RNA molecules, which often lack detailed semantic characterizations in bio-ontologies,

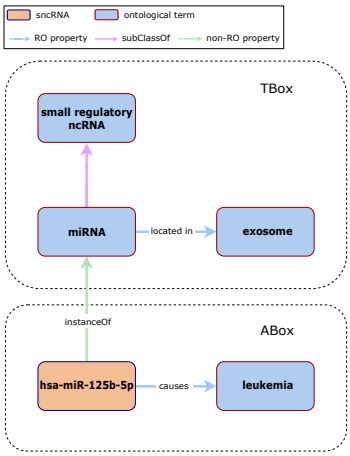


Figure 7: An example of the use of Description Logic (DL) for knowledge modeling. The TBox includes classes (i.e., `miRNA`, `small regulatory ncRNA`, and `exosome`), and the assertions between classes (i.e., "`miRNA subClassOf small regulatory ncRNA`" and "`miRNA is located in exosome`"). The ABox includes instances of classes (i.e., *hsa-miR-125b-5p*) represented in the TBox and assertions about those instances (i.e., "*hsa-miR-125b-5p* `instanceOf` miRNA" and "*hsa-miR-125b-5p* `causes` leukemia").

are represented as subClassOf specific ontological classes, providing a more structured and coherent KG for analysis. The subgraph (Figure 8) highlights the presence of inverse relationships (e.g., `located in` and its inverse `location of`) and the relation RDF subClassOf. PheKnowLator mapping rules specify how to extract triples from data sources, defining the relationships between entities such as mRNA and disease, and aligning them with RO terms (e.g., RO_0003302, which represents "causes or contributes to condition"). These rules allow for row filtering, thresholding, and transformation options during triple extraction.

Additionally, multiple ontologies were cleaned using PheKnowLator tools to remove deprecated entities, errors, and duplicates. The cleaned ontologies were then merged into a single file to define the structure of RNA-KG, ensuring alignment with the meta-graph.

RNA-KG generation and analysis

In the last phase, the mapping rules of PheKnowLator were used on the pre-processed data to generate the KG. After importing RNA-KG into GRAPE [35] library, GRAPE was exploited to implement different types of graph embedding techniques which can only be realized using this tool due to the size of generated KG. The entire RNA-KG can be downloaded at the following link `http://RNA-KG.anacleto.di.unimi.it`.

The current version of RNA-KG has a single connected component containing 673,825

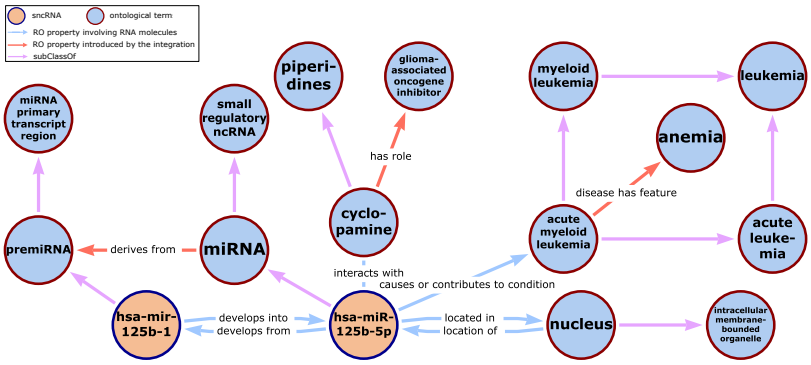


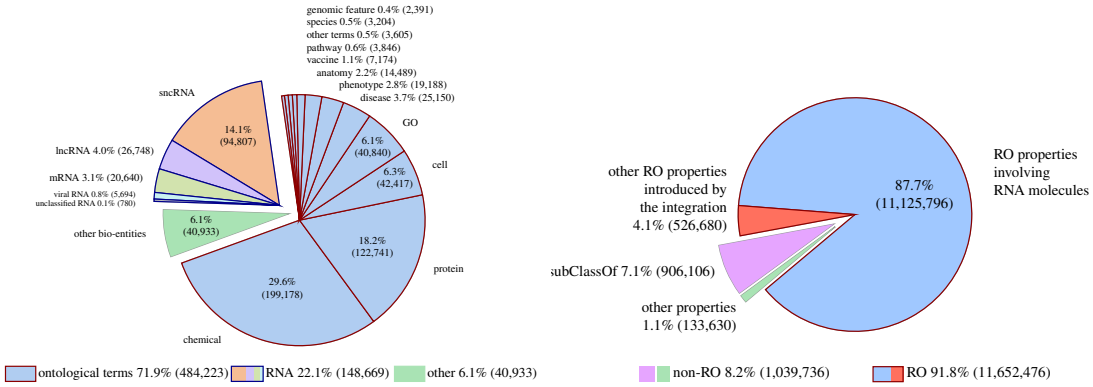
Figure 8: Example of a RNA-KG subgraph realized according to the instance-based, inverse relations, semantically abstracted (OWL-NETS without harmonization) parameters.

nodes and 12,692,212 edges. The number of nodes and edges has been substantially reduced by considering only the relationships with high reliability.

Node Distribution. As illustrated in Figures 9a and 9b, nodes in RNA-KG are categorized into bio-entities and ontological terms. Bio-entities are further divided into RNA nodes—comprising small non-coding RNAs (sncRNAs), messenger RNAs (mRNAs), long non-coding RNAs (lncRNAs), viral RNAs, and unclassified RNA nodes—and non-RNA nodes, which include genes and SNPs. Among these, miRNAs, mRNAs, and lncRNAs are prominently represented due to their extensive study. piRNAs, as are the most numerous sncRNA category. tRNA-derived fragments (tRFs) and riboswitches are also notably abundant. The unclassified RNA category encompasses 780 nodes lacking detailed characterization, while the "other" category includes less common sncRNA types. The overall RNA count is consistent with estimates of approximately 22,000 to 25,000 protein-coding genes in humans [36].

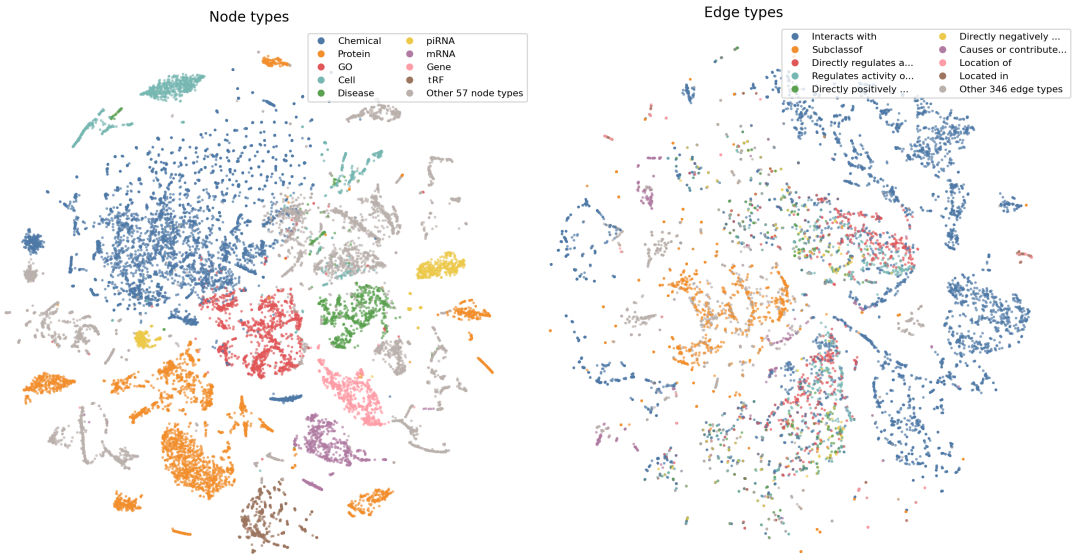
Edge Distribution. Edge types are divided into relationships among RNA molecules and RO properties, subclass relationships, and other relationships (e.g., "has gene template" from PRO). The "interacts with" edge type is most prevalent, indicating generic interactions, while "regulates activity of" edges are common due to the large number of miRNAs regulating genes and mRNAs. The distribution of subclass relationships underscores the integration of bio-ontologies, as each RNA molecule is classified under appropriate classes within the Sequence Ontology (SO)

t-SNE representation. Figure 10 represents the t-SNE representation of the embeddings of the nodes and edges in the complete RNA-KG. Figure 10a demonstrate that node embeddings cluster similar nodes together, indicating their functional relationships



(a) Distribution of nodes. **(b)** Distribution of edges.

Figure 9: Node and Edge distribution



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

Figure 10: Node and Edge Embeddings

within the network. Conversely, Figure 10b presents the edge embeddings for RNA-KG, which similarly reveal the similarities between edges. However, there are overlaps noted with the "interacts with" and "regulates activity of" relations, as the former is a generic relation between nodes.Furthermore, the different subcategories of "regulates activity of" related to miRNA and mRNA—such as "directly regulates activity of" and its specific subtypes—are not distinctly represented because of the homogeneous nature of the algorithm (Node2Vec with continuous bag of words (CBOW), a second-order embedding algorithm based on random walks with a walk length of 5), requiring the need to implement algorithms that can more effectively capture the heterogeneity inherent in RNA-KG.

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 [37]. 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 [32].

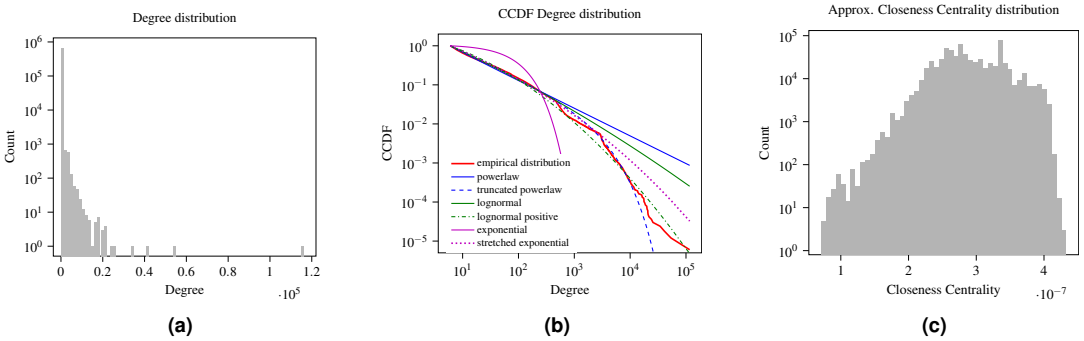


Figure 11: (a) Node degree distribution (semi-log). (b) Complementary cumulative distribution (CCDF) for the node degree. (c) Approximated closeness centrality distribution.

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.

2.3 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 [38, 39] 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 0. Nodes and edges within views 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 [22]. Detailed statistics for each view are presented in Figures 12–14.

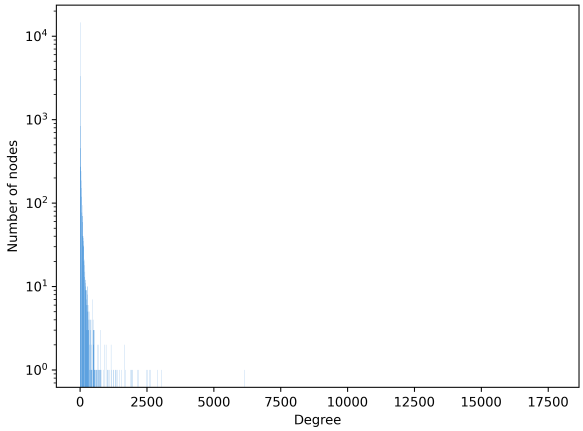
Degree distribution plots and cardinalities for node and edge types are reported when they are relevant to the downstream analysis. The degree distribution plot for each view

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 0	miRNA phenotype disease gene	miRNA-disease miRNA-phenotype miRNA-gene	Mondo HPO RO
View 2	miRNA phenotype disease gene	miRNA-disease miRNA-phenotype miRNA-gene	Mondo HPO SO RO
View 3	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.

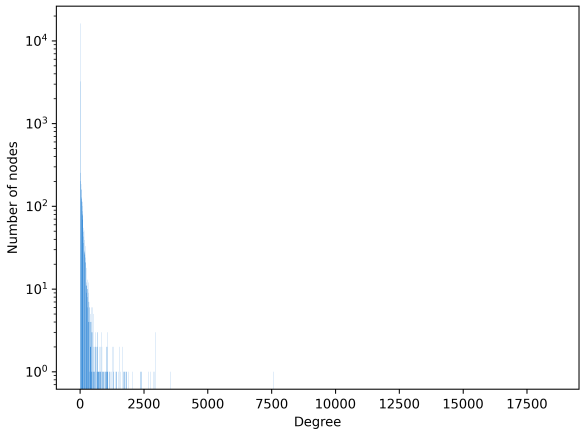
Figure 12: Node degree distribution and counts for nodes and edges in View 0.



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

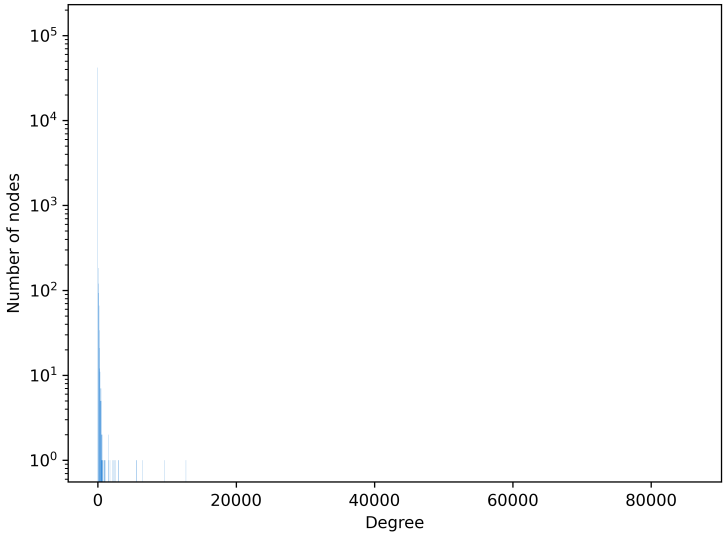
Figure 13: Node degree distribution and counts for nodes and edges in View 2.



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 14: Node degree distribution and counts for nodes and edges in View 3.



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

Graph representation learning is a significant task since it could facilitate various downstream tasks, such as node classification, link prediction etc. Graph representation learning aims to map graph entities to low-dimensional vectors while preserving graph structure and entity relationships. Over the decades, many models have been proposed for graph representation learning. In this chapter we represent a comprehensive picture of graph representation learning models mainly used in today's world. In this section we will discuss about homogeneous graphs and the graph representation methods for that, further extending it to heterogeneous graph and the related methods.

3.1 Homogeneous Graph Representation Learning

A homogeneous graph, denoted by $G = (V, E)$, consists of vertices $V = \{v_1, v_2, \ldots, v_n\}$ and edges $E = \{e_{ij}\}$, where an edge e_{ij} connects vertex v_i to vertex v_j. In a homogeneous graph, all vertices represent instances of the same type and all edges represent the same type of relation.

Graphs are typically represented by an adjacency matrix or a derived vector space representation [40]. The adjacency matrix A of a homogeneous graph G contains non-negative weights associated with each edge, $a_{ij} \geq 0$. If vertices v_i and v_j are not directly connected by an edge, then $a_{ij} = 0$. For undirected (edges having no direction) homogeneous graphs, $a_{ij} = a_{ji}$ for all $1 \leq i, j \leq n$.

.

3.1.1 Graph Representation Learning

Graph representation learning (GRL) is a crucial area in machine learning that focuses on learning low-dimensional embedding of graph-structured data. The primary goal is to capture the structural and semantic information of the graph in a way that can be effectively utilized for various downstream tasks, such as node classification, link prediction, and graph clustering To utilize graphs in data mining and downstream machine learning applications, it is necessary to represent graphs and its constituents, including edges and nodes, using numerical features. The adjacency matrix of a graph is one approach to depict it. However, because an adjacency matrix's size is |V| × |V|, it requires a lot of memory to represent very big graphs. Using their features, we may depict a graph and its constituent parts. In particular, a collection of features that could improve the representation's performance in a certain application can be applied to a node in the graph. Furthermore, manually extracting the features takes a lot of time. As a result, techniques like graph embedding [41] and node embedding have been put forth, addressing the problem and automatically producing representation vectors for graphs. These techniques use the structure and attributes of the graph as input data to build embedding vectors and formulate the graph representation learning as a machine learning challenge.

A **graph embedding** is a mapping of an entire graph $G = (V, E)$ or a subgraph $G[S]$ (where $S \subseteq V$) into a low-dimensional vector space $\mathbb{R}^d$. Let $G = (V, E)$ be an undirected graph, where:

- V is the set of vertices, $|V| = n$.
- E is the set of edges, $|E| = m$.

A graph embedding is defined as a function $f : G \rightarrow \mathbb{R}^d$ that assigns a d-dimensional vector $\mathbf{g} \in \mathbb{R}^d$ to the entire graph G or to a subgraph $G[S]$. The embedding can be represented as a vector:

$\mathbf{g} = f(G)$ or $\mathbf{g}_S = f(G[S])$

Node embedding , where G = (V , E) be a graph, where V and E are the set of nodes and the set of edges of the graph, respectively learns a mapping function $f : v_i \rightarrow \mathbb{R}^d$ that encodes each graph's node v_i into a low dimensional vector of dimension d such that d« |V| and the similarities between nodes in the graph are preserved in the embedding space. Node embedding techniques have shown a remarkable capacity to convert high-dimensional sparse graphs into low-dimensional, dense, and continuous vector spaces (see Figure 15), where structure properties are maximally preserved [42]. The main aim of graph embedding methods is to encode nodes into a latent vector space, i.e., pack every node's properties into a vector with a smaller dimension. In addition to node embedding, there are other types of embedding, e.g., edge embedding, substructure embedding, and whole-graph embedding. [42]

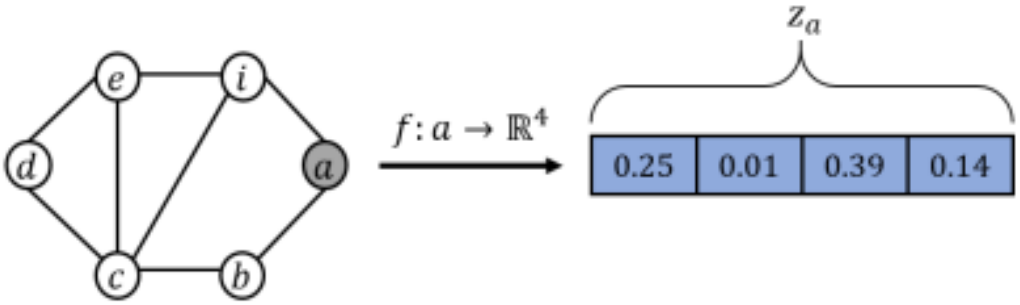


Figure 15: The graph on the left hand side consists of 6 nodes a, b, c, d, e, i and 8 edges. Graph embedding methods map each node of the graph into an embedding vector with dimension d. For the demonstration purpose, the node a is embedded into an embedding vector z_a of dimension 4 with given values.

.

Applications of Node Embedding The embedding vectors generated through graph embedding can be applied across various tasks, such as node classification, link prediction, and graph classification. Below, we provide an overview of some of these applications.

Node Classification: In the node classification task, labels are assigned to nodes within the test dataset. This approach is widely used in different fields. For example, in social networks, one might predict a person's political affiliation based on the affiliations of their friends. In node classification, each node in the training dataset is represented by its embedding vector, and the corresponding label is the node's category. Traditional classification techniques, such as Logistic Regression or Random Forests, can be trained on this data to generate classification scores for the test nodes. Similarly, graph classification can be executed using the graph's embedding vectors.(Figure16) .

Link Prediction: Link prediction is a key application of node embedding methods, aiming to estimate the likelihood of an edge forming between two nodes. This task is prevalent in scenarios like recommending friends in social networks or identifying potential biological connections in biological networks. Link prediction is often framed as a classification task, where an edge label of 1 indicates a likely connection between two nodes, and a label of 0 indicates otherwise. During training, a set of samples is created using both positive and negative examples. Positive examples are the existing edges in the graph, while negative examples are non-existent edges, whose representation can be derived from the node embeddings. As with node classification, any classification algorithm can be trained on this dataset to predict edge labels for test cases.(Figure17)

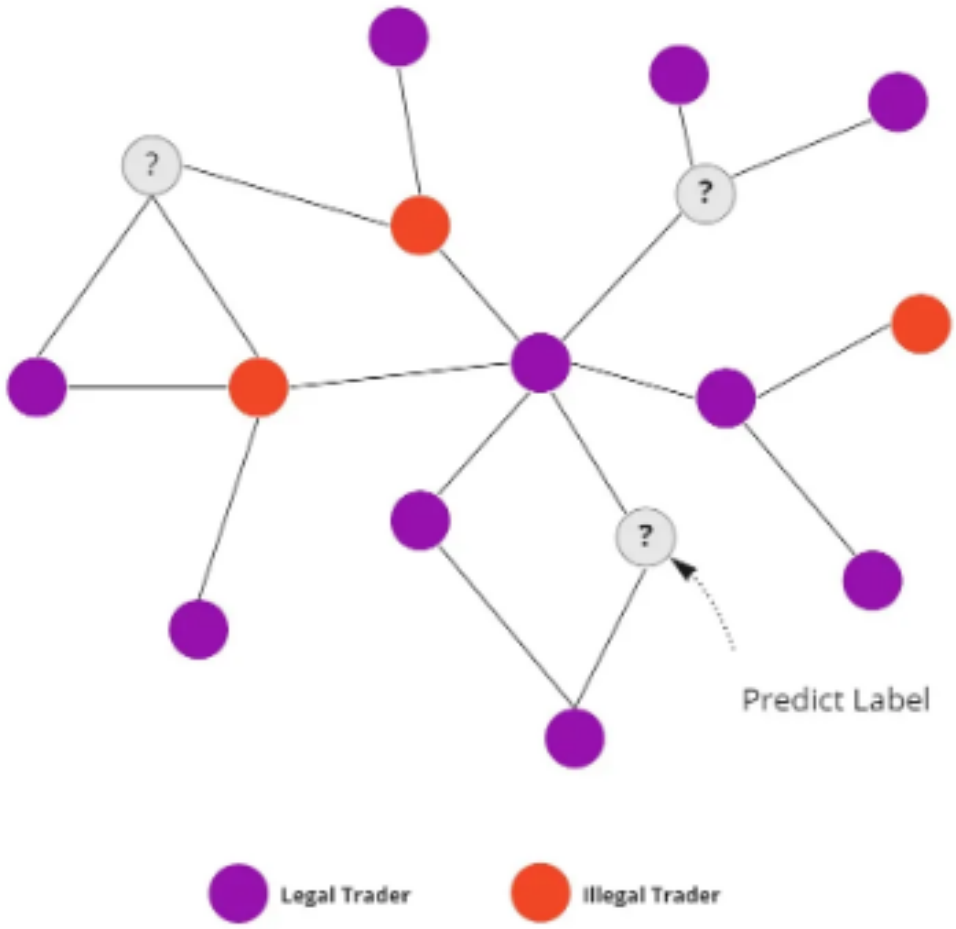


Figure 16: Given a graph with labeled and unlabeled nodes, predict the nodes without labels based on their node features and their neighborhood nodes.

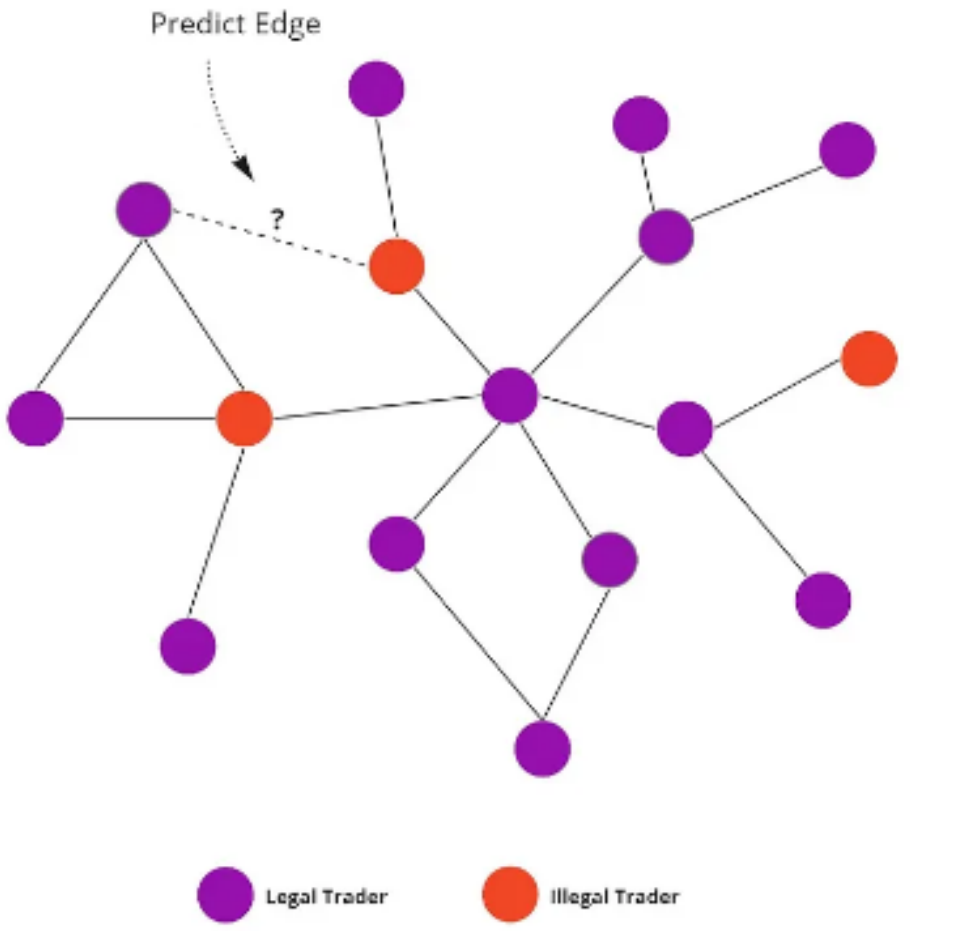


Figure 17: Given a node pair without having a link between them, predict if they become connected in the future based on their node features and neighborhood nodes.

Graph Clustering: Graph embeddings can be used for clustering tasks, such as detecting communities in social networks or identifying similar groups of proteins in biological networks. Methods like K-means can be applied to graph embeddings to group similar nodes, edges, or graphs.

Graph Visualization: Node embedding methods simplify the visualization of large graphs by mapping nodes to lower dimensions, making it easier to see nodes, edges, communities, and other graph properties. This helps researchers gain insights into complex graph data.

3.1.2 Homogeneous Approaches for GRL

Homogeneous approaches for graph representation learning (GRL) can be broadly categorized into several types. **Matrix factorization techniques**, such as Singular Value Decomposition (SVD) and Non-negative Matrix Factorization (NMF), decompose the graph's adjacency matrix into lower-dimensional representations, effectively capturing structural patterns within the graph. **Random walk-based methods**, including DeepWalk and Node2Vec, simulate random walks over the graph to generate sequences of nodes, treating them similarly to sentences in natural language processing. These sequences are then used to learn node embeddings that reflect both local and global graph structure. **Graph Neural Networks (GNNs)**, such as Graph Convolutional Networks (GCNs) and Graph Attention Networks (GATs), aggregate information from neighboring nodes to learn embeddings that capture both node features and graph topology. In particular, GATs extend GCNs by assigning attention weights to neighbors, allowing the model to focus on more relevant nodes during aggregation.

Another category is **kernel methods**, with the Weisfeiler-Lehman (WL) Kernel being a notable example. It maps graph structures into high-dimensional feature spaces by capturing subgraph patterns, aiding in graph comparison and classification tasks. Lastly, **autoencoder-based approaches**, such as Structural Deep Network Embedding (SDNE) and Graph Autoencoders and Variational Graph Autoencoders (VGAE), learn to encode the graph into a latent space while preserving structural properties. These models reconstruct the graph from the latent representation, producing compact and informative node embeddings. Together, these approaches effectively capture the structure and relationships within homogeneous graphs, enabling enhanced performance in tasks like node classification, link prediction, and community detection.

The Graphical learning methods for Homogeneous Graphs frequently used

1. **DeepWalk** is a technique for learning node representations in graphs by generating random walks. It begins by creating random walks [43] from each node, which represent sequences of neighboring nodes selected randomly. These walks capture the local

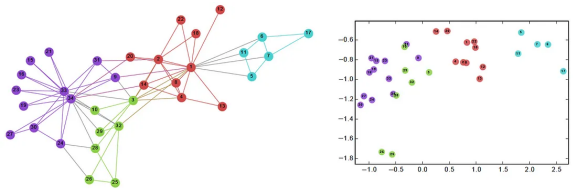


Figure 18: orignal Graph and its embedding

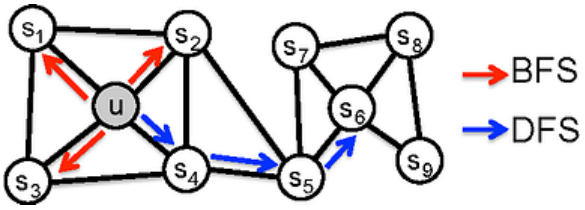


Figure 19: BFS and DFS

structure of the graph. The Word2Vec [44] algorithm, originally used in natural language processing [45], is then applied to these random walks. In this context, the walks act as "sentences" and the nodes as "words", Word2Vec allows for words to be embedded into n-dimensional space, with similar words being locally situated near each-other. This means that words that are often used together / used in similar situations would have smaller cosine distances (figurefig:Deepwalk). Word2Vec learns through a neural network (skip gram [46]) to predict the likelihood of a node appearing in a random walk based on its neighboring nodes. The resulting node representations can be utilized for various tasks, such as node classification and link prediction.

2. Node2Vec: builds on DeepWalk by refining the random walk generation process. It incorporates Depth-First Search (DFS) and Breadth-First Search (BFS) strategies, controlled by two parameters: P (return parameter) and Q (in-out parameter). A high P value leads to longer, exploratory walks, while a low P keeps the walks localized. Conversely, a low Q encourages exploration, while a high Q restricts the walks to local neighborhoods. This approach allows for more flexibility in capturing the graph's structure compared to DeepWalk. The parameter p prioritizes a breadth-first-search (BFS) procedure, while the parameter q prioritizes a depth-first-search (DFS) procedure. The decision of where to walk next . as in Figure 19 it is clear BFS is ideal learning local neighbours, and DFS is better for global variables, Node2vec can switch to and from the two priorities depending on the task

3. LINE : LINE (Large-scale Information Network Embedding) is a scalable graph embedding method designed to capture both first-order and second-order proximities in large-scale networks. **First-order proximity** refers to the direct connections between

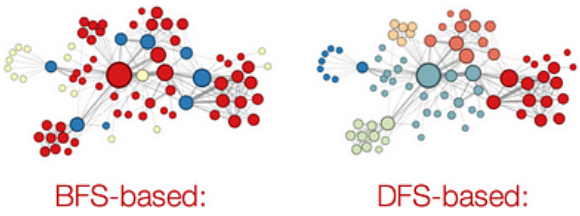


Figure 20: embeddings:BFS VS DFS

nodes (i.e., neighbors), while **second-order proximity** captures shared neighbors, highlighting the relationship between nodes that may not be directly connected. LINE employs a two-phase approach: it first optimizes the embedding of nodes based on first-order relationships and then focuses on second-order similarities. By utilizing a loss function that maximizes these proximities, LINE generates low-dimensional embeddings that preserve the structural properties of the graph. Its efficiency makes it particularly suitable for large networks, such as social media or citation networks, where scalability is crucial.

4.Graph Convolutional Networks (GCNs): Graph Convolutional Networks (GCNs) extend traditional convolutional neural networks to graph-structured data, enabling the aggregation of features from a node's local neighborhood. GCNs operate on the principle of **locality**, where each node's representation is updated by combining its features with those of its neighbors. This process is repeated across multiple layers, allowing GCNs to learn increasingly abstract representations. The key operation involves normalizing the adjacency matrix to incorporate node connectivity, which ensures that the embeddings reflect both the node's features and its structural context within the graph. GCNs are particularly effective for tasks such as node classification and link prediction, making them a popular choice in various applications, including social networks, protein-protein interaction networks, and knowledge graphs.

5. SDNE: SDNE (Structural Deep Network Embedding) is a powerful technique designed to capture both local and global structural information in graphs through deep learning. It combines the strengths of deep learning with graph theory by employing an autoencoder architecture. The approach utilizes two main components: **first-order** and **second-order** proximity. First-order proximity captures the direct connections between nodes, while second-order proximity captures the similarity between nodes based on their shared neighbors.

The SDNE model begins by initializing node embeddings randomly, then iteratively refines them through a two-step training process. The **autoencoder** learns to reconstruct the adjacency matrix, aiming to minimize the reconstruction loss. A crucial aspect of SDNE is its incorporation of a regularization term that preserves the network's local structures. This helps maintain the graph's inherent characteristics while embedding it into a lower-dimensional space.

The **loss function** is a combination of the reconstruction loss and the regularization

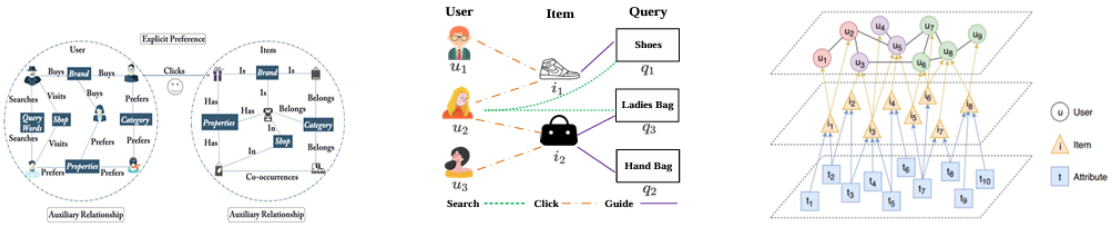


Figure 21: Representation of HGs in E-Commerce [47]

term, which guides the optimization of embeddings. SDNE is particularly useful for tasks like node classification and link prediction, providing high-quality embeddings that capture the complex relationships within the graph. The embeddings can then be visualized using techniques like t-SNE, allowing for insights into the underlying structure of the graph.

3.2 Heterogeneous Graph Representation Learning

A heterogeneous graphs, also known as a heterogeneous information network (Sun and Han 2012), is represented as $G = (V, E, A, R, \varphi, \phi)$, where:

- V is the set of nodes,
- E is the set of edges,
- $\varphi : V \longrightarrow A$ is a node type mapping function,
- $\phi : E \longrightarrow R$ is an edge type mapping function,
- A is the set of node types,
- R is the set of edge types,

such that $|A| + |R| > 2$. We also assume an input feature matrix for the nodes, $X \in \mathbb{R}^{|V| \times d}$.

Heterogeneous nodes consist of multiple types of nodes and edges like in RNA-KG nodes representing genes, proteins, diseases etc 21. with edges representing different types of relations among them allowing more understanding of relationships within the data unlike homogeneous graphs which oversimplifies them treating all entities as same.

Heterogeneous graph [48] is closely related to the real world applications, while many practical problems remain unsolved. For example, constructing an appropriate HG may

require sufficient domain knowledge in a real world application. Also, meta-path and/or meta graph are widely used to capture the structure of HG. However, unlike homogeneous graph, where the structure (e.g., the first-order and second-order structure) is well defined, meta-path selection may also need prior knowledge. Furthermore, to better facilitate the real world applications, we usually need to elaborately encode the side information (e.g., node attributes) or more advanced domain knowledge to the heterogeneous graph embedding process.

Can we use same embedding methods for the Heterogenous Graphs?

- The primary limitation of homogeneous methods is their assumption that all nodes and edges in the graph are of the same type. This assumption does not hold in heterogeneous graphs, where the diversity of node and edge types carries significant semantic importance.
- Loss of Rich Information: When methods like GCNs, DeepWalk, or Node2Vec are applied to heterogeneous graphs without modification, they tend to blend the information from different types of nodes and edges. This blending can lead to a loss of the rich, type-specific information that is crucial for tasks like node classification, link prediction, and community detection in heterogeneous graphs.
- Heterogeneous graphs often involve complex interactions between different types of entities. Homogeneous methods, which treat all interactions uniformly, fail to capture the complexity of these multi-type interactions.
- When these homogeneous methods are adapted to heterogeneous graphs, they often require a separate set of parameters or models for each type of node and edge. This greatly increases the complexity and computational cost of the models, making them less scalable and more prone to overfitting.
- To adapt methods like DeepWalk or Node2Vec to heterogeneous graphs, one might need to define meta-paths or other heuristic rules, which requires significant domain expertise and may not generalize well across different applications.
- Homogeneous graph methods are generally not designed to handle rich node features that differ significantly across types. In heterogeneous graphs, nodes of different types often have different types of attributes (e.g., text for papers, vectors for authors), and methods that fail to consider these differences may not utilize node features effectively.

3.2.1 Method Taxonomy

To consider the information of different aspects in the embedding, including the graph structures, attributes and specific labels, etc. Considering the mentioned challenges the existing methods are categorized on four categories based on the information they used in heterogeneous graph embedding: (1) Structure-preserved heterogeneous graph embedding The methods belonging to this category primarily focus on capturing and preserving the heterogeneous structures and semantics, e.g., the meta-path and meta-graph. (2) *Attribute-assisted heterogeneous graph embedding*. The methods incorporate more information beyond structure, e.g., node and edge attributes, into embedding technology, so as to utilize the neighborhood information more effectively. (3) *Application-oriented heterogeneous graph embedding*. We further explore the applicability of the heterogeneous graph embedding methods (i.e., the ones aim to learn application-oriented node embeddings over HG). (4) *Dynamic heterogeneous graph embedding*. Different from existing survey works that mainly focus on the embedding methods for static heterogeneous graphs. in table 22 we categorize typical HG embedding methods, where the the first two columns indicate whether the method has inductive capability and whether it needs labels for training. and we can deduce that most of the message passing-bassed methods have inductive capability as they update node embedding by aggregating neighbourhood information, with th need of labels as their guide for training process. the middle two column defines the information and task, for ehich these methods are used and the last two columns summarize the techniques used in Heterogeneous embedding along with their characterstics. It is worth noting that we also list the time complexity of each type of techniques, where t is the number of random walks, l is the length of random walk, k is the windows size in skip-gram [49] and ns is the number of samples.

The Graphical learning methods for Homogeneous Graphs frequently used

3.3 Trans-E

TransE [48] (Translation-based Embedding) is a fundamental model in the field of knowledge graph embedding. Developed to address the problem of knowledge graph representation, TransE transforms entities and relationships into continuous vector spaces, allowing for the learning of meaningful embeddings that capture the semantics of the graph data.

TransE represents relationships as translations in the embedding space. For a given triplet (head, relationship, tail), TransE models the relationship as a vector that translates the embedding of the head entity to the embedding of the tail entity. Formally, if **h** represents the embedding of the head entity, **r** represents the embedding of the relationship, and **t** represents the embedding of the tail entity, the model aims to minimize the following objective:

Method	Inductive	Label	Information	Task	Technique	Characteristic
mp2vec [8]			Strcuture	Embedding	Random walk (Shallow model)	• Easy to parallelize • Two-stage training • High memory cost Complexity: $\mathcal{O}(\tau \cdot l \cdot k \cdot n_s \cdot d \cdot \lvert\mathcal{V}\rvert)$
Spacey [59]						
JUST [60]						
BHIN2vec [61]						
HHNE [62]						
mg2vec [41]						
HeRec [2]		√	Strcuture+Task	Recommendation		
PME [17]			Strcuture	Embedding	Decomposition (Shallow model)	• Easy to parallelize • Two-stage training • High memory cost Complexity: $\mathcal{O}(\lvert\mathcal{E}\rvert \cdot d)$
EOE [50]						
HEER [53]						
MNE [57]						
PTE [56]						
RHINE [63]						
HAN [15]	√	√	Structure+Attribute		Message passing (Deep model)	• End-to-End training • Encoding structures and attributes • Semantic fusion • High training cost Complexity: $\mathcal{O}(\lvert\mathcal{V}\rvert \cdot d_1 + \lvert\mathcal{R}\rvert \cdot d_2)$
MAGNN [78]	√	√				
HetSANN [79]	√	√				
HGT [80]	√	√				
HetGNN [16]	√					
GATNE [73]	√					
GTN [83]		√				
RSHN [123]		√				
RGCN [124]	√	√				
IntentGC [20]	√	√	Strcuture +Attribute+Task	Recommendation		
MEIRec [19]	√	√				
GNUD [5]	√	√				
Player2vec [99]	√	√		Classification		
AHIN2vec [100]	√	√				
Vendor2vec [101]	√	√				
HIN2vec [9]			Strcuture	Embedding	Encoder-decoder (Deep model)	• End-to-End training • Flexible goal-orientation Complexity: $\mathcal{O}(\lvert\mathcal{V}\rvert \cdot d_1 + \lvert\mathcal{E}\rvert \cdot d_2)$
DHNE [66]						
HNE [70]	√	√	Structure+Attribute			
SHNE [71]		√				
NSHE [82]						
PAHNE [44]		√	Strcuture +Attribute+Task	Classification		
Camel [97]		√				
TaPEm [98]		√				
HeGAN [18]			Strcuture	Embedding	Adversarial (Deep model)	• Robustness • High complexity Complexity: $\mathcal{O}(\lvert\mathcal{V}\rvert \cdot \lvert\mathcal{R}\rvert \cdot n_s \cdot d)$
MV-ACM [125]						
Rad-HGC [24]		√	Strcuture+Task	Malware detection		

Figure 22: Typical Heterogenous Graph Embedding methods

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 23: The pattern modeling and inference abilities of several models.

$$\mathbf{h}+\mathbf{r}\approx\mathbf{t}$$

where ≈ indicates that the embeddings should be close in the vector space. This translation mechanism captures the essence of the relationship between entities and facilitates the prediction of missing links in the graph. This model provide a unified framework to handle various types of relationships by embedding them as translation vectors. The model´s simplicity allows it to scale effectively to large and complex heterogeneous graphs having millions of entities and relations.

The task of predicting missing links using knowledge graph embedding (KGE) methods has received considerable attention in recent years. The standard approach involves defining a score function for evaluating triplets. Formally, let E represent the set of entities and R the set of relations. A knowledge graph is then composed of factual triplets (h, r, t), where $h, t \in E$ and $r \in R$. Since entity embeddings are typically represented as vectors, the score function is generally denoted as $f_r(h, t)$, where h and t are the embeddings of the head and tail entities. The score function $f_r(h, t)$ assesses the validity of a candidate triplet (h, r, t). The objective during optimization is to assign higher scores to true triplets (h, r, t) compared to corrupted or false triplets, such as (h', r, t) or (h, r, t'). Figure 23 provides a summary of different score functions $f_r(h, t)$. [50] TransE represents each relation as a bijection between source entities and target entities, and thus implicitly models inversion and composition of relations

3.4 RotatE

[50] It represents the entities as complex vectors and relations as rotation in complex vector size. The RotatE model is a knowledge graph embedding technique specifically designed for link prediction tasks. It represents entities as complex vectors in a complex space and models relations as rotations within that space.

In RotatE, entities are represented as complex-valued vectors, where each relation is modeled as a rotation in the complex space. For a given triplet (head, relation, tail), the model applies a relation-specific rotation to the embedding of the head entity to obtain the embedding of the tail entity. Mathematically, if the head entity h is represented by a complex vector $\mathbf{h}$, and the relation r is represented by a rotation vector $\mathbf{r}$, the tail entity t is represented by $\mathbf{h} \circ \mathbf{r}$, where $\circ$ denotes element-wise multiplication, corresponding to rotation in the complex plane.

The score function in RotatE is defined as the distance between the rotated head embedding and the tail embedding:

$$d_r(h, t) = \|\mathbf{h} \circ \mathbf{r} - \mathbf{t}\|_1,$$

where $\|\cdot\|_1$ is the L_1 norm. The model aims to minimize this distance for true triplets and maximize it for false ones, allowing for accurate link prediction.

RotatE's use of complex space allows it to model a variety of relation patterns, including symmetry, antisymmetry, inversion, and composition. This expressiveness makes it suitable for handling complex relationships in knowledge graphs.

Due to its rotational mechanism, RotatE excels at link prediction by inferring how entities relate through rotations. This makes it particularly effective for knowledge graph completion tasks and has demonstrated superior performance in handling complex relationships. In the context of RNA-KG, applying RotatE could capture intricate relationships between RNAs, genes, proteins, and diseases, making it an ideal candidate for link prediction tasks in biomedical knowledge graphs.

3.5 RDF2VEC

RDF2vec was inspired by the word2vec approach [51] for representing words in a numeric vector space. RDF2vec transforms complex knowledge graphs into numeric embeddings, enabling powerful applications in data mining, recommendation systems, and entity classification.

In the context of representing RDF (Resource Description Framework) graphs, which consist of triplets made up of subjects, predicates, and objects, we can leverage neural language models to create embeddings for entities and their relationships. The process begins by transforming the RDF graph into sequences of entities and relations, akin to sentences in natural language. This transformation can be achieved through techniques like graph walks where we traverse the graph, randomly selecting neighboring nodes to record encountered entities and relations or using Weisfeiler-Lehman subtree RDF graph kernels to encode the structural relationships among nodes. For example, a triplet such as "Paris capitalOf France" can be represented in a sequence, along with other relations, to capture the context and connections within the graph. These sequences serve as input

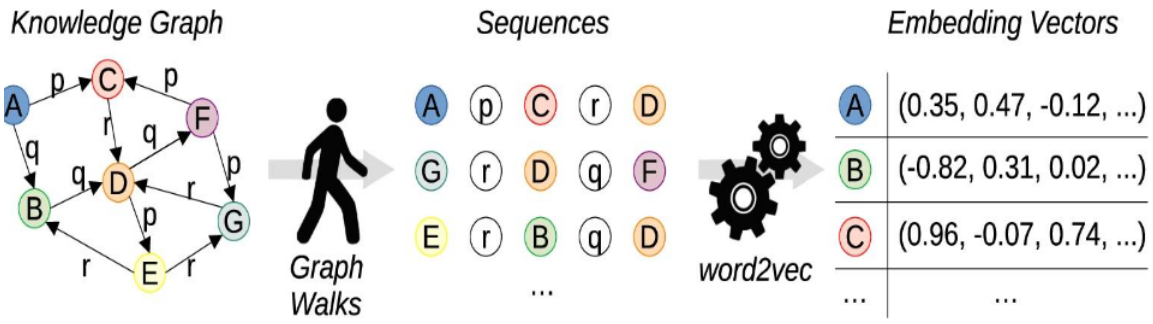


Figure 24: Nodes to embedding vectors

to neural language models like Word2Vec, which trains on the data to generate vector representations of the entities and relations. The training process ensures that similar entities and relations appear close to each other in the embedded space based on their co-occurrence in the sequences. Consequently, the resulting embeddings can be utilized for various applications, including link prediction, entity classification, and graph completion, enabling more meaningful insights and interactions within the RDF data.

3.6 GAT

Graph Attention Networks (GATs) are a type of neural network architecture designed to operate on graph-structured data. They leverage the attention mechanism to improve the way nodes in a graph aggregate information from their neighbors. GATConv, or Graph Attention Convolution, is a layer in Graph Attention Networks (GATs) that leverages the attention mechanism to compute node representations in graph-structured data. It allows nodes to weigh their neighbors' features differently, enabling the model to focus on the most relevant nodes in a neighborhood when generating embeddings. GATConv employs an attention mechanism that assigns different importance to neighboring nodes based on their features. This is particularly useful in heterogeneous graphs, where nodes and edges can represent different types of entities and relationships. By focusing on the most relevant neighbors, GATConv can generate more informative embeddings for each node. In heterogeneous graphs, nodes and edges can have various types and attributes. GATConv can effectively integrate this diverse information by learning to attend to different types of neighbors differently, which is crucial for tasks like link prediction where the relationship between different types of nodes needs to be understood. GATConv supports inductive learning, meaning it can generalize to unseen nodes during training. This is beneficial for link prediction, as it allows the model to predict links for new nodes that were not part of the training set. GATConv generates embeddings that capture the structural and

feature-based information of nodes in the graph. These embeddings can be used for various downstream tasks, including link prediction, where the goal is to predict the existence of edges between nodes. GAT tackles a major challenge in GNNs: effectively aggregating information from neighboring nodes while varying their importance. Unlike traditional GNNs, like Graph Convolutional Networks (GCNs), which use fixed aggregation methods treating all neighbors equally, GAT employs attention mechanisms. This allows each node to dynamically assign weights to the information from its neighbors, enhancing the aggregation process.

The GAT layer utilizes a shared linear transformation matrix W of dimensions (F', F), ensuring that each node's features are transformed uniformly to a common dimensionality F'. This transformation applies to all nodes in the neighborhood of node i, including node i itself.

Feature Transformation: The embedding representation h_i of the target node i is combined with the embeddings of its neighbors, transformed by a second matrix W^a with dimensions $(2F', F')$:

$$h_{ij} = \text{Concat}(h_i, h_j) \quad \text{for } j \in \mathcal{N}(i)$$

Attention Coefficients Attention scalars are generated through a non-linear activation function σ, where LeakyReLU is commonly used:

$$e_{ij} = \sigma(a^T W^a h_{ij})$$

These intermediate attention scalars are normalized using a softmax function to produce attention coefficients:

$$\alpha_{ij} = \text{softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}(i)} \exp(e_{ik})}$$

MULTI-HEAD Attention In a multi-head attention scenario with $K = 3$ heads, attention computations are performed independently for each head. The results are combined via concatenation or averaging, yielding the final representation for node 1:

$$h'_1 = \bigoplus_{k=1}^{K} \text{Attention}_k(h_1, \mathcal{N}(1))$$

This framework enables effective aggregation of neighboring information with varying importance, enhancing the GAT's capability in graph-based tasks.

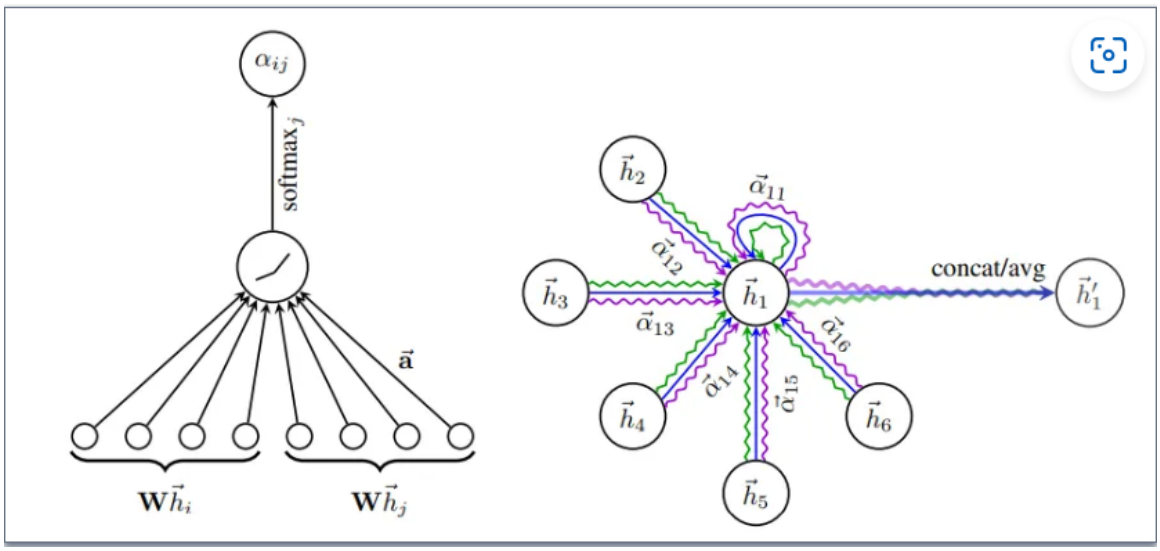


Figure 25: GAT Architecture 1

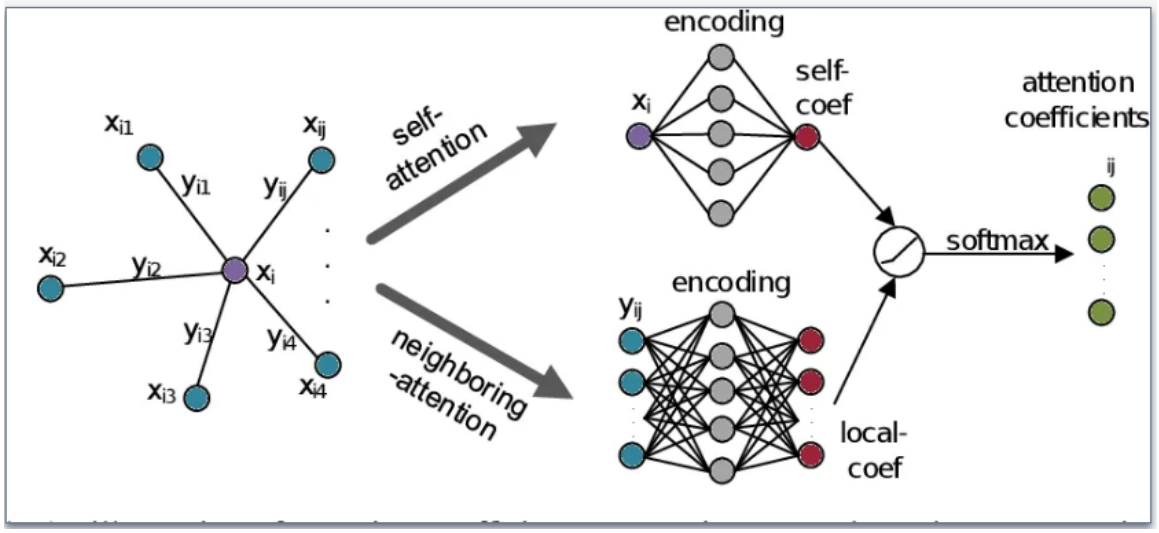


Figure 26: GAT Architecture 2

Chapter 4

Experimental Comparison of Link Prediction Methods on RNA-KG

In this chapter, we will delve into the core methodologies and findings of our research, focusing specifically on the prediction of edges and the node embeddings using different models. Edge prediction plays a pivotal role in many domains such as recommender systems, network theory, machine learning, etc., where anticipating relationships or connections between entities is crucial for enhancing model performance and providing actionable insights.

This chapter will begin by discussing the experimental setup of the models used for node embeddings followed by the prediction of nodes performed. We will then explore the edge prediction done using the node embeddings obtained from the non neural network and also edge prediction using a neural network GATCONV, with a particular emphasis on the results obtained from each and how it differs from one another providing a deeper analysis. Experiments were conducted on Google Colab using a T4 GPU.

4.1 Experimental setup

4.1.1 Embedding Model and Parameters

Among the different embedding approaches that we have discussed in the previous chapter, we here consider RotatE, TransE, and RDF2Vec. Each model was evaluated for its performance on both node type and edge type prediction tasks. After generating embeddings for the nodes, the **Random Forest Classifier** was used to classify node and edge embeddings. The embeddings generated by each model were scaled using StandardScaler, and the data was split into training and testing sets with an 80/20% ratio. Random Forest Classifier and desicion tree were trained to predict the node types, and its performance was evaluated using accuracy score, classification report, and confusion matrix.

4.1.2 Experimental Parameters

All experiments share parameters: one of them is the use of random seed initialization to guarantee replicable results (seed = 42). For the TransE and RotatE models, data splitting was performed using the RandomLinkSplit method, creating training, validation, and test datasets. RandomLinkSplit is a method used to divide the edges of a graph into three subsets: training, validation, and test sets. To evaluate the model effectively, negative samples (non-existent links) are also included in the validation and test sets. This approach ensures a fair and unbiased assessment of the model's link prediction performance while maintaining the integrity of the graph structure. This method ensures that the edges are randomly split while preserving the structure of the graph.

RDF2Vec was implemented with a 50/50 split for the train/test data.

One-hot encoding was employed for node types.

The dimensionality of the vectors representing node embeddings was set to 50 across the considered models.

Both TransE and RotatE employed the Adam optimizer [52] with a learning rate of 0.01 and 0.001 respectively. This consistency in optimization techniques enables a fair evaluation of model performance.

The focus will be testing classification models and evaluating their performance on predicting node types and new links uncovered in the current release RNA-KG.

4.1.3 Node type prediction

In this section, we explore the task of predicting node types within RNA-KG using different embedding models. Node classification was performed using embeddings from the TransE, RotatE, and RDF2Vec models. Due to the imbalanced distribution of node types in the complete graph, where some classes contain very few nodes, we also evaluated the classification performance on a subgraph with selected node types: miRNA, Gene, and Disease.

Table 5 shows the node classification for the graph of the different VIEWS of RNA-KG, Random Forest was used as the classifier to predict the node types based on the embeddings generated by each model. The accuracy of node type prediction for the graph varied significantly across different views and models, with the best overall results achieved using TransE embeddings on View 3.

Due to the imbalanced nature of the complete graph, a subgraph was developed for each view that contains only the three most significant node types: miRNA, Gene, and Disease (shown in Table 6). This selection aimed to improve classification performance by focusing on nodes with sufficient representation.

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
TransE										
View 0	0.53	1.00	0.39	0.91	0.92	0.51	0.91	0.85	0.75	0.91
View 2	0.49	0.97	0.37	0.85	0.78	0.50	0.87	0.66	0.74	0.89
View 3	0.55	0.89	0.42	0.94	0.82	0.53	0.92	0.75	0.72	0.91
rdf2vec										
View 0	0.26	0.20	0.28	0.24	0.00	0.39	0.22	0.00	0.62	0.20
View 2	0.48	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
View 3	0.26	0.06	0.28	0.22	0.00	0.38	0.22	0.00	0.60	0.21
rotate										
View 0	0.52	0.86	0.46	0.74	0.41	0.56	0.74	0.27	0.56	0.78
View 2	0.49	0.72	0.45	0.66	0.12	0.56	0.70	0.06	0.73	0.75
View 3	0.48	0.78	0.41	0.70	0.17	0.53	0.75	0.10	0.53	0.80

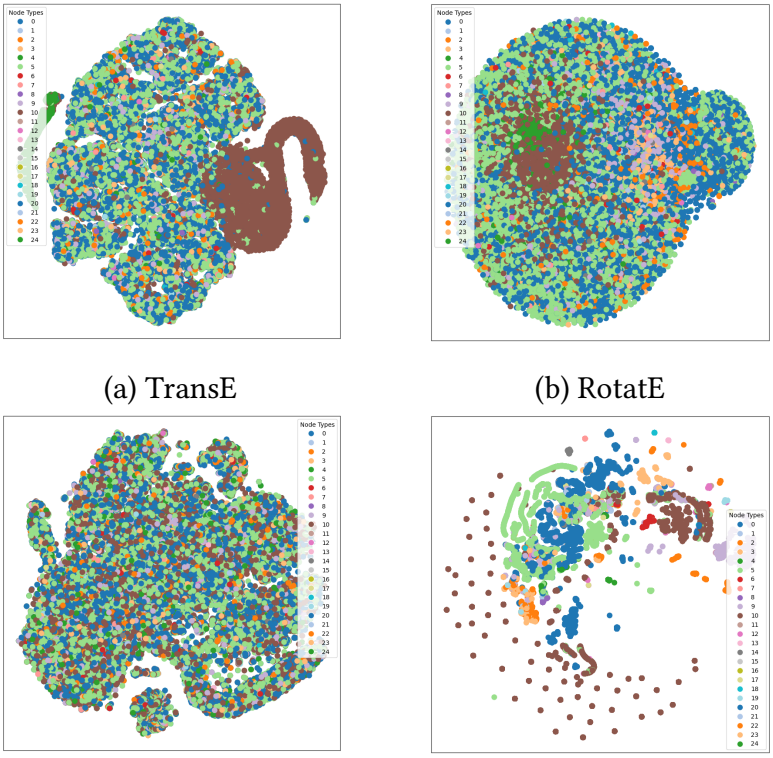
Table 6: Node type prediction evaluation of subgraphs of different VIEWS

Models/Views	Accuracy	Precision			Recall		
		miRNA	Disease	Gene	miRNA	Disease	Gene
TransE							
View 0	0.93	0.83	0.91	0.94	0.83	0.96	0.91
View 2	0.89	0.98	089	0.88	0.67	0.94	0.89
View 3	0.92	0.75	0.89	0.96	0.75	0.97	0.90
rdf2vec							
View 0	0.49	0.33	0.52	0.42	0.00	0.77	0.24
View 2	0.48	0.00	0.50	0.40	0.00	0.80	0.20
View 3	0.47	0.00	0.50	0.39	0.00	0.76	0.23
rotate							
View 0	0.82	0.87	0.84	0.79	0.26	0.92	0.80
View 2	0.75	0.79	0.78	0.71	0.06	0.90	0.74
View 3	0.79	0.76	0.81	0.76	0.10	0.91	0.80

4.1.4 Comparison of the Models Trained on RNA-KG views

In this section, we present a comparative analysis of the three models—TransE, RotatE, and RDF2Vec—based on their performance and the quality of node embeddings visualized using t-SNE. The comparison aims to illustrate how each model captures the relationships between entities within the RNA-centered knowledge graph (RNA-KG) and the implications for link prediction and node classification tasks.

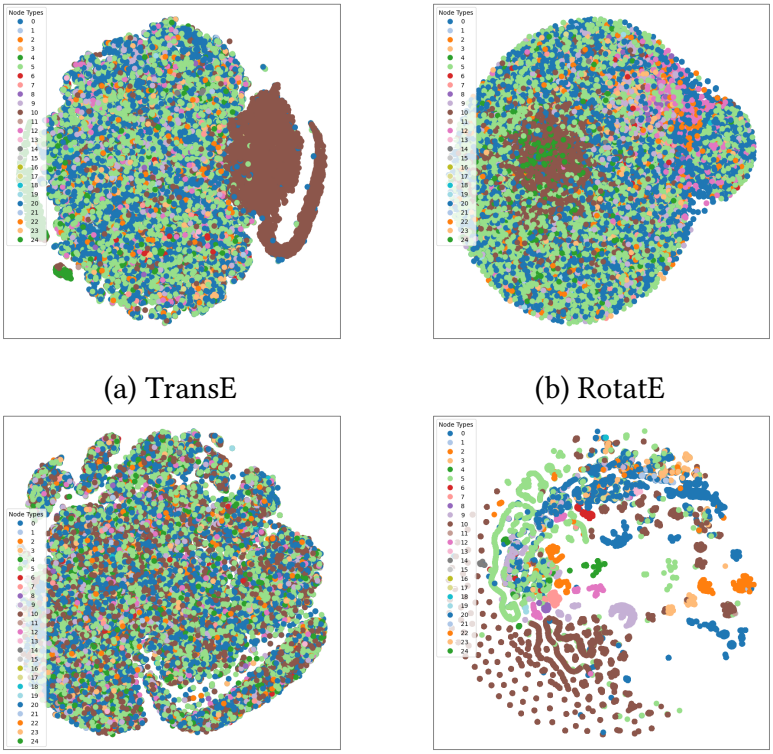
To facilitate the comparison, we utilized t-SNE to reduce the dimensionality of the generated embeddings from each model to two dimensions. This enabled a visual representation of the embeddings, making it easier to analyze the clustering and distribution of different node types.



(a) TransE (b) RotatE

(c) RDF2Vec (d) GATConv

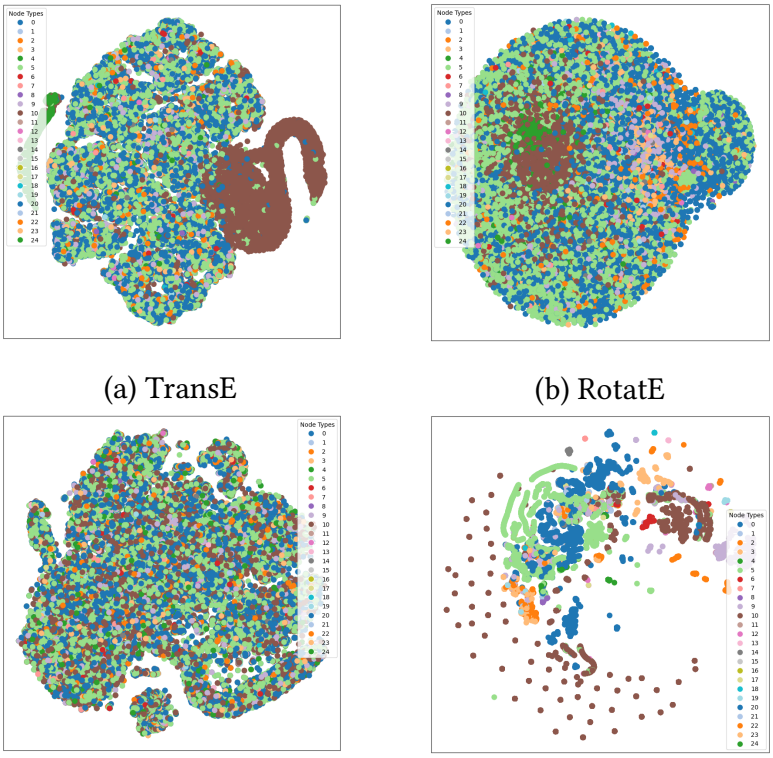
Figure 27: Comparison of Different Embeddings on View 0



(a) TransE (b) RotatE

(c) RDF2Vec (d) GATConv

Figure 28: Comparison of Different Embeddings on View 2



(a) TransE (b) RotatE

(c) RDF2Vec (d) GATConv

Figure 29: Comparison of Different Embeddings on View 3

4.2 Generic Edge prediction

4.2.1 Embedding Model and Parameters

The embedding models used for the link prediction task include TransE, RotatE, RDF2Vec, and GATConv. The embeddings generated by these models serve as input for the edge prediction task. All experiments are conducted using 5 holdouts for cross-validation, an 80/20 train/test split, and a fixed random seed for reproducibility.

4.2.2 Edge Prediction Model and Parameters

Random Forest model and Desicion Tree model were employed to predicted plausible links not covered by RNA-KG. The key parameters of this model are presented in Tables 7. All experiments use 5 holdouts with an 80/20 train/test split, and a fixed random seed (seed = 42) is set for reproducibility. The edge prediction evaluation is performed using the GRAPE library. The pipeline leverages the precomputed embeddings generated by

the aforementioned models, with edge prediction being evaluated using the parameters outlined.

Table 7: 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

Table 8: 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

The edge prediction pipeline for both Decision Tree and Random Forest begins with generating node embeddings, which encode graph structure and semantics into vector representations. Embeddings of the two nodes for each relationships were concatenated. Both methods use an 80/20 train-test split to ensure fair evaluation. The Decision Tree classifier

uses a maximum depth of 10, while the Random Forest model uses multiple decision trees with ‘sqrt‘ as the maximum features, bootstrap sampling enabled, and parallel processing for efficiency.

In terms of performance, Decision Tree models showed more straightforward interpretability but were prone to overfitting, with limited generalization when compared to Random Forest. On the other hand, Random Forest improved robustness by averaging multiple trees, reducing variance, and achieving slightly higher accuracy and balanced metrics like informedness and likelihood ratios. However, Decision Trees had a simpler architecture, making them faster to train and easier to interpret compared to the computational overhead of Random Forest. Overall, Random Forest demonstrated better generalization and predictive reliability at the cost of interpretability and resource consumption.

Table 9: Random Forest Train Metrics for View 0

Model	**Precision**	**Recall**	**F1 Score**
RDF2VEC	0.50	0.05	0.10
RotatE	0.55	0.64	0.60
TransE	0.50	0.98	0.66
GATConv	0.50	0.10	0.18

Table 10: Random Forest Test Metrics for View 0

Model	**Precision**	**Recall**	**F1 Score**
RDF2VEC	0.50	0.05	0.10
RotatE	0.55	0.63	0.58
TransE	0.50	0.99	0.66
GATConv	0.50	0.06	0.11

Table 11: Random Forest Train Metrics for View 2

Model	**Precision**	**Recall**	**F1 Score**
RDF2VEC	0.50	0.006	0.01
RotatE	0.56	0.63	0.60
TransE	0.50	0.97	0.66
GATConv	0.74	0.16	0.26

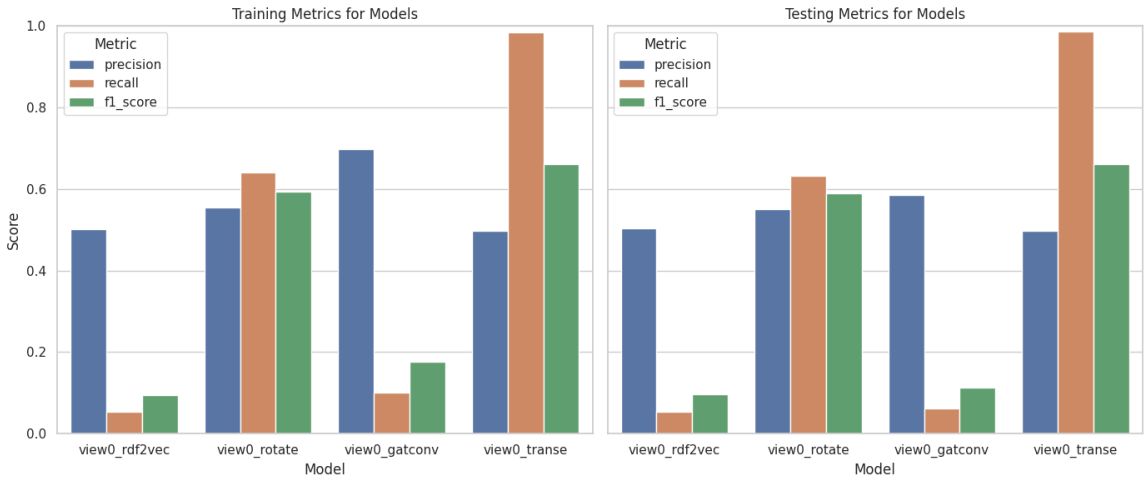


Figure 30: Evaluation of edge prediction on View0 across different models

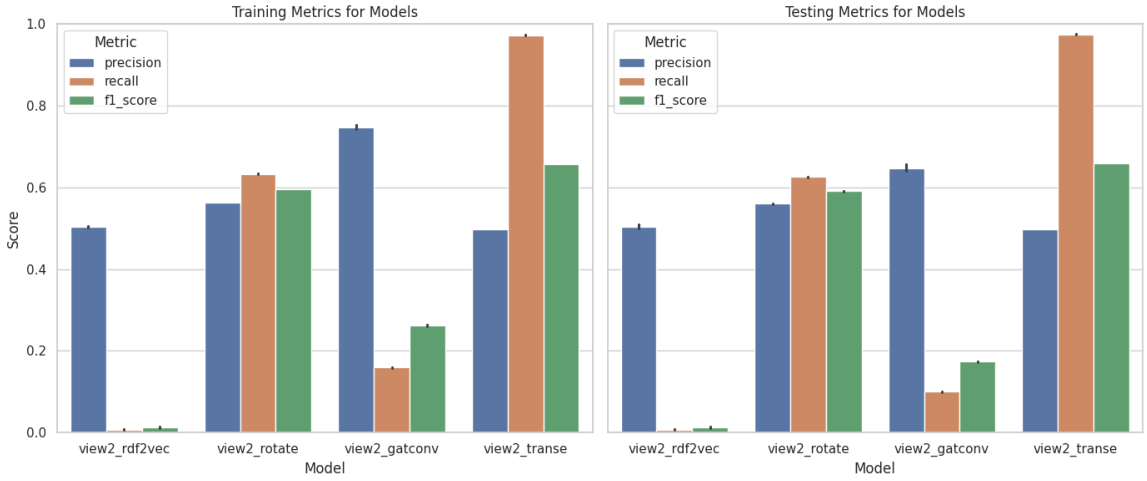


Figure 31: Evaluation of edge prediction on View2 across different models

Table 12: Random Forest Test Metrics for View 2

Model	Precision	Recall	F1 Score
RDF2VEC	0.50	0.006	0.01
RotatE	0.56	0.62	0.59
TransE	0.50	0.97	0.66
GATConv	0.65	0.10	0.17

The edge prediction results for Random Forest and Decision Tree classifiers across different RNA-KG views highlighted several key findings.Random Forest consistently outperformed Decision Trees, particularly in handling complex and multi-relational edge. In View 0, Random Forest achieved balanced precision and recall, especially with TransE and RotatE, where TransE reached a precision of 0.50 and recall of 0.99, resulting in F1 score of 0.66. In View 2, Random Forest with GATConv yielded a precision of 0.65, though its recall remained low at 0.10, indicating a challenge in capturing underrepresented edge types. This view emphasized Random Forest's strength in generalization compared to Decision Trees, which showed lower precision and recall values, especially with models like GATConv.

Table 13: Random Forest Train Metrics for View 3

Model	Precision	Recall	F1 Score
RDF2VEC	0.51	0.15	0.23
RotatE	0.52	0.32	0.33
TransE	0.50	0.97	0.66
GATConv	0.79	0.10	0.18

Table 14: Random forest Test Metrics for View 3

Model	Precision	Recall	F1 Score
RDF2VEC	0.51	0.15	0.21
RotatE	0.49	0.31	0.32
TransE	0.50	0.98	0.66
GATConv	0.70	0.10	0.12

In View 3, TransE continued to perform strongly, with high precision (0.50) and recall (0.98), indicating its robustness in capturing consistent edge patterns across large-scale biological data. However, GATConv, despite high precision (0.70), faced issues with low

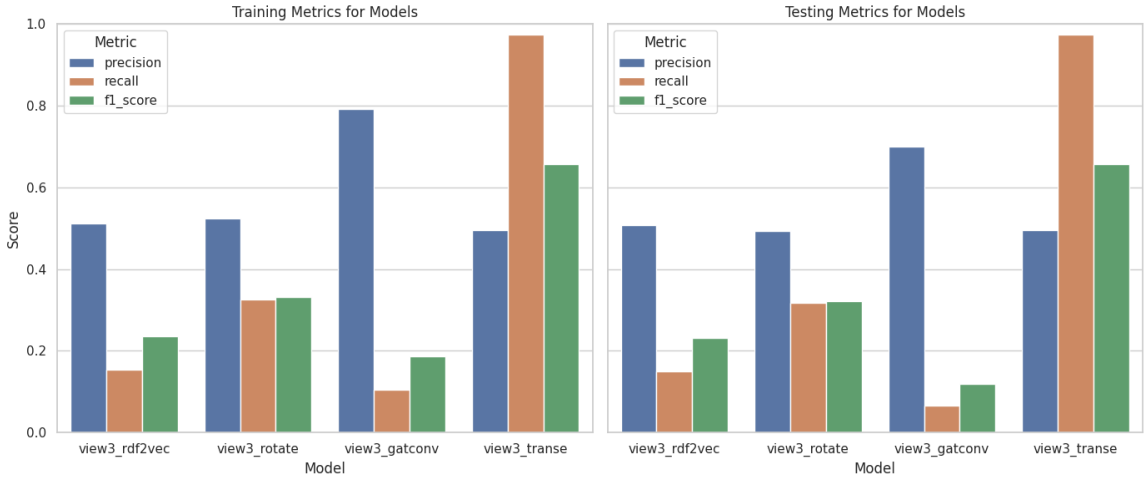


Figure 32: Evaluation of edge prediction on View 3 across different models

recall (0.10), highlighting the model's inability to predict certain relationships effectively, particularly with sparse or imbalanced edge types. Decision Trees demonstrated faster training times but struggled with overfitting, leading to poorer performance in generalization across the views. Although they had good recall for some edge types, such as TransE in View 0 (recall 0.95), the overall metrics were less consistent compared to Random Forest.

Table 15: Decision Tree Train Metrics for View 0

Model	**Precision**	**Recall**	**F1 Score**
RDF2VEC	0.50	0.21	0.29
RotatE	0.53	0.60	0.56
TransE	0.50	0.95	0.65
GATConv	0.65	0.11	0.18

Overall, Random Forest proved to be a more reliable classifier, offering robust performance across varying edge types and views, particularly with embeddings from TransE and RotatE. However, both classifiers faced challenges in dealing with sparse or underrepresented edge types, which impacted their performance, especially in View 2 and View 3. GATConv, while effective in capturing attention-based relationships, struggled with low recall, suggesting the need for improvements in its ability to generalize across heterogeneous biological networks. These results emphasize the importance of model selection based on data characteristics, such as the sparsity of relationships, and point to the need

Table 16: Desicion Tree Test Metrics for View 0

Model	Precision	Recall	F1 Score
RDF2VEC	0.50	0.21	0.29
RotatE	0.53	0.60	0.56
TransE	0.50	0.95	0.65
GATConv	0.54	0.06	0.193

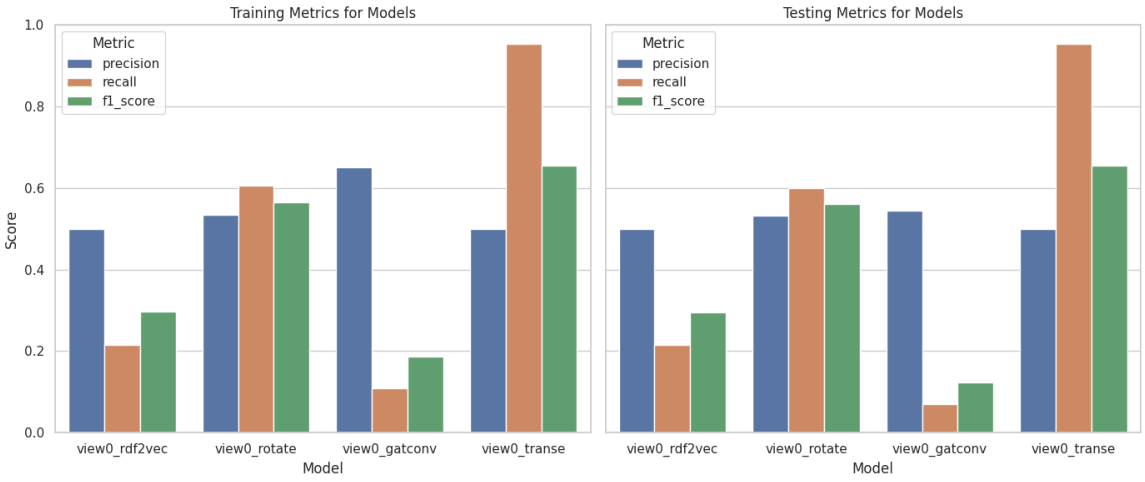


Figure 33: Evaluation of decision tree on View 0 across different models

for further refinement of edge prediction methods, especially in handling complex, multi-relational biological data.

4.2.4 GATCONV Method

Unlike traditional methods such as transE, rotatE which rely on mathematical transformations in embedding spaces, GATConv learns the importance of neighbouring nodes and edges for each node and can prioritize relations like "causes or contributes to condition" and connections shown by achieving high precision. GATConv can clearly represent structure of the graph and show distinct and well separated clusters in visualization tools like t-SNE, enabling clearer and more meaningful visualizations compared to others.

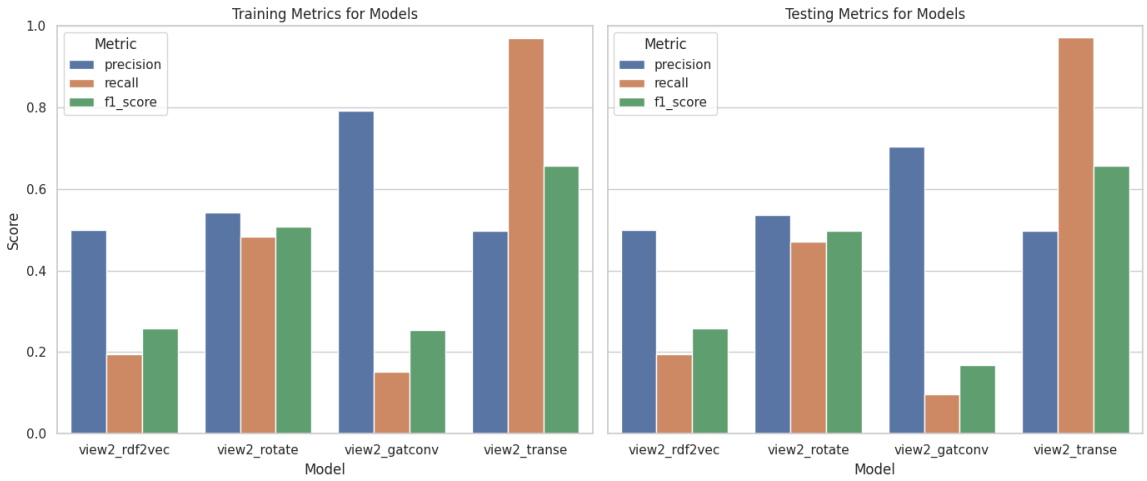


Figure 34: Evaluation of decision tree on View 2 across different models

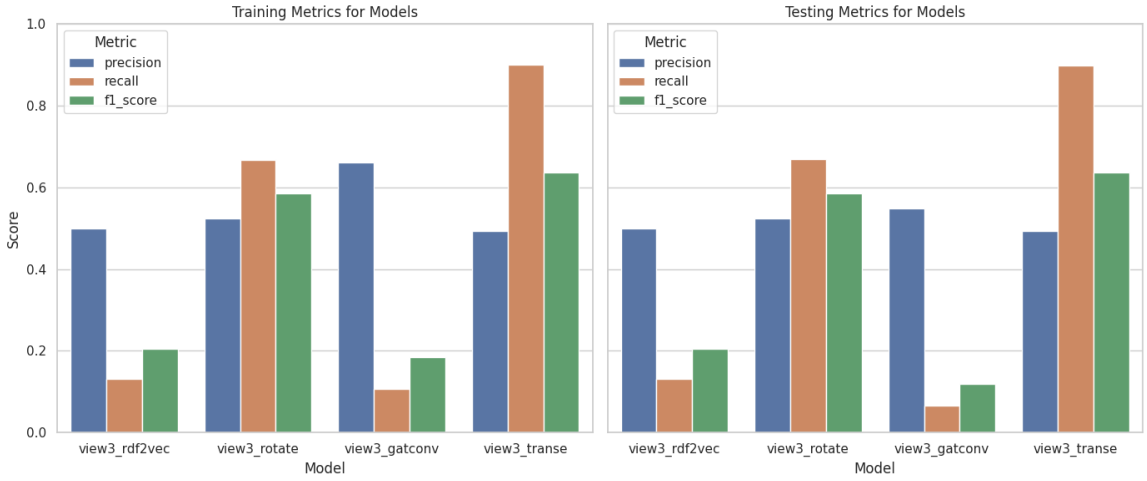


Figure 35: Evaluation of decision tree on View 3 across different models

Table 17: Decision Tree Train Metrics for View 2

Model	Precision	Recall	F1 Score
RDF2VEC	0.50	0.19	0.25
RotatE	0.54	0.48	0.51
TransE	0.50	0.97	0.66
GATConv	0.79	0.15	0.25

Table 18: Decision Tree Test Metrics for View 2

Model	Precision	Recall	F1 Score
RDF2VEC	0.50	0.19	0.25
RotatE	0.53	0.47	0.50
TransE	0.50	0.95	0.65
GATConv	0.70	0.09	0.16

Table 19: Decision Tree Train Metrics for View 3

Model	Precision	Recall	F1 Score
RDF2VEC	0.50	0.13	0.20
RotatE	0.52	0.67	0.58
TransE	0.49	0.90	0.63
GATConv	0.66	0.10	0.18

Table 20: Decision Tree Test Metrics for View 3

Model	Precision	Recall	F1 Score
RDF2VEC	0.50	0.13	0.20
RotatE	0.52	0.67	0.58
TransE	0.49	0.90	0.63
GATConv	0.55	0.66	0.12

Chapter 5

Conclusions and Future Work

This study demonstrates the potential of using heterogeneous link prediction methods to analyze RNA-centered Knowledge Graphs (RNA-KG). RNA-KG is a pioneering framework designed to uncover complex relationships among RNA molecules, genes, proteins, diseases, and other key biomedical entities. By drawing data from over 60 public repositories, RNA-KG has proven to be a valuable tool for gaining new biological insights and driving progress in drug discovery and understanding disease mechanisms.

To analyze RNA-KG, embedding models such as TransE, RotatE, and RDF2Vec were employed to generate low-dimensional representations of the entities and relationships within the graph. These embeddings served as a critical foundation for subsequent edge prediction tasks, offering a compact and informative representation of the graph's structure. GATConv, on the other hand, was used directly for edge prediction, leveraging its attention-based mechanism to effectively model the complex, multi-relational nature of RNA-KG. Performance metrics such as accuracy,precision, and recall highlighted the strengths of these approaches, offering a strong foundation for their use in solving real-world biomedical challenges.

In addition to these techniques, machine learning classifiers like Random Forest and Decision Tree were integrated into the edge prediction workflow. These models were employed to enhance the edge prediction process by utilizing the embeddings generated from models such as TransE and RDF2Vec as input features. Random Forest, with its ensemble learning capability, provided robust predictions by combining multiple decision trees and reducing the risk of overfitting. Decision Tree models, while simpler, offered clear interpretability, allowing for an in-depth understanding of how specific features contributed to the prediction of edges. Together, these models added an extra layer of predictive power and interpretability to the study, complementing the capabilities of neural and non-neural embedding techniques.

The results of this study provide valuable insights into the performance of these diverse methods when applied to RNA-KG. Among the evaluated models, GATConv stood

out as the most effective for edge prediction, capturing intricate relationships and producing highly accurate results. At the same time, TransE emerged as the best-performing non-neural embedding model, combining simplicity and computational efficiency for large-scale datasets. The integration of Random Forest and Decision Tree further demonstrated the utility of combining graph embeddings with machine learning models, enhancing predictive accuracy and providing actionable insights.

These findings highlight the complementary strengths of embedding models, neural edge prediction methods, and machine learning classifiers, showing how each can be tailored to different needs. Moving forward, further improvements could focus on optimizing GATConv for scalability, refining embedding models like TransE to capture more complex relationships, and exploring hybrid approaches that combine the interpretability of decision trees with the depth of neural methods. These advancements will further unlock the potential of RNA-KG as a tool for biomedical research and discovery.

RNA-KG, like many biological networks, evolves with new research and discoveries. Incorporating dynamic graph learning methods could help models predict not only current relationships but also future trends. This is especially valuable in fields like drug discovery, where understanding how interactions evolve under different conditions (e.g., drug treatments) can reveal new therapeutic targets. Temporal graph networks could also track disease progression and the evolution of regulatory relationships over time.

One exciting future possibility is developing an interactive Gene-Disease Prediction App. This tool would use RNA-KG and edge prediction models to visualize gene-disease relationships in real time. By integrating data such as genetic sequences and clinical information, it could enhance predictions and uncover new associations. Using models like GATConv, the app could offer insights into drug repurposing and allow researchers and clinicians to explore gene-disease interactions interactively. Such a platform could support personalized medicine and make complex biological networks more accessible and actionable.

As these models evolve, they have the potential to transform biomedical research by helping scientists uncover new relationships, predict disease outcomes, and identify novel treatments. Continued advancements in edge prediction methods will further improve their accuracy and scalability, deepening our understanding of complex biological systems and driving the development of more effective therapies.

Bibliography

[1] R. Johnson, M. M. Li, A. Noori, O. Queen, and M. Zitnik. Graph ai in medicine, 2023.

[2] Alice J Barbier, Andrew Y Jiang, Ping Zhang, Richard Wooster, and Daniel G Anderson. The clinical progress of mrna vaccines and immunotherapies. *Nature Biotechnology*, 40(6):840–854, 2022.

[3] Sebastian Hombach and Markus Kretz. *Non-coding RNAs: Classification, Biology and Functioning*, pages 3–17. Springer International Publishing, 2016.

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

[5] Neo4j. Neo4j - the world's leading graph database. Available at `http://neo4j.org/`, 2012.

[6] Dave Beckett and Brian McBride. RDF/XML Syntax Specification (Revised) - W3C recommendation. Available at `https://www.w3.org/TR/REC-rdf-syntax/`, February 2004.

[7] Daniele Alocci, Maarten van Vliet, Julien Mariethoz, Frédérique Lisacek, and Miguel A Rojas-Macias. Property graph vs rdf triple store: A comparison on glycan substructure search. *PLOS ONE*, 10(12):e0144578, 2015.

[8] OWL Working Group. Web ontology language (owl) - w3c recommendation. Available at `https://www.w3.org/OWL/`, December 2012.

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

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

[11] Anke Sparmann and Jörg Vogel. Rna-based medicine: From molecular mechanisms to therapy. *The EMBO Journal*, 42(16):e114760, 2023.

[12] Luca Statello, Chongjian J. Guo, Ling-Ling Chen, et al. Gene regulation by long non-coding rnas and its biological functions. *Nature Reviews Molecular Cell Biology*, 22:96–118, 2021.

[13] Qian Zhang, Nicholas J. McKenzie, Rebecca Warneford-Thomson, et al. Rna exploits an exposed regulatory site to inhibit the enzymatic activity of prc2. *Nature Structural & Molecular Biology*, 26:237–247, 2019.

[14] Christine Roden and Amy S. Gladfelter. Rna contributions to the form and function of biomolecular condensates. *Nature Reviews Molecular Cell Biology*, 22:183–195, 2021.

[15] Xuefei Jia, Xiaofan He, Chao Huang, et al. Protein translation: biological processes and therapeutic strategies for human diseases. *Signal Transduction and Targeted Therapy*, 9:44, 2024.

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

[17] Chenguang Wang, Lianzong Wang, Yu Ding, Xiaoyan Lu, Guosi Zhang, Jiaxin Yang, Hewei Zheng, Hong Wang, Yongshuai Jiang, and Liangde Xu. Lncrna structural characteristics in epigenetic regulation. *International Journal of Molecular Sciences*, 18(12):2659, 2017.

[18] Frederike Lam, Matthias S Leisegang, and Ralf P Brandes. Lncrnas are key regulators of transcription factor-mediated endothelial stress responses. *International Journal of Molecular Sciences*, 25(17):9726, 2024.

[19] Kathrin Plath, Susanna Mlynarczyk-Evans, Dmitri A Nusinow, and Barbara Panning. Xist rna and the mechanism of x chromosome inactivation. *Annual review of genetics*, 36(1):233–278, 2002.

[20] Agnieszka Anna Rawłuszko-Wieczorek, Agnieszka Siera, and Paweł Piotr Jagodziński. Tet proteins in cancer: Current 'state of the art'. *Critical reviews in oncology/hematology*, 96(3):425–436, 2015.

[21] 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.

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

[23] 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.

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

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

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

[27] 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.

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

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

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

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

[32] AnacletoLAB. Rna-kg: A knowledge graph framework for rna biology. `https://github.com/AnacletoLAB/RNA-KG`, 2023. Accessed: 2024-11-21.

[33] Emanuele Cavalleri et al. A meta-graph for the construction of an rna-centered knowledge graph. In Ignacio Rojas, Oscar Valenzuela, Francisco Rojas Ruiz, Luis Javier Herrera, and Francisco Ortuño, editors, *Bioinformatics and Biomedical Engineering*, pages 165–180. Springer Nature Switzerland, Cham, 2023.

[34] Tiffany J. Callahan, William A. Baumgartner Jr., Michael Bada, Adrianne L. Stefanski, Ignacio Tripodi, Elizabeth K. White, and Lawrence E. Hunter. OWL-NETS: Transforming owl representations for improved network inference. *Computational Bioscience Program, University of Colorado Denver Anschutz Medical Campus*, 2024. E-mail: tiffany.callahan@ucdenver.edu.

[35] 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.

[36] S. L. Salzberg. Open questions: How many genes do we have? *BMC Biology*, 16, 2018.

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

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

[39] 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.

[40] 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.

[41] Mengjia Xu. Understanding graph embedding methods and their applications. *SIAM Review*, 63(4):825–853, 2021.

[42] 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.

[43] Zhonglin Ye, Haixing Zhao, Ke Zhang, Yanlin Yang, and Lei Meng. Deep walk algorithm based on improved random walk with equal probability. *International Journal of Performability Engineering*, 15(8):2237, 2019.

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

[45] Tosin P Adewumi, Foteini Liwicki, and Marcus Liwicki. Word2vec: Optimal hyper-parameters and their impact on nlp downstream tasks. *arXiv preprint arXiv:2003.11645*, 2020.

[46] Chris McCormick. Word2vec tutorial-the skip-gram model. *Apr-2016.[Online]. Available: http://mccormickml. com/2016/04/19/word2vec-tutorial-the-skip-gram-model*, 2016.

[47] 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*, 2021.

[48] 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.

[49] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[50] 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.

[51] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations (ICLR)*, 2013.

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

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