Otto-von-Guericke University Magdeburg

Faculty of Computer Science

Master Thesis

Extending the Capabilities of Clinical Knowledge Graph for the Integration and Analysis of Multi-Omics Data

Author:

Ammar Ateeq

May 8, 2024

Advisors:

Prof. Dr. rer. nat. habil. Gunter Saake
Department of Computer Science
Otto-von-Guericke University Magdeburg, Germany

Prof. Dr.-Ing. Robert Heyer
Faculty of Technology
Bielefeld University, Bielefeld, Germany

Ateeq, Ammar:
Extending the Capabilities of Clinical Knowledge Graph for the Integration and Analysis of Multi-Omics Data
Master Thesis, Otto-von-Guericke University Magdeburg, 2024.

Abstract

Biomedical research and healthcare systems face many obstacles due to the complexity and rapidly growing amount of omics and clinical data, e.g. efficient storage, querying and analyzing. As the volume of data grows, conventional database systems and methods are becoming ineffective in managing and interpreting the vast graphs of interconnected data. These problems can be effectively solved by graph databases because of their natural capacity to effectively store and retrieve highly connected data. One example of such graph is the Clinical Knowledge Graph how this graph can be utilised. On the other hand, there are challenges associated with Clinical Knowledge Graph's implementation are that it contains millions of nodes and edges making it more difficult to understand what information is stored in the knowledge graph and how those information are connected. Furthermore, these large graphs' structural and functional characteristics are sometimes not well understood, which hinders their practical deployment in clinical and research settings. In order to improve our comprehension of the Clinical Knowledge Graph's structure and the crucial relationships between its nodes and edges, this study provides a thorough examination of the information stored in the graph and its connections. Additionally, a web application created specifically to improve the accessibility of the Clinical Knowledge Graph is proposed in this thesis. Users can query the graph easily via to this application, which also provides interactive features that help them extract useful information from the dataset more quickly. By making the graph more helpful for researchers and medical practitioners, who do not have expertise in Neo4j, Python and other scripting and programming languages.

Acknowledgments

I am deeply thankful to Prof. Dr.-Ing. Robert Heyer for providing me the opportunity to collaborate on a research project, which sparked my interest and led to the selection of this thesis topic. Throughout the development of my thesis, I received the most insightful feedback and suggestions from him.

I received exceptional guidance from Kay Schallert, whose humility and dedication profoundly influenced my approach to research and concept development. During this relatively longer time of research, he found time and supported me with his meta-omics data and programming knowledge. I feel I have learned a lot from him.

I am impartially grateful to Daniel Walke for his unwavering support throughout my thesis. His readiness to explore and discuss various challenges has been invaluable. Every meeting and discussion came with packets of information which assisted me with productive working.

I also extend my gratitude to Prof. Dr. rer. nat. habil. Gunter Saake, whose innovative research inspired me to pursue my thesis within the research area and this department.

Lastly, I owe a heartfelt thanks to my friends and family, specially my mother, Uzma Ateeq, who happens to be a great teacher and has always encouraged me to leverage my strengths and fostered resilience in me.

Contents

List of Figures **ix**

List of Tables **xiii**

1 Introduction **1**
1.1 Motivation 1
1.2 Research Scope 2

2 Background **5**
2.1 Omics 5
2.1.1 Genes 5
2.1.2 Transcripts 6
2.1.3 Proteins 6
2.1.4 Types of Omics 7
2.1.5 Multi-Omics 8
2.1.6 Meta-Omics 8
2.1.6.1 Metaproteomics 9
2.1.6.2 Application of Metaproteomics 9
2.2 Graphs 10
2.2.1 Types of Graphs 10
2.2.2 Graph Databases 11
2.2.2.1 Labeled Property Graph (LPG) 13
2.2.2.2 Resource Description Framework (RDF) 13
2.2.2.3 LPG vs RDF 13
2.2.3 Relational Database vs Graph Database 14
2.2.4 Types of Graph Databases 16
2.2.4.1 Representation based on Storage 16
2.2.4.2 Representation based on Data Model 17
2.2.5 Neo4j Database 17
2.2.6 Knowledge Graphs 18
2.2.6.1 Types of Knowledge Graphs 18
2.2.7 Clinical Knowledge Graphs 19
2.2.7.1 Pillars of Clinical Knowledge Graphs 19
2.2.7.2 Applications of Clinical Knowledge Graphs 20
2.2.8 Graph Analysis 21
2.2.8.1 Random Walks 21
2.2.8.2 Embedding 22
2.2.8.3 Node2Vec 22

2.2.8.4 MetaPath2Vec 23
2.3 Web Application 24
2.3.1 Frontend 25
2.3.2 Backend 25
2.3.3 API 26
2.3.4 REST 26
2.3.5 Types of Web Applications 26
2.3.6 Angular 27

3 Related Work **29**
3.1 The heterogeneous pharmacological medical biochemical network PharMeBINet 29
3.2 Prototyping an Interactive Visualization of Dietary Supplement Knowledge Graph 30

4 Methodology **33**
4.1 Workflow 33
4.1.1 Frontend 34
4.1.2 Backend 34
4.1.3 Graph Analysis 35
4.1.4 Communication Flow 35
4.2 Data Abstraction and Processing 35
4.2.1 Graph Data 36
4.2.2 Abstraction Level 1 38
4.2.2.1 Node Type Count 38
4.2.2.2 Edge Type Count 38
4.2.2.3 Data Distribution 39
4.2.3 Abstraction Level 2 39
4.2.4 Abstraction Level 3 40
4.3 User Interface Development 40
4.3.1 Frontend 41
4.3.2 Backend 41
4.4 Hardware and Software Requirements 41

5 Evaluation **45**
5.1 Findings of Abstraction Level 1 45
5.2 Findings of Abstraction Level 2 49
5.3 Findings of Abstraction Level 3 55
5.4 Web Application 59
5.5 Discussion of Research Questions 61

6 Conclusion **63**
6.1 Scope of Future Work 63

Bibliography **65**

List of Figures

2.1 Diagram explaining the central dogma of molecular biology. (Reverse transcription is very error prone and only seen in some virus species.) [Bernaola et al., 2023] 9

2.2 Evolution of database models. Rectangles denote database models, arrows indicate influences, and circles denote theoretical developments. A time-line in years is shown on the left (based on a diagram by A. O. Mendelzon). [Angles and Gutierrez, 2008] 12

2.3 Graph Representation of 3 Connected Nodes, "Person" is the "author of" Book 1 and Book 2 18

2.4 Types of Embedding and it's comparison represented by "2D PCA projections of the 128D embeddings of 16 top CS conferences and corresponding high-profile authors" [Dong et al., 2017] 24

2.5 Abstract of a Frontend connects to a Backend using API, with responsible stakeholders [Bondel et al., 2021] 26

4.1 This is an image of Architecture of the Workflow and implementation components, where we see the communication patterns within and outside of frontend and backend. It also illustrated how Neo4j database is connected to multiple functionalities. 34

4.2 Levels of Abstraction of data based on Aggregation for Analysis. Each level defines a level deeper and granular data, it is done for better understanding and usage of the data. The levels of abstraction increases from left to right. 36

5.1 Count of 10 most abundant nodes in the Clinical Knowledge Graph. Where "Known Variant" being dominates the table by 10630108 nodes, while the least abundant in this table is "Gene", which accounts to 42571 nodes of the total. 46

5.2 Count of 10 least abundant nodes in the Clinical Knowledge Graph. Where "User" being the most deficient in the table by just 2 nodes, while the most abundant in this table is "Complex", which accounts to 2700 nodes of the total. 46

5.3 Cumulative Distribution Function plot based on all Nodes. The flat regions at the lower end of the x-axis indicate that there are large ranges of values where the cumulative probability does not increase, suggesting that there are no or very few observations in these ranges. The median (50th percentile) would be found at the point where the CDF is 0.5. This graph appears to reach 0.5 at a relatively high value on the x-axis, which would be consistent with a right-skewed distribution. 47

5.4 The image explains the variation of 10 most occurring edges, having ”Mentioned in Publication” as the highest count (472511840), while ”Detected in Pathology Sample” (3394496) being the 10th value in the chunk. 48

5.5 The image explains the variation of 10 lowest occurring edges, having ”Participates In”, ”Is Responsible”, ”Studies Tissue” and ”Studies Disease” having count equal to 14 48

5.6 Cumulative Distribution Function plot based on all Edges, the x-axis represents the relationship counts, scaled logarithmically. This scale is used to manage a wide range of values, allowing for a clearer visualization of data when relationship counts vary over several orders of magnitude. The slow initial growth of the CDF at lower relationship counts implies that many entities in the CKG are sparsely connected. 49

5.7 Clustered Heat-map based on Euclidean distance for all Node Type pair where we see that most of the connected nodes have 0 values, this is because most of the nodes are not connected to each other. Moreover, we have high connection for Drug, Disease and Modified Protein to Publication. 50

5.8 The image represents the highest occurring nodes type pairs, i.e., ”Publication-Tissue” (205739716) and ”Disease-Publication” (111058056). 51

5.9 The image represents the lowest occurring nodes type pairs, i.e.,”Project-Tissue” and ”Disease-Project” (14) each. 52

5.10 Cumulative Distribution Function Plot for all Node types pair, it shows how the CDF is distributed amongst the node type pairs.The graph covers a wide range of values on the x-axis, indicating significant variation in the dataset. The flat, almost horizontal section at the lower part of the x-axis suggests that a substantial proportion of the dataset has very low counts. This means that many entities or observations have minimal occurrences. The steep ascent later in the graph indicates that there is a rapid increase in cumulative frequency as the counts grow. The sharp vertical rise near the end of the x-axis indicates that the maximum count or highest value of the dataset is reached abruptly. 52

5.11 Node Degree Distribution for the Node ”Subject”, it show clearly that it contains low or no information that could contribute to our research. This the reason why we are focusing only interesting node types. . . . 53

5.12 Node Degree Distribution for the interesting nodes i.e., Gene, Drug, Transcript, Protein, Disease and special mention Metabolite. All of the graphs show high distribution of node degrees, with Disease having an outlier in the first bar, while Transcript shows an odd distribution of node degree 54

5.13 Protein In and Out Degree Distribution (1), here we see the node out degree in red and node in green. This figure shows ”Protein” - ”Associated With” - ”Drug, Cellular Component and Biological Process” and ”Protein” - ”Annotated In” - ”Pathway”. 55

5.14 Protein In and Out Degree Distribution (2), here we see the node out degree in red and node in green. This figure shows ”Protein” - ”Associated With” - ”Molecular Function and Tissue” abd ”Protein” - ”Curated Interacts With and Compiled Interacts with” - ”Protein”. . . 56

5.15 Protein In and Out Degree Distribution (3), here we see the node out degree in red and node in green. This figure shows ”Protein” connection to respective node types via certain edge types. 57

5.16 Protein In and Out Degree Distribution (5), here we see the node out degree in red and node in green. This figure shows ”Protein” connection to respective node types via certain edge types. 58

5.17 Clinical Knowledge Graph - Query Panel of Web Application, the user have the space to type in a Cypher query and retrieve the results. This is the primary functionality of the Web Application. 59

5.18 Clinical Knowledge Graph - Drop down functionality, in this view users can use the drop down option to select nodes types and edge types. They result in tabular manner are then displayed for selected query. We can also see the Cypher query generated when a respective node type and edge type is selected. 60

5.19 Clinical Knowledge Graph - Accession List Plugin, is the figure, we see an option of inserting a custom accession file, which can be used to filter specific proteins. This is aimed for researchers with dedicated uses cases like drug re-purposing. 61

List of Tables

2.1 Comparison between Relational and Graph Databases [Lazarska and Siedlecka-Lamch, 2019] 15

2.2 Graph Databases based on Storage Type 16

2.3 Graph Databases based on Data Model 17

4.1 Focused Nodes and with respective Relationship Types 37

4.2 Environment Versions and Usage of Neo4j Tools in the Thesis 42

4.3 Environment Versions and Usage of Angular for Development in the thesis 42

4.4 Environment Versions and Usage of Python for Development in the thesis 43

4.5 Environment Versions and Usage of Java for Development in the thesis 44

1. Introduction

1.1 Motivation

Researchers are increasingly engaging in the fields of genomics [Bustamante et al., 2011], transcriptomics [Lowe et al., 2017], and proteomics [Kellner, 2000] in their search for new biomarkers and innovative treatments for diseases. Through this method, they examine the connections between differentially expressed genes 2.1.1, transcripts 2.1.2, and proteins 2.1.3 and specific health conditions, as well as the effects of therapeutic interventions such as drug treatments. By linking these biological molecules to diseases and their responses to various treatments, scientists have the potential to make substantial advancements in medical diagnostics and treatment strategies. The integrated omics (discussed in Section 2.1) approach is transforming our comprehension of complex biological systems and their impact on disease progression and treatment efficacy.

High throughput techniques are methods and technologies that allow enormous amounts of data to be processed quickly and effectively in a short amount of time. These methods have greatly increased the speed, scale, and precision of data capture, analysis, and interpretation, revolutionising a number of scientific domains including omics. Now that this massive amount of data is available, we must address how to store and analyze it. Additionally, the data should be organised so that it can meet a number of performance requirements i.e. fast readability and reliable storage. Scalability is becoming a more crucial factor to take into account as omics datasets get bigger and more complicated. Challenges may arise from traditional relational databases' incapacity to scale horizontally in response to growing processing demands and data volumes. Differences in metadata, standards and data formats are the root cause of the challenges in integrating omics data from several platforms and sources. Metadata and different data format standards could be different naming conventions, different schema and data formats i.e Json and XML [Robinson et al., 2015]. Achieving seamless data integration requires data accessibility standards, and dependable pipelines for data in

Graph databases (discussed in Section 2.2.2) stand out as an efficient approach for handling data characterized by complex relationships and graph structures. They are particularly suited for deploying knowledge graphs, which are complex networks of connected data points utilized to integrate and analyze large sets of related information. The Clinical Knowledge Graph 2.2.7 serves as an example of how graph databases are applied. It consolidates diverse biomedical data ranging from genomics and drug data to clinical outcomes into a unified, navigable format. This allows researchers and healthcare providers to tap into a rich reservoir of interconnected information, enhancing both research capabilities and clinical decision making.

Though they have many benefits, clinical knowledge graph is very large and can be difficult to understand and navigate due to their extensive connections and content. It can be challenging to see how different pieces are connected or to anticipate the outcomes of these relationships due to the scale and complexity of these graphs, which can also hide insights and prevent the extraction of important information. Due to this, in order to efficiently investigate and make use of the enormous volumes of data included inside huge knowledge graphs, specific analytical tools and procedures are required.

To navigate and fully exploit the potential of Clinical Knowledge Graph, specialized methods are essential. Visual analytics tools and interactive querying interfaces can help researchers and clinicians explore actionable insights from our graph data.

However, accessing these massive knowledge graphs often presents a practical challenge due to the demanding hardware requirements and the specialized expertise needed to set up and maintain such systems. These barriers can limit the usability of knowledge graphs to a smaller group of technically skilled users. To democratize access and make these powerful tools more universally usable, it is crucial to develop more user friendly software that minimizes these demands.

Solution such as web application, which is available without any prior installations and downloads can lower these barriers. This can be an easy and intelligent solution without requiring users to invest in expensive infrastructure. Additionally, the development of intuitive interfaces that make use of the underlying complexity of graph databases would make it easier for non-specialists to navigate and extract information from clinical knowledge graph. These advancements would widen the accessibility of knowledge graphs, allowing a broader range of users to contribute to and benefit from the rich insights they offer.

1.2 Research Scope

In this study, I have contributed to the following research questions about analysis of underlying structure of the clinical knowledge graph and made it more accessible:

Q1. How can we visualize different graph properties in terms of Nodes, Edges and their structure?

Exploratory data analysis of the Clinical Knowledge Graph is extensively done to explore and understand how the data is structured and the extremities across all nodes and edges 5.1. Different types of visualisations and aggregations further helped us understand the data. A detailed EDA and data profiling of the Clinical Knowledge Graph. The analysis include:

- Evaluate spread of data abundance across all nodes and edges to better understand structure, quality and distribution.
- Different plots to visualize the contribution of each node and it's magnitude. Furthermore, exploring the topology, patterns and connections of the graph.

Q2. What can we learn from different abstraction levels of the Clinical Knowledge Graph?

As the knowledge graph is huge and consists of a lot of nodes and edges, we highlight the most important and interesting nodes, which helps us trace the knowledge based on the graph. In depth work is done to highlight the presence, abundance, connections and meaning of the particular nodes. We further focus on the following points:

- Abstraction level 2 (5.2) encompasses the analysis of node degrees based on nodes and their relationships, with the assumption of graph to be undirected.
- Abstraction level 3 (5.3) encompasses the analysis of node degrees based on node - relationship - node, with the assumption of graph to be directed.

Q3. What are user-friendly methods for querying data from a Knowledge Graph via a front-end interface?

Develop a detailed pipeline and a user interface for analysts, students and researchers for querying data from the Clinical Knowledge Graph. There is no requirement of downloading and installing any graph database management system, which makes it easier for non-technical audiences. Since it could be time consuming and complicated, setting up the entire graph database is not required as it comes connected with the web interface, The User Interface contains:

- A pipeline which allows the user to select node types (e.g. Disease or Proteins) and edge types (e.g. any relationship type) 4.3. Selecting edge type is completely optional as the user might only be interested in a certain node. Finally, users retrieve the required set of information.
- A search panel to enter queries and retrieve relevant data in a easy to use environment. For this, our user needs to have some understanding of Cypher query language (2.2.5), which can be from basic to advanced queries.

2. Background

2.1 Omics

Our primary motivation is to obtain a thorough grasp of the fundamental mechanisms that underlie the causes and therapy of various diseases at the start of our investigation. Gaining this fundamental understanding is essential to creating therapeutic strategies that work. In this project, omics analysis proves to be a highly important tool, providing a variety of methods to unravelling the complex molecular landscapes linked to various diseases. Through the integration of data derived from diverse omics disciplines such as proteomics, metabolomics, transcriptomics, and genomes, researchers might reveal promising patterns, biomarkers, and therapeutic targets that could advance our comprehension and treatment of a range of medical disorders. We can decipher the complex interactions between genetic, molecular, and environmental elements that contribute to the pathogenesis of disease by using the lens of omics analysis. The motivation for Omics studies comes from the pursuit of a comprehensive understanding of biological systems at various molecular levels. These large scale, high-throughput studies provide valuable insights into the intricate complexities of living organisms and thereby, we avail its usage for solving problem space which contains high involvement of Disease, Proteins and Genes etc [Schneider and Orchard, 2011].

2.1.1 Genes

Genes are strands of DNA that serve as the blueprints for protein synthesis and are fundamental to hereditary transmission. The gene's instructions are subsequently carried by this mRNA molecule to the ribosome, where they serve as a guide for the creation of proteins. Transfer RNA (tRNA) facilitates translation, converting the nucleotide sequence of the mRNA into a sequence of amino acids. tRNA guarantees proper amino acid assembly by serving as a transporter and matching partner for mRNA codons. The mRNA sequence then directs the synthesis of peptide bonds, which join these amino acids to form a polypeptide chain. In order for the developing protein to function biologically, it must finally fold into its functional three-dimensional structure [Pruitt et al., 2007]. They carry genetic information passed from parents to their offspring and significantly influence an individual's traits, such as physical features, behaviors, and disease predisposition. Genes regulate a myriad of biological activities, encompassing growth, development, and metabolic processes. As vital components for the operation and persistence of all life forms, genes are a central subject of study in genetics, molecular biology, and biotechnology [Hutchins, 2014].

Following are a few types of Genes discussed:

- **Structural Genes**: Those that encode specific proteins or RNA molecules that are directly involved in an organism's structure [Maniatis et al., 1978].
- **Regulatory Genes**: Control the expression of other genes, influencing when and to what extent they are transcribed [Krützfeldt and Stoffel, 2006].
- **Housekeeping Genes**: Essential for basic cellular functions e.g. basic maintenance and cell survival, and are constitutively expressed in all cells [Eisenberg and Levanon, 2013].

2.1.2 Transcripts

Transcripts are kinds of RNA that are produced by gene transcription and serve as blueprints for protein synthesis. They serve as a link in gene expression, transmitting genetic instructions from DNA to mRNA.Their involvement is critical in controlling how genes are expressed and biological systems function. These transcripts are essential for converting genetic instructions into active proteins and are subjected to a variety of modifications and processing prior to protein synthesis. Understanding transcripts, as an essential field of study in molecular biology and genetics, gives light on the complex mechanisms of gene regulation and the pathway to protein synthesis. They are converted into mRNA molecules through transcription and subsequently into distinct amino acid sequences to create proteins. Alongside that, it carries the genetic information from DNA to the ribosomes, serving as a template for protein synthesis during translation. [Burnard, 1991]

2.1.3 Proteins

Proteins essentially are molecules as amino acid as their building blocks and are one of the basic constituents of every biological being, starting from microorganisms to fully grown animals and humans. In terms of functions, Proteins are responsible for holding a cell together, working as catalyst for multiple chemical reactions and protect cells from diseases. In cells, most of the work done is by proteins and they are also required to hold the structure together. Proteins can be visualised as a sequence of Amino acids, or rather, a large concatenation of them i.e. anything between 50 to 2000. With a different combination of amino acid, that could be 20 different types, and also several folding mechanisms like Spontaneous Folding and Co-translational Folding, a protein is formed [Grantcharova et al., 2001]. The function of each protein can be directly derived from its construction [Hepler and Gilman, 1992; Tanaka and Scheraga, 1976]. The connection between genes, transcripts, and proteins represent a tightly regulated and intricate process fundamental to the functioning of living organisms. Some examples of proteins and their functions:

- **Phenylalanine hydroxylase**: Help in the creation on new molecules via retaining knowledge from genetic DNA
- **Growth hormone**: Works as a Messenger. Produces signals and maintain communication between several cells and organs.
- **Actin**: Responsible for locomotion of a body and provides supports to the cells.

Figure 2.1 represents how genetic information is transferred within a cell. It's starts from DNA, then to RNA and lastly, Proteins. The arrows show the flow of information and conversion. The overall process is shown step by step in a generic sense.

2.1.4 Types of Omics

Omics is a collective term encompassing various high-throughput, large-scale, and comprehensive biological studies that aim to characterize and analyze the complete set of biological molecules within a biological system. The term Omics is used as a suffix in life science data vocabulary and is now being largely accepted in world of Bioinformaticians and molecular biologists. Omics disciplines include genomics, transcriptomics, proteomics, and metabolomics, each focusing on a specific class of molecules. These studies have become integral to advancing our understanding of the intricate molecular mechanisms governing life processes [Fiehn, 2002].

Omics research is rooted in the desire to obtain a dominant understanding of biological systems and unravel the complexities that govern them. These studies contribute to advancements in personalized medicine, disease diagnosis and treatment, agricultural biotechnology, and our fundamental knowledge of life processes. Omics approaches are facilitated high-throughput technologies, such as next-generation sequencing, mass spectrometry, and microarray technologies. These technologies generate massive datasets, necessitating sophisticated computational and bioinformatic analyses to extract meaningful information [Alberts, 2017].

The key motivations for omics research include:

- **Holistic View of Biological Systems**: Omics approaches, such as genomics, proteomics, and metabolomics, allow researchers to examine biological systems as integrated wholes. This holistic perspective is crucial for uncovering the interconnectedness of molecular components and understanding the emergent properties of living organisms [Gupta and Misra, 2016].
- **Identification of Biomarkers**: Omics studies contribute to the discovery of biomarkers, which are specific molecules or signatures indicative of physiological states, diseases, or responses to treatments. Biomarkers play a vital role in disease diagnosis, prognosis, and personalized medicine [Olivier et al., 2019].
- **Precision Medicine**: Precision medicine is a medical approach that leverages information regarding an individual's genetic makeup or protein characteristics to prevent, diagnose, or treat diseases. Within oncology, precision medicine utilizes detailed data concerning a patient's tumor to aid in diagnosis, treatment planning, assessment of treatment efficacy, and prognosis determination[Ashley, 2016].

- **Advancing Biological and Medical Research**: Omics technologies contribute to the advancement of basic biological research by generating vast datasets that can be analyzed to answer fundamental questions about cellular processes, signaling pathways, and regulatory mechanisms [Chenab et al., 2019].
- **Biotechnological Applications**: Omics data are crucial in various biotechnological applications, such as the development of genetically modified organisms, optimization of industrial processes, and the engineering of microorganisms for specific purposes [Horikoshi, 1999].

In the vast and investigative domain of biological sciences, the understanding of molecular biology fundamentals; genes, transcripts, proteins and metabolites forms the cornerstone of plenty disciplines, including genetics, biochemistry, and molecular pathology. The genetic material contained in a gene is subjected to transcription during the synthesis of proteins, producing messenger RNA (mRNA).

2.1.5 Multi-Omics

Following could fall under an umbrella of Omics studies [Anderson and Seilhamer, 1997; Wang et al., 2009]:

- **Genomics**: Genomics, as a discipline in molecular biology, concentrates on the in-depth examination of an organism's entire set of genes, referred to as the genome. This field encompasses the scrutiny of gene structure, function, organization, and interactions to achieve a comprehensive understanding of the genetic information governing the development, functioning, and evolutionary processes of living organisms [Bustamante et al., 2011].
- **Transcriptomics**: It is a field of biological study that involves the comprehensive analysis of the complete set of RNA transcripts produced by the genome of a cell at a specific point in time or under particular conditions. This field aims to understand the patterns and dynamics of gene expression, encompassing processes such as transcription, mRNA processing, and RNA stability [Lowe et al., 2017].
- **Proteomics**: Proteomics, undertakes a thorough examination of the entire ensemble of proteins generated by a cell, tissue, or organism. This field encompasses the scrutiny of protein characteristics, including structure, function and expression patterns [Kellner, 2000].
- **Metabolomics**: It involves studying the entirety of small molecules, metabolites, circulating through cells which are the biochemical functionality of cell's life ongoing processes [Roessner and Bowne, 2009].

2.1.6 Meta-Omics

The term "meta-omics" refers to the study of complete proteome of a community within a biological system at the same time e.g the microbial community in the human gut. It combines information from multiple omics fields e.g. genomes, transcriptomics and proteomics.By taking into account the interactions and linkages between different molecular components, meta-omics aims to present a more comprehensive and linked understanding of biological processes. It entails system-level studies that go beyond the scope of individual omics viewpoints. Meta-omics encapsulates the simultaneous holistic examination of various biological datasets (omics), within the same biological samples. This approach garners a thorough insight into the functional roles and taxonomic structures of microbial areas, along with their environmental interactions. Such research is valuable in the study of microbiomes, offering a window into the intricate and varied microbial ecosystems and their potential connections to human health and diseases [Mallick et al., 2021].

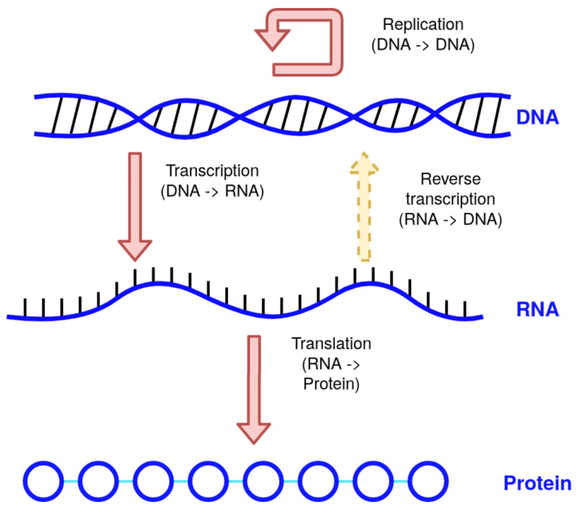


Figure 2.1: Diagram explaining the central dogma of molecular biology. (Reverse transcription is very error prone and only seen in some virus species.) [Bernaola et al., 2023]

2.1.6.1 Metaproteomics

Metaproteomics is dedicated to investigating the entirety of proteins within a complex combination of biological samples, such as those found in tissues, saliva, and bone marrow etc. Its primary objective is to discover the functional roles and interactions of expressed proteins within intricate communities, shedding light on the metabolic and ecological processes to better understand a specific ecosystem. Differentiating from conventional proteomics, it emphasizes on the comprehensive protein composition of a sample rather than concentrating on a single organism or a predefined set of proteins. The information derived from metaproteomics data contribute significantly to fields like microbiology, and biotechnology [Karczewski and Snyder, 2018].

2.1.6.2 Application of Metaproteomics

Metaproteomics involves a series of technical steps to analyze soil microbial communities by examining the proteins they express. The process begins with sample collection and protein extraction, where the most crucial part is ensuring the extracted proteins are representative of the sample, given the complexity of microbial communities and the presence of interfering substances like phenolic compounds and humic acids. The extraction strategy depends on the targeted protein fraction (prokaryotic/eukaryotic, extracellular/cell-associated) and the subsequent methods of protein analysis planned. Other applications of Metaprotemoics can be; Disease Marker Discovery, Process Monitoring, Environmental Studies and Routine Clinical Diagnostics etc. One of the prominent goals is to associate identified proteins with their corresponding gene sequences, linking biological functions to genetic data. This requires optimized technologies for characterizing the protein complement of the genome, which are still being developed and refined compared to nucleic acid-based methods [Heyer et al., 2019; Maron et al., 2007].

2.2 Graphs

Effective data representation and analysis are more important than ever in the big data and multi-omics research era. A substantial solution is provided by graphs, which give a clear and understandable structure for capturing the complicated relationships and interactions found in various biological systems. With the increasing amount and complexity of multi-omics datasets, graphs provide a flexible framework for integrating various omics data kinds.Graphs allow researchers to find hidden patterns, identify important molecular networks, and obtain a greater understanding of the underlying mechanisms driving biological processes by based on biological entities as nodes and their connections as edges. Graphs are an effective tool for exploring complexity and realising the potential of interdisciplinary data integration in the dynamic field of multi-omics research.Graphs are combination of nodes and edges, nodes contain data and value of a certain use-case, while the edges define how two or more are connected to each other. Overall, a huge network of nodes and edges are combined to form a detailed structure, which can then be used for specific analysis.

Graphs posses a pivotal role in problem formulation, serving as models for addressing an array of practical challenges, they prove particularly valuable in portraying and resolving problems tied to communication networks [Jiang, 2022], data structuring [Tarjan, 1983], computational devices [Zhang et al., 2013], and computational flow [Buning and Steger, 1985]. For instance, within the domain of computer networks, the link structure of a website can be articulated through a directed graph, where web pages constitute the vertices and directed edges symbolize links between pages. Moreover, graph-theoretic approaches have demonstrated efficacy in linguistics, where the discrete nature of natural language aligns well with graph structures. Alongside that, using graph databases, we have the access to the extensive graph learning algorithms. The use of graphs in problem-solving is advantageous because they provide enormous efficiency in terms of computation and also helps with a visual and intuitive representation of relationships and connections between entities. This makes it easier to analyze and solve complex problems, such as finding the shortest route in a network, optimizing resource allocation, or modeling and simulating real-world systems.

2.2.1 Types of Graphs

There are several types of graphs used in functional and implementation aspect of a project scope. Following list contains the summary of graphs types and their usage:

- **Regular graphs**: In graph theory, two vertices are said to have an identical degree if they are connected to the same number of other vertices in the graph. These graphs exhibit a uniformity in their structure, with every vertex maintaining an identical degree, signifying an equal in and out going edges to for each node [Wormald et al., 1999].
- **Integral graphs**: Integral graphs are a class of graphs where all the eigenvalues of their adjacency matrix are integers. The adjacency matrix of a graph is a square matrix that represents the connections between vertices [Balińska et al., 2002].
- **Directed graphs**:directed graphs—also referred to as digraphs—are graphs in which each edge is connected to a particular direction. In a digraph, directed edges indicate a one-way link between vertices [Commoner et al., 1971].
- **Undirected graphs**: An undirected graph is one in which each edge lacks a defined direction. Rather, they depict symmetric connections between vertex pairs [Kamada et al., 1989].
- **Weighted graphs**: A weighted graph is a type of graph in which each edge is assigned a numerical value, known as a ”weight.” Unlike an unweighted graph, where edges merely represent connections between vertices, the edges in a weighted graph carry additional information indicating the strength, cost, distance, or some other quantitative measure associated with the relationship between the connected vertices [Althöfer et al., 1993].

2.2.2 Graph Databases

We are seeing an exponential rise in data volume and quality due to technological improvements. This explosion calls for better ways to handle and store large amounts of data, which is driving the foundational advancement of database systems. Traditional relational database management systems (RDBMS) have given way to a wide range of database technologies that are tailored to the requirements of modern applications in the evolution of databases. The databases have to change after being mostly dominated by RDBMS, which are recognised for their structured data arranged in tables with set schemas. RDBMS is typically linked with strong transactional capabilities. This gap prompted the advent of NoSQL databases, which offer flexible data models and the capability for horizontal scalability, thus supporting various data types and large volumes. Key-value stores, document stores, wide-column stores, and graph databases have become prominent within the NoSQL category, each serving specific data handling requirements.

The emergence of NoSQL significantly challenged the conventional roles of relational databases, diversifying the database landscape. Concurrently, object-oriented database management systems (OODBMS) were developed to provide native support for complex data structures and object-oriented programming frameworks, effectively closing the gap between application code and database storage. This period also saw the rise of NewSQL systems, which blend NoSQL's scalability and flexibility with the robust transactional capabilities traditionally associated with RDBMS [Kamada et al., 1989].

The the following image we visualise Evolution of database models.

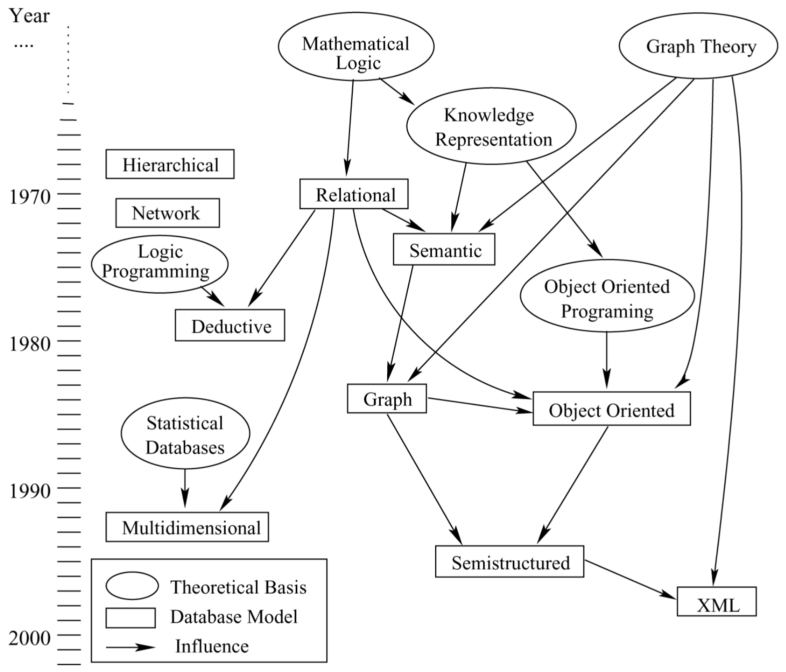


Figure 2.2: Evolution of database models. Rectangles denote database models, arrows indicate influences, and circles denote theoretical developments. A time-line in years is shown on the left (based on a diagram by A. O. Mendelzon). [Angles and Gutierrez, 2008]

Graph databases are specialized systems designed to manage and process graph-like data, following the fundamental principles of database systems such as persistent data storage, logical data independence, data integrity, and consistency. These databases are specifically tailored to handle large, evolving, and rich graph datasets, which may contain trillions of vertices and edges. Examples of such datasets include social networks or web graphs [Besta et al., 2023].

Database is a compilation of ordered or unordered data, which is very structured. It contains all elements of data, its tables, relationships and connections and meta-data. All of the management, controlling and monitoring is catered by Database Management System [McCarthy and Dayal, 1989]. Database queries are syntax based vocabulary, which follow a direct pattern, to extract or recall data from any database [Jarke and Koch, 1984]. Specifically engineered to tackle the distinctive challenges arising from the vastness and complexity of these datasets, graph databases also meet the requirements for low latency and high throughput associated with graph queries. These queries encompass both local actions, involving the access or modification of a small segment of the graph, and global actions, encompassing the access or modification of a substantial portion of the graph. The design of graph databases encompasses several key dimensions, including the general structure of the graph database structure, data models and organization, data distribution, and query execution.

Online transaction processing is designed to manage and process large amounts of transactional data quickly. It is made to effectively manage frequent, brief transactions, such INSERT, UPDATE, and DELETE operations, which are commonly connected to regular business operations [Harizopoulos et al., 2018]. Online analytical processing is designed to make advanced searches and analysis of large quantities of data easier. Since OLAP systems are built to handle multidimensional analysis, users can drill down, slice, and dice data to obtain insights and make well-informed choices [Chaudhuri and Dayal, 1997]. In addition to their core functionalities, graph databases are also compared and categorized based on their support for different types of queries and workloads, such as OLTP (Online Transaction Processing) and OLAP (Online Analytical Processing).

The landscape of graph databases includes various systems, each with its own design choices, data models, and representations, offering a wide range of capabilities for managing and querying graph data [Jouili and Vansteenberghe, 2013]. To tackle the inherent complexity of graph databases, we have two most important graph models i.e. Labeled Property Graph and Resource Description Framework [Weikum and Schek, 1992].

2.2.2.1 Labeled Property Graph (LPG)

LPG, a commonly employed graph data model in databases like Neo4j, encompasses nodes and edges, each equipped with unique identifiers and the capability to be labeled and endowed with properties in the form of (key, value) pairs. Renowned for its robust performance, especially within graph database management systems, LPG is crafted for versatile graph data processing. It accelerates Create, Read, Update, and Delete (CURD) operations on data. The widespread adoption of LPG can be attributed to the success of graph databases management systems like Neo4j [Anikin et al., 2019].

2.2.2.2 Resource Description Framework (RDF)

RDF is a graph data model used in the Semantic Web for representing and linking data in a global Web of Data. RDF graphs are composed of ”atomic” nodes and edges, represented as triples, where the object might be a literal or another node identifier, the predicate is an edge label, and the subject is a node identifier. RDF provides a formal language for expressing data and is based on the idea of linking pieces of data from different domains to create a global Web of Data. RDF is designed to be semantically linked and integrated [Pan, 2009].

2.2.2.3 LPG vs RDF

When comparing RDF and LPG across various aspects, distinct differences emerge. RDF operates on a data model of triples, fostering a semantic, linked data representation facilitated by standardized serialization formats like RDF/XML, Turtle, or JSON-LD. It offers RDF Schema (RDFS) for formal schema definition and reasoning, supporting easy integration with external data sources and vocabularies through URIs and enabling ontology-based reasoning for inferring new knowledge. Additionally, RDF provides native support for federated querying across distributed data sources and boasts wide adoption and strong community support with standardized specifications from W3C. In contrast, LPG focus on graph structure and properties with a schema-flexible approach. While LPGs may lack standardized serialization formats and native reasoning capabilities, they offer powerful graph pattern matching capabilities in proprietary or custom query languages tailored to specific graph databases. LPGs are particularly popular in domains such as social networks, recommendation systems, and graph analytics, emphasizing direct relationships between entities and their properties. However, their adoption and community support may vary depending on the chosen LPG database and ecosystem, posing potential challenges in standardization and integration compared to RDF [Di Pierro et al., 2023]. Combining the semantic triples of RDF with the graph structure of LPG is a means of merging the two models. With the help of this hybrid process, a flexible graph representation that captures both semantic content and direct links can be created. It preserves the flexibility and graph-focused nature of LPGs while facilitating compatibility with RDF information.[Purohit et al., 2021]

2.2.3 Relational Database vs Graph Database

There are significant differences between Relational and Graph Databases, it depends on the purpose of research or implementation. Conventionally, relational databases are the most commonly used data storage option. However, with growing technologies and data talking options, we have graph databases which make our lives easier when it comes to Life Science data. Relational databases are still a suitable choice for many scenarios, especially when data is more tabular in nature, and ACID properties (Atomicity, Consistency, Isolation, Durability) are critical. The decision often involves considering the nature of the data, the types of queries, and the performance requirements of the application. Following is detailed description on how and what each database looks like in terms of key properties and functionalities [Vicknair et al., 2010].

Following is a table which enables us to understand better about the similarities and differences between Relational Databases and Graph Databases.

Table 2.1: Comparison between Relational and Graph Databases [Lazarska and Siedlecka-Lamch, 2019]

Property	**Relational Database**	**Graph Database**
Data Model	Organized into tables with rows and columns.	Utilizes graph structures with nodes, edges.
Schema	Requires a predefined schema.	Typically schema-less or schema-flexible.
Relationships	Relies on foreign keys.	All kind of operations are supported Relationships.
Use Cases	Well-suited for applications with well-defined and stable schemas.	Ideal for scenarios emphasizing relationships (e.g., social networks).
Query Language	Uses SQL (Structured Query Language).	Utilizes graph query languages (e.g., Cypher, Gremlin).
Performance	Suitable for complex queries.	Parallel processing and index free structure.
Scalability	Traditionally scaled vertically.	Designed for horizontal scalability.

An instance where relational database would be less helpful than graph database, we discuss about a social media platform like Facebook or LinkedIn [Walke et al., 2023]. On a platform like this:

- **Complex relationship**: Friends, followers, coworkers, and family members are just a few of the links that connect users to one another. These relationships are not always evident and may be connected to certain traits (such as the strength of a friendship, a person's employment, or a family connection). Dynamic Schema: Over time, relationship structures can alter and become more dynamic. A user can, for instance, create new partnerships, unfollow users, and add new friends.
- **Dynamic Schema**: Over time, relationship structures can alter and become more dynamic. A user can, for instance, create new partnerships, unfollow users, and add new friends.
- **Real-Time Suggestions and Insights**: Users anticipate receiving real-time suggestions and insights that are tailored to their behaviours, interests, and network connections. This necessitates navigating the network graph and efficiently querying intricate relationships.

2.2.4 Types of Graph Databases

2.2.4.1 Representation based on Storage

Table 2.2 tries to explain graph databases based on Storage.

Table 2.2: Graph Databases based on Storage Type

Storage Type	Property	Example
Native graph storage	These storage systems function according to the unit of nodes and edges and are specifically made for the administration and storage of graphs on disc. They are well-suited for complex relationship studies since they are excellent at supporting deep-link (multiple-hop) graph analytics.	Neo4J, Tiger-Graph
Relational Storage	Utilizing a relational model for storage, this method entails preserving separate tables for vertices and edges. During runtime, it employs relational join operations to concatenate these two tables	GraphX
Key-Value Store	This database model combines key-value pairs with a graph structure. Nodes are assigned unique keys, and relationships (edges) are stored using keys that link to connected nodes. This approach offers simplicity, scalability, and efficiency for scenarios where quick access to individual nodes and their relationships is crucial.	ArangoDB, Amazon Dy-namoDB.

2.2.4.2 Representation based on Data Model

The table 2.3 to explain graph databases based on Data Model.

Table 2.3: Graph Databases based on Data Model

Data Model	Property	Example
Property Graph	When a property graph is used, data is organized as nodes, relationships, and properties (data stored in nodes or relationships).	Neo4j, AWS Neptune
Hypergraph	Hypergraph is a graph theory information version wherein a connection also known as a hyperedge, can connect any quantity of given nodes. It permits any wide variety of nodes at both stop of a relationship. It proves beneficial when a dataset involves extensive many-to-many relationships.	HyperGraphDB
LPG vs RDF model	It is designed to manage and store data in the form of RDF triples. An RDF triple consists of three components: a subject, a predicate, and an object.	AWS Neptune, AllegroGraph.

2.2.5 Neo4j Database

Neo4j is a graph DBMS that is known for its native graph storage and processing capabilities. It implies that graph data is stored directly, hence streamlining storage and retrieval without the need for table mapping. Processing capabilities allow efficient implementation of graph algorithms directly on data. This results in speedier searches, scalable performance, easier query authoring, and deeper insights from complicated network topologies. It is designed to efficiently handle and traverse complex relationships within data. Unlike traditional relational databases, Neo4j is specifically designed for graph-oriented applications, which makes it more adaptable for scenarios where relationships between entities and the types of relationships are of high significance [Miller, 2013].

- **Graph**: The Graph visualization type in the Neo4j bloom (a tool exclusively designed for visualizing and exploring graph data in Neo4j) presents data in a visual graph format, showcasing nodes as circles and relationships as lines connecting them. This view provides an intuitive representation of the interconnected nature of the data, allowing users to explore relationships, identify patterns, and gain insights into the structure of their graph database.
- **Table**: The Table visualization type organizes data in a tabular format, presenting nodes and relationships in rows and columns. This structured view provides a

comprehensive overview of the data, offering details about each node's properties and relationships. It is particularly useful for a systematic examination of specific data attributes and their values.

- **Code**: The Code visualization type allows users to interact with the Neo4j database using Cypher, the query language of Neo4j. This view enables users to write, execute, and analyze Cypher queries directly. It is a powerful tool for users familiar with Cypher or those who prefer a text-based interface for querying and manipulating the data in a more programmatic manner.

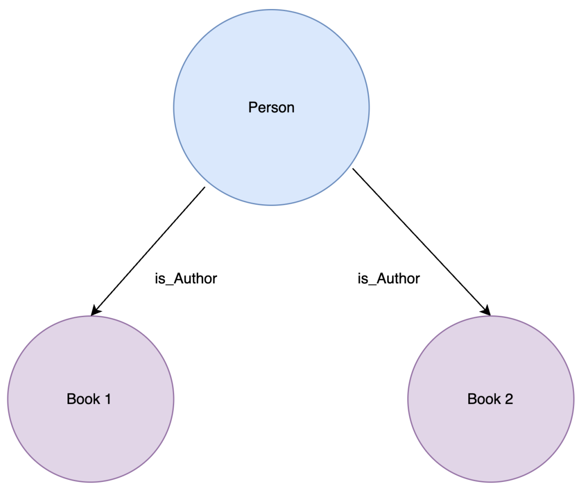


Figure 2.3: Graph Representation of 3 Connected Nodes, "Person" is the "author of" Book 1 and Book 2

There are several versions that we need to keep in mind for implementation for our setup. In our system we have used the following configurations:

2.2.6 Knowledge Graphs

Knowledge Graphs are a versatile and effective tool for expressing data and the complex interrelationships that exist within it. They are essentially data graphs designed to collect and disseminate knowledge about the real world, with nodes representing entities (objects, events, situations, or concepts) and edges representing the interactions between them. Knowledge graphs in computer science domain are advanced data structures that combine elements of semantic networks and graph theory to represent information in a structured and interconnected manner. They are essentially a network of real life objects, including items, actions, or ideas, as well as the connections between them. It allows you to organise and connect information, making it easier to analyse complicated relationships and extract insights from vast datasets [Nickel et al., 2015].

2.2.6.1 Types of Knowledge Graphs

- **Generic Knowledge Graphs**: These are broad-scope graphs that contain general information about the world. They are not limited to a specific domain and aim to cover as wide a range of facts as possible. For example, Google's Knowledge

Graph, which powers search and other services by connecting information about entities from various sources across the web [Lin et al., 2021].

- **Domain-Specific Knowledge Graphs**: These graphs focus on a specific domain or area of knowledge, such as medicine, finance, or music. They are tailored to provide in-depth information and relationships relevant to the particular field. For example, medical knowledge graph might include entities like symptoms, diseases, medications, and side effects, with relationships showing what symptoms are associated with what diseases, what diseases can be treated by what medications, etc [Abu-Salih, 2021].
- **Enterprise Knowledge Graphs**: Used by businesses to integrate and make sense of internal data. They link and contextualize various types of company data, from internal databases, documents, emails, and more. For example, A company might use an enterprise knowledge graph to connect data about its products, suppliers, customers, and internal processes to identify new business opportunities or improve operational efficiency [Galkin et al., 2017].
- **Social Knowledge Graphs**:These graphs focus on people and their relationships, interactions, and behaviors. They are often used in social networks and customer relationship management. For example, LinkedIn's graph connects professionals based on their work history, skills, and professional relationships [Yang et al., 2015].
- **Linguistic Knowledge Graphs**: These graphs are used to analyze and understand natural language text. They connect words, phrases, sentences, and sometimes more abstract linguistic concepts. For example, A linguistic graph might connect a word to its synonyms, its various meanings in different contexts, and its translations in different languages [Zhang et al., 2020].

2.2.7 Clinical Knowledge Graphs

Clinical Knowledge Graphs safely falls into the category of Domain Specific Knowledge Graphs. A Clinical Knowledge Graph is a specialized form of a knowledge graph that's particularly tailored for the healthcare and medical domain. It's a structured, graphical representation of medical knowledge that connects various clinical entities such as diseases, symptoms, medications, treatments, and patient data. The primary aim of a Clinical Knowledge Graph is to enhance medical research, improve healthcare delivery, and enable better clinical decision making [Huang et al., 2017]. They can also be explained as sophisticated data structures in the healthcare sector, designed to organize and interpret vast amounts of medical information by establishing a network of interconnected nodes and edges representing various medical entities and their relationships. These entities can include diseases, symptoms, drugs, patient demographics, and more, linked through relationships that articulate medical associations like symptomatology, causality, and treatment efficacy.

2.2.7.1 Pillars of Clinical Knowledge Graphs

- **Entities and Relationships**: The nodes (entities) in a Clinical Knowledge Graph can represent a wide array of clinical concepts including diseases, symptoms, diagnostic tests, patient demographics, treatments, drug interactions, and side effects. The edges denote the relationships between these entities, such as which symptoms are associated with which diseases, what medications are used to treat which conditions, or what the contraindications of a particular drug are.

- **Semantic Understanding**: Clinical Knowledge Graphs are designed to understand medical terminology and context. They make use of ontologies, which are formal explanations of a collection of concepts inside a domain and the connections within those concepts. For instance, ontologies in the medical field like SNOMED CT or ICD-10 provide standardized vocabularies for diseases, symptoms, and procedures, which can be used to enrich the graph with reliable data.
- **Evidence-based**:They often incorporate data from scientific research, clinical trials, and other authoritative sources. This ensures that the information in the graph reflects real world problems and solutions.

2.2.7.2 Applications of Clinical Knowledge Graphs

In essence, a Clinical Knowledge Graph is a powerful tool in terms of applications of the intersection of data science and healthcare, providing a structured and intelligent way to navigate the complex web of clinical information. This technology holds great promise for improving healthcare outcomes, advancing medical research, and personalizing patient care [Nicholson and Greene, 2020].

- **Drug Discovery and Repurposing**: By analyzing relationships between drugs, diseases, and biological pathways, they can aid in identifying new uses for existing drugs or in the development of new drugs.
- **Predictive Analytics**: Used in predicting disease outbreaks, patient outcomes, or the likelihood of disease progression.
- **Personalized Treatment Recommendations**: They can help in suggesting customized treatment plans based on a patient's unique medical history and current condition.
- **Research and Development**: Facilitate the aggregation and analysis of vast amounts of medical research data, helping in the advancement of medical science.
- **Enhancing Electronic Health Records**: By integrating with Enhancing Electronic Health Records, they can provide healthcare professionals with more contextual and relevant information.

The creation and ongoing maintenance of these graphs is fraught with difficulties, such as ensuring data quality, managing the sheer volume of information, maintaining privacy and security in accordance with regulations such as HIPAA [Moore and Frye, 2020], and continuously updating the graph to reflect the most recent medical discoveries and insights. Despite these obstacles, the future of clinical knowledge graphs looks bright, with potential advancements such as the incorporation of genomic data for personalised medicine, the use of AI for deeper insights and predictions, and the development of real time analytics for immediate clinical decision support. Clinical knowledge graphs have the potential to radically revolutionise the healthcare environment as they advance, improving the accuracy of diagnosis, the effectiveness of therapies, and the overall quality of patient care.

2.2.8 Graph Analysis

Graph analysis is a technique for studying and representing complicated networks as a collection of nodes and edges, such as biological, social, or technological systems. It entails analysing the relationships and patterns within these networks in order to find underlying structures, identify relevant nodes or clusters, and extract meaningful information. Networks are generally represented as graphs in graph analysis, with nodes (also known as vertices) and edges (sometimes known as links or connections). Edges reflect the connections or interactions between nodes, which represent the things inside the network, such as proteins in a protein-protein interaction network or persons in a social network. To investigate the properties and characteristics of these graphs, several computational tools and algorithms are used [Royer et al., 2008].

One of the key aspects of graph analysis is the identification of network motifs, which are recurring patterns or substructures within the network. These motifs can provide valuable information about the organization and function of the network. For example, in biological networks, motifs such as cliques (fully connected subgraphs) and bicliques (complete bipartite subgraphs) are often of particular interest due to their biological significance. The computation and analysis of alternative graph representations, such as power graphs, is also included in graph analysis. Power graphs are a unique network model based on two abstractions: power nodes and power edges. Power nodes are collections of nodes, and power edges connect two power nodes, indicating that all nodes in the first power node are connected to all nodes in the second power node. This method simplifies the investigation and visualisation of complicated networks by allowing for the concise depiction of network patterns.

Furthermore, graph analysis involves the development of algorithms and methodologies for tasks such as node clustering, module detection, network motif composition, network visualization, and network models. These techniques enable researchers to uncover hidden structures, identify biologically relevant modules, and gain insights into the connectivity and organization of the network. In addition to its applications in biological networks, graph analysis has broader implications across various domains, including social network analysis, transportation network analysis, and communication network analysis. It provides a powerful framework for understanding the relationships and interactions within these complex systems, leading to advancements in fields such as network science, computational biology, and data analytics.

2.2.8.1 Random Walks

Random walks serve as versatile tools with wide ranging applications across multiple disciplines, including mathematics, physics, computer science, and algorithm development. In the area of mathematics and physics, random walks model have various phenomena such as card shuffling and the erratic movement known as Brownian motion. Additionally, in statistical mechanics, they represent how particles move within a physical system. In computer science, random walks aid in exploring the depths of large datasets and in generating random elements within complex structures. Examples include navigating the points within a convex body or finding perfect matching in a graph. Furthermore, random walks play a crucial role in network analysis, statistical modeling, and in approximating solutions to looping problems. Their algorithmic applications are particularly notable, providing a robust framework for creating random elements from combinatorial structures. Random walks also enable efficient sampling techniques, which are integral to various algorithmic strategies. The broad utility and adaptability of random walks underscore their significance in both theoretical and practical applications within mathematical and computational fields [Barnes and Feige, 1993; Lovász, 1993].

2.2.8.2 Embedding

Node embedding is a technique used in graph representation learning, particularly in the field of network analysis and machine learning. It involves representing nodes (or entities) in a graph as dense, fixed-size vectors in a continuous vector space. Each node is mapped to a vector such that similar nodes in the graph are mapped to nearby points in the vector space. It is beneficial because it allows machine learning algorithms to operate on graphs by treating nodes as points in a continuous space, enabling the use of various mathematical operations and machine learning techniques that are not directly applicable to discrete structures like graphs [Abu-El-Haija et al., 2018].

The process of generating node embeddings typically involves iterative optimization methods that aim to preserve important structural properties of the graph, such as node proximity or network connectivity, in the embedding space. Following are the the types of embedding:

- **First-order proximity**: This type of node embedding calculates the pairwise node similarity based on the direct connections between nodes in the graph.
- **Second-order proximity**: In this type, the pairwise node similarity is calculated based on the relationships between nodes that are not directly connected, such as the k-step neighbors relations.
- **Community embedding**: This approach considers a community-aware proximity for node embedding, where a node's embedding is similar to its community's embedding. It aims to capture the similarities between nodes belonging to the same community.
- **Substructure embedding**: This type of node embedding focuses on embedding the graph structure between two possibly distant nodes to support semantic proximity search. It also includes learning the embedding for subgraphs (e.g., graphlets) to define the graph kernels for graph classification.
- **Higher-order proximity**: Some methods explore higher-order proximity to capture more complex relationships between nodes in the graph.

2.2.8.3 Node2Vec

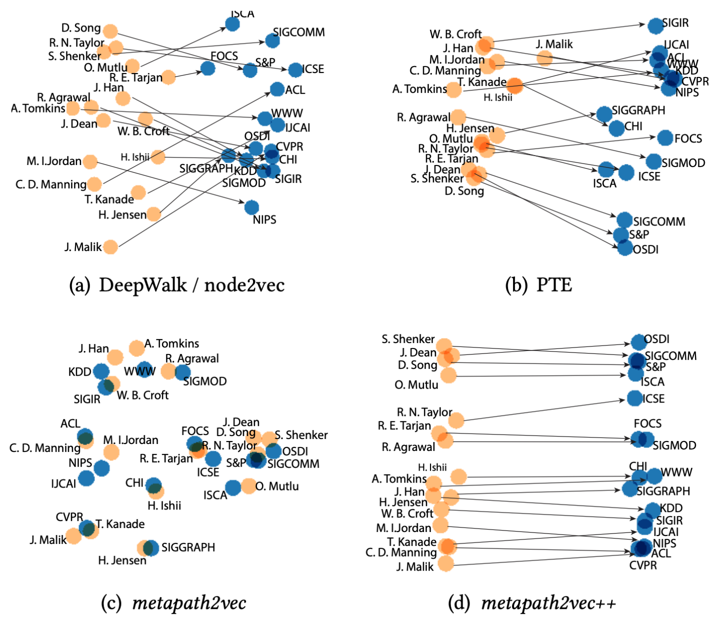
Node2vec is an algorithmic framework designed for learning continuous feature representations for nodes in networks. It aims to automate prediction tasks by learning the features themselves, with a focus on capturing the diversity of connectivity patterns observed in networks. The algorithm employs a flexible notion of a node's network neighborhood and a biased random walk procedure to efficiently explore diverse neighborhoods, allowing it to learn representations that calculates nodes with regards to what role they perform. Node2vec is a semi-supervised algorithm for scalable feature learning in networks, optimizing a custom graph-based objective function using stochastic gradient descent (SGD). It can learn representations where nodes have similar roles and similar embeddings, and it can learn representations that embed nodes from the same network community closer together. The algorithm has shown superior performance in multi-label classification and link prediction tasks over various real-world networks, demonstrating its effectiveness, scalability, and robustness to perturbations [Grover and Leskovec, 2016].

2.2.8.4 MetaPath2Vec

Scalable representation learning model Metapath2vec was created to meet the special difficulties heterogeneous networks pose. These difficulties result from the variety of nodes and linkages that exist, which restricts the applicability of traditional network embedding methods. The approach uses a heterogeneous skip-gram model (A neural network architecture used to learn word embeddings in natural language processing. In Metapath2vec, it is adapted to learn node embeddings in heterogeneous networks by predicting neighboring nodes along metapaths, capturing structural and semantic similarities between nodes) to produce node embeddings and formalizes meta-path-based (Metapath is a preset sequence of node and edge types that establishes a certain pattern in a diverse network. It directs the development of random walks, allowing nodes to go around the network on organized pathways) random walks to build a node's heterogeneous neighborhood. It also makes it possible to represent structural and semantic relationships in diverse graphs simultaneously. It has been demonstrated that Metapath2vec performs better than cutting-edge embedding models in a number of heterogeneous graph network mining tasks, including similarity search, clustering, and node categorization. As a useful tool for representing and analyzing big heterogeneous graphs, it captures both the structural and semantic relationships between various graphs nodes [Dong et al., 2017].

Metapath2vec++ goes a step further by enabling the simultaneous modeling of both structural and semantic correlations in heterogeneous networks. This additional capability allows metapath2vec++ to capture and represent the underlying structural and semantic relationships between multiple types of nodes in heterogeneous networks more effectively than metapath2vec. Overall, metapath2vec++ offers a more comprehensive approach to learning representations in heterogeneous networks by considering both structural and semantic correlations.

A visual comparison could be inferred from the following image and clear differences can be seen in terms of how they behave.



(a) DeepWalk / node2vec (b) PTE

(c) *metapath2vec* (d) *metapath2vec++*

Figure 2.4: Types of Embedding and it's comparison represented by "2D PCA projections of the 128D embeddings of 16 top CS conferences and corresponding high-profile authors" [Dong et al., 2017]

2.3 Web Application

A Web Application, is software that can be accessed in a web browser to access data, processing information and displaying content. There is no need of any installation of the software that needs to be run, rather is has a unique web address to access. The user is allowed to use different protocols to send and receive data between a browser and web server. The data the user intend to recollect is present on stationary servers, which are separately owned and maintained. Mostly, there is a real time back and forth data travel and it's amazingly fast. Every web application is built with a specific purpose and a thorough set of requirements, following is a list to illustrate the building blocks of a website [Garrett et al., 2005].

The rationale for utilising a web application for knowledge graphs stems from the need for quick access to massive quantities of information without requiring complex setup or extensive analysis. This enables democratised access to important information and significant insights from vast and complex data collections. The application can also generate random elements in huge and sophisticated sets, such as the set of lattice points in a convex body or the set of perfect matchings in a graph, which can then be asymptotically enumerated. Furthermore, the programme can give tools for investigating and understanding the relationships between random walks and eigenvalues, electrical networks, and other processes, opening up possibilities for random walks' applications in a variety of domains.

- **HTML**: Hypertext Markup Language (HTML) is a tag based language used for placing text on a website. This can contain text boxes, paragraphs, drop downs and buttons. Its fair to say that it is the skeleton of a website.

- **CSS**: Cascading Style Sheets (CSS) is language for styling a website. Everything from font styles to background colors is basically generated by this language. It helps the developers to broaden their visual aspect of webpage. Moreover, an enhanced responsive website is the ultimate target of CSS.
- **JavaScript**: JavaScript is a dynamic programming language primarily used in web development. Executed within web browsers, it enhances the interactivity of web pages by allowing the manipulation of HTML content. JavaScript is instrumental in creating dynamic features, facilitating actions like form validations and real-time updates without requiring a page refresh.
- **Python**: Python is a versatile programming language, is widely used in web development for creating dynamic and functional websites. Utilizing dedicated libraries, Python's readability and extensive package support to streamline the development process. Python is efficient at handling server side logic, database management, and the creation of web APIs, hence, making it easy to build scalable websites.
- **Java**: Java is fast, secure and object oriented Programming language, which is essentially used for server side development. There are different Java frameworks, which are built on top of it to attain specific development needs (i.e, Springboot and Mircronaut). Java posses dynamic runtime features like reflection and code modification, which helps in better compilation and efficient programming.

2.3.1 Frontend

In the world of web development, the 'front-end' specifically refers to the portion of a website or application with which users directly interact. This includes all the elements you see and engage with through a browser, encompassing the site's overall structure, aesthetics, and dynamic features. Developers create the front-end using coding languages such as HTML for structural design, CSS for aesthetic styling, and JavaScript for interactive functionality [Lara et al., 2013].

2.3.2 Backend

The back end refers to the server-side of web development, which includes the technologies and processes that occur on the server and are not directly visible to the user. It encompasses the logic, databases, and server-side scripting that enable the functionality of a website or web application. The back end is responsible for managing and processing data, user authentication, and server-side operations.

There are different types of back-end technologies, including traditional server-side scripting languages such as PHP, ASP.NET, and Ruby on Rails, as well as newer technologies like Nodejs, which is a JavaScript-based server environment. Additionally, there are various types of databases used in the back end, including relational databases like MySQL and PostgreSQL, as well as NoSQL databases such as MongoDB and Cassandra. These technologies and processes work together to power the functionality and data management of web applications [Connolly, 2019].

2.3.3 API

An API (Application Programming Interface) comprises a collection of protocols, methods, and tools essential for creating software and applications. It delineates the manner in which various software components should interact and offers a mechanism for diverse software elements to communicate. APIs facilitate the integration among disparate systems and devices [Zhou et al., 2014].

2.3.4 REST

A REST API (Representational State Transfer API) is a kind of API that follows to the REST architectural principles, which are important to creating network-based applications. To retrieve and manipulate data, these APIs communicate via HTTP. Because servers do not keep any session data for the client, each client request must be self-contained with all relevant information[Zhou et al., 2014]. The need of APIs in knowledge graphs is the sending and receiving of information like node data, edge data and further corresponding extracted values.

The image 2.5 shows how the data flows from frontend to backend while an API assists the process. The arrows between the components and teams indicate the direction of responsibility and interaction. The backend team develops the infrastructure that the API will serve, the API team creates the access point for the frontend to communicate with the backend, and the product team develops the user-facing part of the application that the consumers will use.

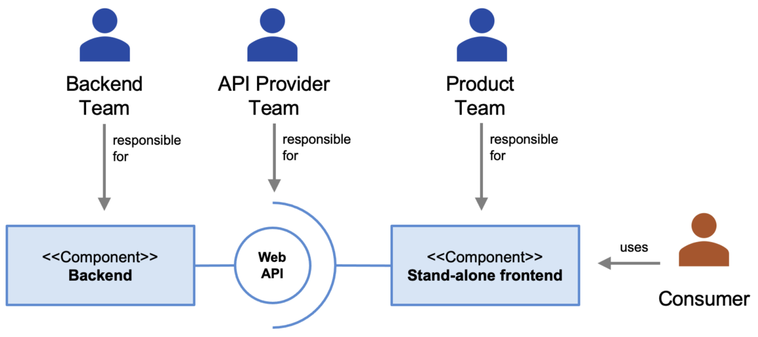


Figure 2.5: Abstract of a Frontend connects to a Backend using API, with responsible stakeholders [Bondel et al., 2021]

2.3.5 Types of Web Applications

- **Static Web Applications**: Static web applications are made up of unchanging content created using HTML, CSS, and occasionally JavaScript, which is used for basic interactivity and styling rather than dynamic updates or server-side data handling. These apps are noted for their quick loading times and enhanced security due to the absence of server-side processing or database interactions. However, static web apps are limited by their inability to dynamically respond to user inputs or update content without developer intervention.[Warren et al., 1999].

- **Dynamic Web Applications**: Dynamic web applications deliver content that changes in response to user inputs or other factors. They use server-side programming for real-time content generation, offering personalized experiences and interactive features. These applications are designed to scale effectively, handling variable user activity and data loads through advanced backend technologies that manage databases, server logic, and data caching. This scalability is crucial for supporting high traffic levels without excessive infrastructure expansion[Olston et al., 2005].
- **Single-page Applications**: Single-page Applications (SPAs) are web apps that load once and dynamically update content on the same page based on user actions. They primarily use JavaScript to manage data fetching and content rendering without requiring full page reloads, offering a smooth, application-like user experience. Frameworks like React, Angular, and Vue.js are commonly used to develop SPAs[Scott Jr, 2015].
- **Progressive Web Applications**: PWAs are designed to combine the features of native apps and traditional web applications, offering a seamless, cross-platform user experience. These applications work on any device and enhance functionality through features like offline accessibility and push notifications. PWAs aim to simplify development across multiple platforms while providing a consistent user experience akin to native applications, all accessible through a web browser [Majchrzak et al., 2018].

Each type serves different business needs and user experiences. For example, dynamic web applications are ideal for accessing and interacting with clinical knowledge graphs, providing real-time, tailored data to users in medical or healthcare settings.

2.3.6 Angular

Angular development involves using Angular, a comprehensive platform and framework that supports the construction of single-page applications using HTML and TypeScript. This framework equips developers with a robust set of integrated libraries for various functionalities including routing, forms management, and client-server interactions. Angular enhances HTML by adding additional attributes and uses an expressive and readable syntax to bind data, which simplifies the development of complex web applications[Fain and Moiseev, 2017].

Angular is utilized for its potent capabilities in creating interactive and dynamic web applications. It extends HTML's capabilities, facilitating the construction of sophisticated applications in an organized manner. The framework's dependency injection and modularity aid in structuring code effectively, making it easier to maintain, particularly for large-scale projects. Moreover, Angular optimizes performance and application efficiency through ahead-of-time compilation and tree shaking, reducing application size and improving load times. Tree-shaking is an optimisation technique and it's very useful when combined with JavaScript frameworks and tools like Angular. It illustrates the procedure for eliminating unnecessary code from a final package prior to production deployment. The objective of tree-shaking is to minimise the size of an application by removing unnecessary code, hence improving the application's overall performance and load times[Huang, 2019].

Using Angular to develop a knowledge graph website is highly effective due to its comprehensive features suited for complex and interactive applications. Angular's component-based architecture facilitates the construction of different user interfaces necessary for displaying and managing knowledge graphs. The framework's two-way data binding is essential for real-time UI updates when the underlying data changes, such as when nodes are added or modified. Angular's services and dependency injection offer a structured way to handle backend data interactions and state management, crucial for operations like fetching and updating graph data from a server. Angular's routing capabilities enable seamless navigation between different views or components within the application, enhancing the user experience in multi-page setups. Performance optimization techniques like lazy loading in Angular ensure that the application remains responsive and efficient as it scales, making it an ideal choice for building sophisticated knowledge graph interfaces that require robust functionality and interactive elements.

3. Related Work

In Chapter 3, I navigate through a comprehensive collection of pertinent literature within the domains of Knowledge Graphs, Querying Graph Data from front-end interfaces, and the vast array of Clinical Knowledge Graph analysis. Drawing upon this diverse sources, I try to extract valuable insights into the underlying principles, methodologies, and computational techniques that underpin the deep analysis of knowledge graphs and clinical datasets. Furthermore, I delve into the evolving landscape of querying graph data, shedding light on innovative approaches and tools designed to facilitate seamless interaction with complex network and graph structures.

3.1 The heterogeneous pharmacological medical biochemical network PharMeBINet

In the field of bioinformatics, heterogeneous biomedical pharmacological databases are pivitol in advancing medical discoveries, education, and diagnostic processes. PharMeBINet is a website that has the ability to visualize graph databases and helps researchers dive deep in a particular use case (i.e. Disease discovery). PharMeBINet is constructed by combining diverse entities and relationships from 29 public resources, including databases such as DrugBank (database of drugs) [Wishart et al., 2018], ClinVar (data of germline and somatic variants of any size, type or genomic location) [Landrum et al., 2016], and Comparative Toxicogenomics Database (includes chemicals, genes, phenotypes, diseases) [Davis et al., 2021], and integrating them into Neo4j. The merging process involves mapping entities from different data sources, using various strategies such as external identification systems or name mapping. The resulting database provides a rich source of information on drug-drug interactions, gene variations, and more. Moreover, the database holds significant potential for various research applications, including drug repurposing and the analysis of ADRs, genes, proteins, and diseases [Königs et al., 2022].

PharMeBINet addresses the necessity for a comprehensive database containing heterogeneous information on drugs, ADRs, genes, proteins, gene variants, and diseases. The integration process involves careful consideration of various data sources, mapping methods, and data validation to ensure the accuracy and reliability of the created knowledge graph. Furthermore, the PharMeBINet database undergoes technical validation to ensure that only human information is present for genes, proteins, pathways, and variants, and that the integrated data is filtered to include only relevant and reliable information with literature references or experimental evidence.

The main data sources integrated into the PharMeBINet database. These sources provide diverse information on drugs, gene variations, proteins, pathways, and diseases, which are integrated into the Neo4j database structure. The integration process involves careful consideration of data formats, mapping methods, and validation to ensure the accuracy and reliability of the database. Interestingly, PharMeBINet, here we can find a general overview of the website.

3.2 Prototyping an Interactive Visualization of Dietary Supplement Knowledge Graph

People are largely unaware about the effectiveness and safety of dietary supplements. There are different opinions regarding the safety and advantages of dietary supplements because they are not subject to the same regulatory approval requirements as general medications. People looking for health information online have increased significantly since the it is available. However, there can be variation in the quality and dependability of this information, which makes it challenging consumers to identify reliable sources. The information on dietary supplements that is now available is dispersed and usually not integrated. A coherent system that compiles data from multiple reliable sources into a single, simple interface is required[He et al., 2019].

ALOHA (dietAry suppLement knOwledge grapH visuAlization) is a tool that is developed to browse integrated DIetary Supplement Knowledge base (iDISK) with the help of interactive graph based visualization [He et al., 2018; Zhang et al., 2018]. The goal of the study is to improve ALOHA's usability and reliability in giving information about dietary supplements by developing and improving it through a user-centered design process. Iterative design and testing are used in the process to get user feedback and keep the system getting better. ALOHA seeks to make complex connections between dietary supplements and different health factors easier to understand and traverse through the use of graph-based visualisation tools. This study's ultimate goal is to improve public health by giving consumers a tool to utilise when making decisions about dietary supplements, therefore lowering the hazards that come with using dietary supplements without proper knowledge.

The implementation has the following features features like, Node Expansion, where by expanding nodes, users can examine the graph and observe relationships and direct connections. This enables consumers to view a supplement's interactions with medications, related medical problems, and other dietary supplements by clicking on the ingredient. Moreover, The tool retrieves and displays data using high-level semantic queries that take advantage of the knowledge included in the iDISK ontology. The underlying Neo4j graph database makes this easier by enabling queries to extract related data.

ALOHA was developed using an iterative design process that combined user-centered design with numerous rounds of focus group usability testing. Iteratively improving the interface and functionalities was done using the feedback from these sessions. To measure user satisfaction and identify areas for improvement, the usability of ALOHA was assessed using the system usability scale throughout the development process.

4. Methodology

Section 4.1 depicts the general overview and how the graph analysis is connected with the web application. This chapter will then explain my approaches to tackle the my research questions and how I came about particular solutions. The Section 4.2 indicates how the data was analysed and further transformed to be utilised our needs. Furthermore, a workflow is introduced for analyzing the graph, this workflow contains different abstraction levels each comprising different granularity of the underlying data. We talk about the implementation of the frontend, backend and the how they are structured. The Section 4.3 focuses on the hardware and software configurations required for the setup. Finally the Section 4.4 updates us with th requierment of the system.

4.1 Workflow

This is the first section where we understand how everything is connected to each other, the components and what purpose they have. Finally, we deeply discuss the Frontend, Backend and Implementation.

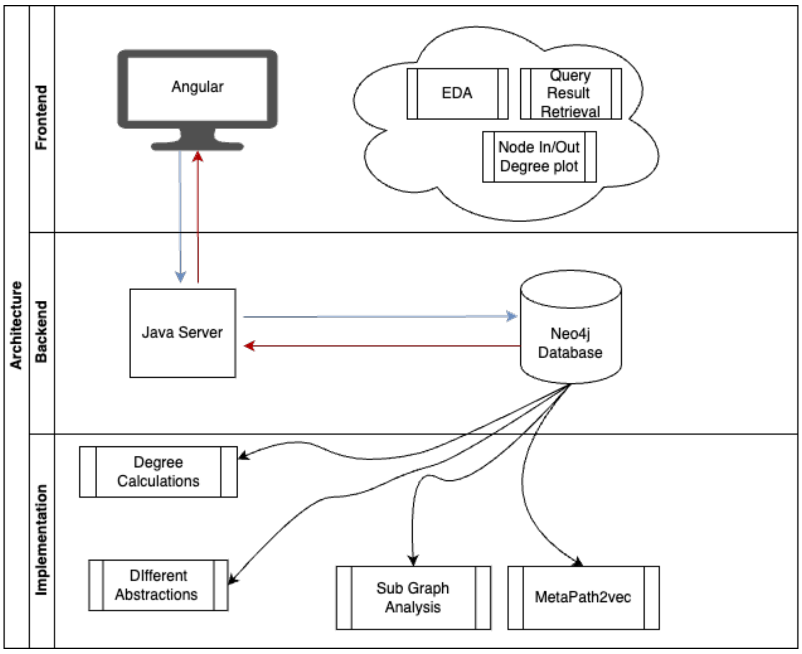


Figure 4.1: This is an image of Architecture of the Workflow and implementation components, where we see the communication patterns within and outside of frontend and backend. It also illustrated how Neo4j database is connected to multiple functionalities.

The figure 4.1 depicts software architecture with both frontend 4.1.1 and backend 4.1.2 components, as well as the implementation details of a system that uses Java server and Neo4j database.

4.1.1 Frontend

The frontend 4.1.1 is the user interface of the system, it sends and retrieves data from the Java server, and displays the results. This channel is accessible to everyone and basically the face of the the architecture. It is responsible to depict all the visualisations and query results.

4.1.2 Backend

The Java server is a key component which is responsible for all the data exchange. It enables date to be queried from the database and sends its back in a formatted way to the frontend to communicate. The graph database holds and handles all our data, which is of course in graph data format. The initial response of the graph is based in Json format, however, in the frontend, it then converted in a human readable form.

Another Python based process for backend is writing back the calculated node degree to Neo4j database. The first part is connecting to the database with neo4j driver, which is one of the most common ways to connect to any neo4j database instance. We then extract the data as tensors and calculate in degrees, out degrees and the accumulation of both, called degrees. After the calculation, the python script writes the calculated node degrees to the neo4j database instance.

4.1.3 Graph Analysis

Implementation phase has quite a few list of ownership as it consist of all the calculations, visualisations and graph learning capabilities. Following is a list of all the different activities taking place in this particular phase.

- Abstraction levels 4.2 are set for different conceptual visualisations and aggregations.
- Degree Calculations are made in this section via directly talking to the database and further more, written back to the graph database.
- We analyze the subgraph and run tests and specific nodes based analysis.

4.1.4 Communication Flow

Setting up the graph database was a critical component. The process began with downloading the necessary data dump onto a local machine. Following this, a Neo4j database was established to serve as the foundational platform for our graph database structure. After setting up Neo4j, the next step involved loading the previously downloaded data dump into the database on the server. This action set the stage for the integration and updating of the Clinical Knowledge Graph. To enhance and update the CKG, files from DrugBank, PhosphoSitePlus, and other sources were parsed. This method ensured that our graph database was not only set up with the initial data but was also updated with the latest and relevant information from various authoritative sources. This setup process was crucial for ensuring the database's accuracy and comprehensiveness, forming the backbone of our research infrastructure.

In figure 4.1, the double-sided arrow between the Java Server and Neo4j Database is a two-way communication for data retrieval, for updating the nodes, Python is used. The single red arrow pointing from the frontend to the Java Server refers the frontend that sends requests to the backend server. The single blue arrow pointing from the Java Server to the frontend indicates a response from the backend server to the frontend after processing data. The blacks arrows indicate that the calculations, shaping of data and analysis are directly connected to the database itself. The Degree calculations and the visualisation are done by exporting csv files specific to the use case. This is a complete data flow information, which intuitively informs how and when each set of data is being sent and received.

4.2 Data Abstraction and Processing

The image 4.2 shows the follow of increasing intensity of granularity throughout the analysis, starting from raw data and ending in a tensor format for graph learning.

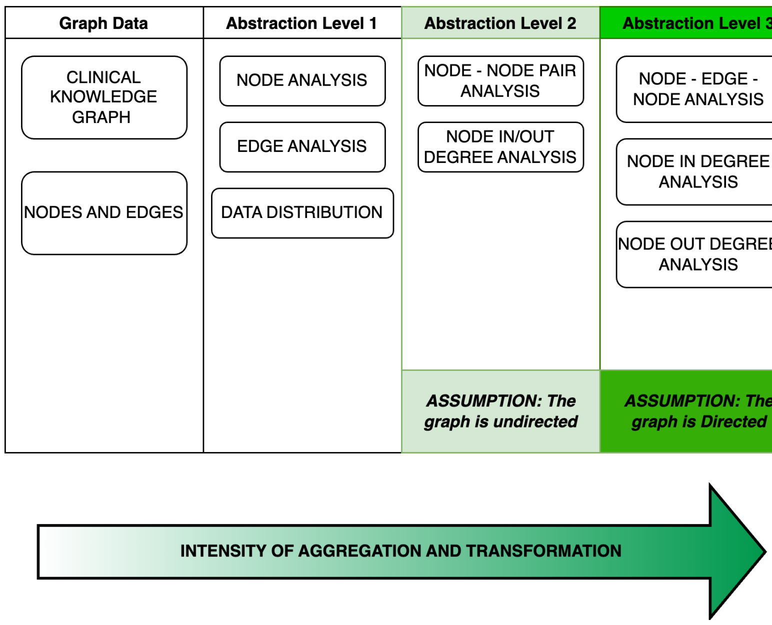


Figure 4.2: Levels of Abstraction of data based on Aggregation for Analysis. Each level defines a level deeper and granular data, it is done for better understanding and usage of the data. The levels of abstraction increases from left to right.

4.2.1 Graph Data

In this chapter, I plan to present a graph schema that includes various node types and edge types, aiming to enrich the comprehension of the graph's structure and the distribution of its data. Given the extensive assortment of node and edge types present, this work will selectively concentrate on those that are deemed most critical for our investigation. For a thorough examination of all node and edge types within the Clinical Knowledge Graph, please refer to the detailed analysis provided in the Appendix or Supplementary section of this document.

The first stage of the processing starts from the basic structure of graph data, which is Clinical Knowledge Graph [Santos et al., 2022]. The knowledge graph contains nodes and edges. It contains close to 20 million nodes and 220 million relationships, which includes literature, experimental data and public databases. There is an opportunity to add more data with relevant nodes to enhance further graph features and definition. However, we have majorly worked with the following 5 nodes and their related edges which are the more interesting part of this research.

Table 4.1 represents the nodes of interest and their relationships. Each node may have their individual set of relationships, which may or may not be a part of other nodes.

Node	**Relationship Types**
Disease	HAS PARENT
	STUDIES DISEASE
	DETECTED IN PATHOLOGY SAMPLE
	IS BIOMARKER OF DISEASE
	MENTIONED IN PUBLICATION
Drug	ACTS ON
	COMPILED TARGETS
	MENTIONED IN PUBLICATION
	INTERACTS WITH
	CURATED TARGETS
	TARGETS CLINICALLY RELEVANT VARIANT
Transcript	TRANSCRIBED INTO
	TRANSLATED INTO
	LOCATED IN
Protein	BELONGS TO PROTEIN
	IS QCMARKER IN TISSUE
	IS BIOMARKER OF DISEASE
	TRANSLATED INTO
	CURATED AFFECTS INTERACTION WITH
	DETECTED IN PATHOLOGY SAMPLE
	HAS STRUCTURE
	IS SUBUNIT OF
	VARIANT FOUND IN PROTEIN
	HAS MODIFIED SITE
	HAS SEQUENCE
	ASSOCIATED WITH
	CURATED INTERACTS WITH
	COMPILED INTERACTS WITH
	IS SUBSTRATE OF
	ACTS ON
	COMPILED TARGETS
	HAS QUANTIFIED PROTEIN
	ANNOTATED IN PATHWAY
	FOUND IN PROTEIN
	MENTIONED IN PUBLICATION
Gene	TRANSCRIBED INTO
	TRANSLATED INTO
	VARIANT FOUND IN GENE
	CURATED TARGETS

Table 4.1: Focused Nodes and with respective Relationship Types

4.2.2 Abstraction Level 1

In Abstraction Level 1, the data is shaped for the first time and it the most basic analsyis of the Clinical Knowledge Graph. The main topic is on the following aspects of data:

- Distribution and exploratory data analysis of Nodes.
- Distribution and exploratory data analysis Edge.
- Data Distribution

4.2.2.1 Node Type Count

An essential component of our effort was to carefully examine the the graph's nodes structure. Understanding the overall distribution and prominence of individual nodes was the goal of this investigation, which provided insights into the graphs's hierarchy and connection.The first step of our investigation involved a numerical evaluation in which the total number of nodes per node type in the graph was ascertained. This fundamental measure prepared the groundwork for a more thorough investigation of the complexity of the graph.

Subsequently, we directed our focus towards the extremities of the node distribution. By examining the the first ten nodes with highest and the lowest node degrees in terms of connectivity, we were able to emphasize the most and least central nodes within the network. This contrast offered a panoramic understanding of the graph's structure and this is done by investigating the number of nodes and their occurrences, highlighting influential nodes as well as peripheral ones. For a more intuitive visualization of our findings, we employed bar graphs to represent the node distribution. This approach facilitated the identification of patterns and outliers, making the data more accessible for further interpretation and analysis. The bars offered a clear depiction of each node's weight within the overall structure, with their lengths proportionally reflecting the count of connections.

Complementing the bar graphs, pie charts were utilized to present the node data. These pie graphs offered a segmental view of the nodes' distribution, allowing for an immediate visual comparison of the proportional representation of each node within the network. This is done by storing the each node type in newly created table and the visualisation takes place.

The outcomes of these approaches combine to provide a comprehensive understanding of the node dynamics inside the network, which is a strong foundation for network research in both theory and practice.

4.2.2.2 Edge Type Count

We then carried out a detailed examination of the edges that constructs the graph paths in our graph-based model. Determining the functional dynamics and interaction patterns that underpin the network's architecture required an understanding of the nuances of these linkages. We started by enumerating the edges, which encompass the whole range of connections that are woven across the network. This analysis represent a measure of the graph's connectivity and sets the foundation for a more profound edge analysis.

By classifying the edges into high and low interaction frequency categories, the analysis was further deeply looked into. We were able to highlight the most and least used pathways in the network by determining the top 10 and lowest 10 number of our edge set. Understanding the spread of connectivity and identifying possible bottlenecks or underutilised routes were made possible because to this differentiation.

Visual representation was integral to our analysis. Bar graphs were constructed to display the frequency of interactions across individual edges, presenting a clear depiction of connectivity distribution. Each bar's magnitude offered a direct correlation to the frequency of edge utilization, granting measurable connection densities. Simultaneously, pie charts rendered another perspective, showcasing the relative frequencies of edge usage as portions of the entire network. This pictorial depiction elucidated the edge distribution in a manner that was immediately perceptible and comparably analyzable.

4.2.2.3 Data Distribution

We have used Cumulative Distribution Function, refer to chapter 2 in order to analyse the spread of the data all over. It includes both node and edge spread. To encapsulate the interactions and distribution of the nodes and edges mathematically, we explored the use of a Cumulative distribution function. This mathematical representation was essential in illustrating the compounded relationships and interactions among the nodes, providing a data perspective to the otherwise visual and numerical data.

4.2.3 Abstraction Level 2

Abstraction Level 2 explains nodes being connected to one another and how the shape of data discovers if there is a Node-Node distribution in overall graph. There are two analysis being majorly catered. Node-Node Pair Analysis, Node Degree Analysis.

In the Node-Node pair analysis, we have merged each node pair combination in Clinical Knowledge Graph, which helps us understand to evaluate which node types are connected in the graph. We quantified these connections by their frequency, thus distinguishing between the various levels of pair engagements ranging from weak, occasional interactions to strong, recurrent connections. To effectively communicate the dynamics of node-node interactions, we leveraged a variety of visualization techniques, bar graphs and clustered heat map (seaborn library).

With the assumption that this phase of the analysis the graph is kept undirected and we don't assume any start or end node specifically. We then calculate the node degrees based with the help of the library "PyT Torch Geometric" and visualise it with histograms. The data is stored as feature matrix as tensor and edge index, then from edge index, we calculate node degrees. The binning of histogram was a challenge since the spread of the values was uneven and some of the nodes and edges have really high values with some with surprisingly low occurrences.

We were able to extract all the trends and the graphs representing them, however, only the five most interesting nodes are featured in this literature.

4.2.4 Abstraction Level 3

Inherent in our analysis was the assumption for this abstraction level that the graph is directed, which provided a richer context for interpreting the data. This directionality allowed us to infer the asymmetric relationships between nodes, offering clarity on the roles of individual nodes as either sources or targets of investigated edges in the graph. This perspective was particularly insightful for recognizing influential nodes and understanding the direction of processes within the graph.

Our research methodically investigated the in-degree distribution of nodes, reflecting the number of inbound connections to a node. This analysis highlights nodes that are significant receivers of information or influence, serving as critical junctions or aggregation points within the network. Nodes with high in-degrees emerged as potential key players, often indicative of authoritative or highly-regarded entities in the network.

Conversely, the out-degree analysis focused on the count of outbound connections from a node, illuminating those nodes that disseminate information or exert influence upon others. Nodes with elevated out-degrees, identical to in-degrees, were indicative of active broadcasters or hubs in the network, suggesting their role as originators or distributors in the communicative dynamics of the graph.

In the graphical representation and quantitative synthesis of our network, we have intricately shaped the Node-Edge-Node relationships to reflect complex associations such as ”Disease - Has Parent -Disease”. This not only shows the complex relationships within the graph but also clarifies the lineage and potential inheritance patterns that are pivotal in understanding the structure and connection of respective nodes. To understand the directional subtleties of the network, we have calculated the nodes’ in- and out-degrees by several graph investigations. Our research offers a dual perspective by differentiating between these two measures. It reveals the dynamic balance of influence and dependency by identifying nodes as either important recipients or sources within the graph.

To visually capture the distribution of node degrees, we employed bar charts, utilizing binning techniques to effectively categorize the range of in-degrees and out-degrees observed. The bin sizes were meticulously chosen to ensure clarity in the visualization of degree frequencies, allowing for easy and intuitive understanding of distribution patterns and anomalies within the degree data.We have used log scaling where the distribution was skewed for either very small or very high values in bar charts and cumulative distribution function plots.

4.3 User Interface Development

The goal of the user interface is to provide the users an interactive playbook, where users can easily query the graph database. The approach used is developing the frontend based on angular, while the backend is developed using java. The frontend 4.1.1 options to interact with the database, while the backend 4.1.2 maintains and shapes the data to be possible to be read and written. Frontend and backend structures are explained in this section and we argue their existence, need and process.

4.3.1 Frontend

Frontend in our system is basically the Angular based Web Application which is an essential part of the implementation. It allows technical and non-technical users to actively participate in the system. It is a generic application and since it is a proof of concept, we have use minimal styling i.e. CSS and SCSS.

Firstly, we have incorporated a querying mechanism, which directly contacts to our graph database (clinical knowledge graph) and brings back the data. It's very convenient to not have being connected to any database, or server, rather connected to the database through a web application. Results of the query are then displayed in web page.

Another use case of the frontend is dedicated to the non-technical users, where the users have the option of choosing nodes and edge types and then the results matching the query are displayed. This specific feature are required when the user has no scripting experience.

The third functionality of the web application is to input a list of proteins based on their accession codes. This will help to query the data based on the specific protein requirements. Many researchers investigate the proteome (section 2.1.3) or metaproteomes of different samples. Therefore, we provide a functionality to directly query proteins based on accession codes., hence, it's an intuitive idea to enable the exploitation of area.

One thing which is not covered is the handling on enormous data, since this is a proof of principle, we have restricted the query to be used with a limit statement. The reason for this is that our website can not handle enormous amount of data, it's a best practice to use limit and not get stuck with rendering. Additionally, some nodes are not covered in the proof of principle of the web application.

4.3.2 Backend

The backend of the web application is developed with Java and we used Eclipse IDE for the development and compilation. We have used Java Neo4j Driver to connect to the graph database. The goal of the backend is supply frontend with necessary data handling and storage. The backend is robust enough to handle the enormous data coming from the neo4j graph database, and provides a performant connection to the frontend. Real-time communication (for limited number of nodes) and an efficient and intuitive user experience are all made possible by this connection.

Through this backend infrastructure, our platform is well-equipped to meet the demands of graph data analysis, facilitating a seamless flow of information between the Angular based frontend, the Java backend, and the Neo4j database. We want to increase the accessibility for users without Cypher knowledge and without the necessity to setup the complete clinical knowledge graph independently.

4.4 Hardware and Software Requirements

The section below contains a comprehensive list of tables detailing the installation requirements for various systems and software used in the thesis. Each table is organized to provide clear, structured information on the necessary specifications, and software prerequisites needed for successful re-usage. The tables are designed to assist users in ensuring that their systems meet all the necessary criteria before proceeding with the installation, thus minimizing compatibility issues and optimizing performance.

Table 4.2 contains a list of requirements for Graph Database (Neo4j) for storing clinical knowledge graph setup.

Table 4.2: Environment Versions and Usage of Neo4j Tools in the Thesis

Tool	**Version**	**Usage**
Neo4j Desktop	1.5.8	For querying the database
Neo4j Version Database	4.3.0	This is the version of the used database
Neo4j Graph Data Science Library	1.6.1	Library used for graph learning features

Table 4.3 contains a list of requirements for frontend development (Angular) environment setup.

Table 4.3: Environment Versions and Usage of Angular for Development in the thesis

Tool	**Version**	**Usage**
angular/cli	**16.2.0**	for command-line interface for Angular
angular-devkit/build-angular	**16.2.0**	for development kit for building Angular applications
compodoc/compodoc	**1.1.22**	for documentation tool for Angular applications
ngx-formly/schematics	**6.2.2**	for tool for building dynamic forms in Angular

Table 4.4 contains a list of requirements for graph analysis setup.

Table 4.4: Environment Versions and Usage of Python for Development in the thesis

Tool	**Version**	**Usage**
Jupyter Notebook	6.4.8	for interactive computing
Python	2.7.18	for general programming
Pandas	1.4.2	for data manipulation
Numpy	1.21.5	for numerical operations
Seaborn	0.11.2	for statistical data visualization
matplotlib	3.5.1	for plotting graphs
torch_geometric	2.5.2	for handling torch type data
torch	2.2.1	for handling torch type data

Table 4.5 contains a list of requirements for Java environment setup.

Table 4.5: Environment Versions and Usage of Java for Development in the thesis

Tool	Version	Usage
com.sparkjava:spark-core	2.9.3	for micro framework for web apps
httpcomponents:httpclient	4.5.13	for support for HTTP client-side communications
httpcomponents:httpmime	4.3.1	for handling of MIME types
com.google.api-client:google-api-client	1.30.2	for access Google APIs
psidev.psi.tools:xxindex	0.16	for indexing XML files
com.compomics:mascotdatfile	3.6.1	for parser for Mascot MS data files
com.datastax.oss:java-driver-core	4.16.0	for cassandra database driver
org.neo4j.driver:neo4j-java-driver	4.1.1	for driver for Neo4j graph database
org.neo4j:neo4j-graphdb-api	3.0.1	for API for Neo4j graph database operations
com.google.code.gson:gson	2.8.5	for JSON processing library
commons-io:commons-io	2.8.0	for utilities for IO functionality
javax.xml.bind:jaxb-api	2.3.1	for Java architecture for XML binding
com.sun.xml.bind:jaxb-impl	2.3.1	for Implementation of JAXB API
org.apache.maven.plugins	3.2.1	for Maven plugin framework

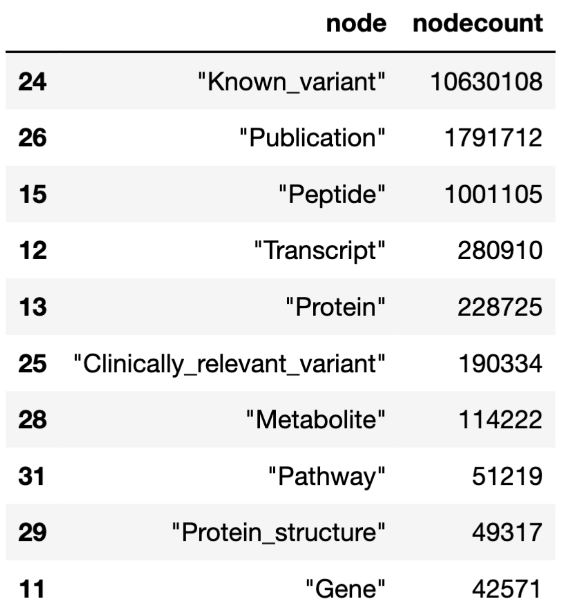
5. Evaluation

This chapter provides a thorough analysis of the findings at three different levels of abstraction specifically for Clinical Knowledge Graph. In Section 5.1, the results for abstraction level 1 are examined. Bar plots and cumulative distribution function plot are used as visual aids to help explain the results. Similarly, Sections 5.2 and 5.3 elaborate on the conclusions related respective Abstraction levels findings. In addition, Section 5.4 describes the frontend outcomes and looks closely at them. Finally, a thorough discussion is undertaken to address the research questions proposed in Chapter 1.

5.1 Findings of Abstraction Level 1

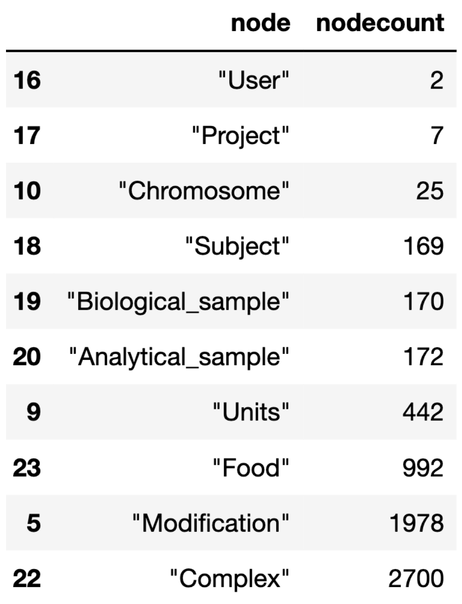
In Abstraction Level 1, exploratory data analysis is extensively done, we first take into account all node and edge count for an overall view. We divided 10 most abundant and 10 least abundant node types of the Clinical Knowledge Graph. It is clear that ”Known Variant” (10630108 nodes) and ”Publication” (1791712 nodes) were very dominant in terms of occurrence, however, ”User” (2 nodes) and ”Project” (7 nodes) had a quite low abundance. Known Variant node consist of different variants available in the clinical knowledge graph, hence it is in such high abundance. Publication is mentioning how many different publications are present in our vocabulary and moreover attached to each node. User is the a profile for addressing each node, whereas Project refers to the inclusion of those nodes into a specific working or research project.

We have only considered discussions based on our nodes of interests, with supplementary resources offering additional details. The goal of this abstraction level is to investigate the data distribution of the nodes type and edge type.



	node	nodecount
24	"Known_variant"	10630108
26	"Publication"	1791712
15	"Peptide"	1001105
12	"Transcript"	280910
13	"Protein"	228725
25	"Clinically_relevant_variant"	190334
28	"Metabolite"	114222
31	"Pathway"	51219
29	"Protein_structure"	49317
11	"Gene"	42571

Figure 5.1: Count of 10 most abundant nodes in the Clinical Knowledge Graph. Where "Known Variant" being dominates the table by 10630108 nodes, while the least abundant in this table is "Gene", which accounts to 42571 nodes of the total.



	node	nodecount
16	"User"	2
17	"Project"	7
10	"Chromosome"	25
18	"Subject"	169
19	"Biological_sample"	170
20	"Analytical_sample"	172
9	"Units"	442
23	"Food"	992
5	"Modification"	1978
22	"Complex"	2700

Figure 5.2: Count of 10 least abundant nodes in the Clinical Knowledge Graph. Where "User" being the most deficient in the table by just 2 nodes, while the most abundant in this table is "Complex", which accounts to 2700 nodes of the total.

Since the data seems very dispersed, we have created a cumulative distribution function to analyze the behavior of the node spread. We can assume that the overall distribution of all nodes are having a lot of variation. For the purpose of having a high-level overview, in fig. (5.3) a cumulative distribution plot is visualized.

According to the figure (5.3), the distribution of values is highly skewed, with most data points lying towards the higher end of the scale. The long flat stretch followed by a quick rise indicates that middle-range values are sparse or absent, concentrating most data points at higher values.

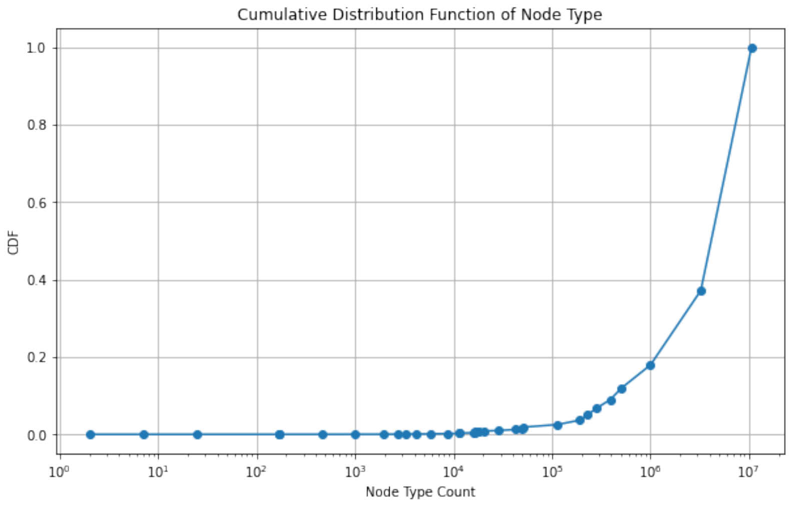
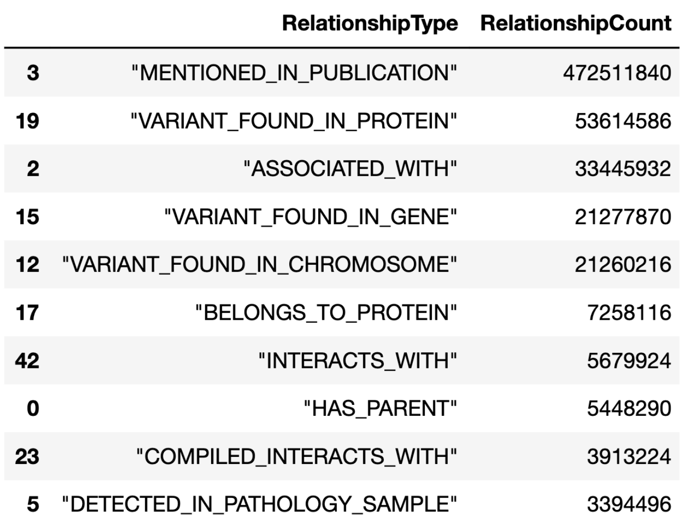


Figure 5.3: Cumulative Distribution Function plot based on all Nodes. The flat regions at the lower end of the x-axis indicate that there are large ranges of values where the cumulative probability does not increase, suggesting that there are no or very few observations in these ranges. The median (50th percentile) would be found at the point where the CDF is 0.5. This graph appears to reach 0.5 at a relatively high value on the x-axis, which would be consistent with a right-skewed distribution.

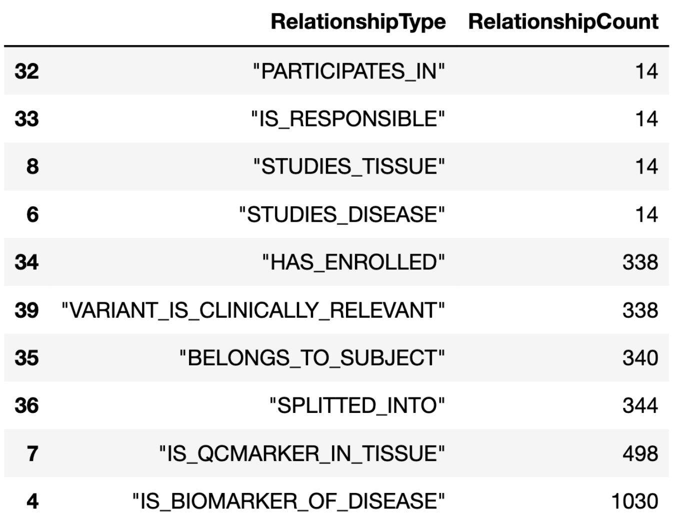
After the node type count analysis, we investigated the edge type counts for the complete clinical knowledge graph. There exists 43 different type of Edges and the following analysis is a comprehensive study of edge spread. Similar to node type count analysis, the edge type counts are also separated between 10 most and 10 least occurrences. In table 5.4 we can see that the highest value count is for ”Mentioned in Publication” (472511840) and ”Variant Found in Protein” (53614568), on the other hand, in the figure 5.5 the lowest edge count for ”Studies Disease” and ”Studies Tissue”.

Edge types connect two nodes and play a significant role to address how and why those nodes are connected together, For example the edge type ”Mentioned in Publication” shows that a particular node is present in a specific publication. Only when we see which two nodes are connected via this edge type, we can identify the linkage and also cause of high or low occurrences. It is very intuitive that ”Mentioned in Publication” is the most abundant edge type since most of the nodes would correspond to at least one, if more publication. Similarly ”Variant Found in Protein” is the next most occurring edge type, it related to the fact that different proteins (nodes) can be a variants of one and other and hence, connected to each other.



	RelationshipType	RelationshipCount
3	"MENTIONED_IN_PUBLICATION"	472511840
19	"VARIANT_FOUND_IN_PROTEIN"	53614586
2	"ASSOCIATED_WITH"	33445932
15	"VARIANT_FOUND_IN_GENE"	21277870
12	"VARIANT_FOUND_IN_CHROMOSOME"	21260216
17	"BELONGS_TO_PROTEIN"	7258116
42	"INTERACTS_WITH"	5679924
0	"HAS_PARENT"	5448290
23	"COMPILED_INTERACTS_WITH"	3913224
5	"DETECTED_IN_PATHOLOGY_SAMPLE"	3394496

Figure 5.4: The image explains the variation of 10 most occurring edges, having ”Mentioned in Publication” as the highest count (472511840), while ”Detected in Pathology Sample” (3394496) being the 10th value in the chunk.



	RelationshipType	RelationshipCount
32	"PARTICIPATES_IN"	14
33	"IS_RESPONSIBLE"	14
8	"STUDIES_TISSUE"	14
6	"STUDIES_DISEASE"	14
34	"HAS_ENROLLED"	338
39	"VARIANT_IS_CLINICALLY_RELEVANT"	338
35	"BELONGS_TO_SUBJECT"	340
36	"SPLITTED_INTO"	344
7	"IS_QCMARKER_IN_TISSUE"	498
4	"IS_BIOMARKER_OF_DISEASE"	1030

Figure 5.5: The image explains the variation of 10 lowest occurring edges, having ”Participates In”, ”Is Responsible”, ”Studies Tissue” and ”Studies Disease” having count equal to 14

The image 5.6 shows a CDF plot for all Relationships. The graph indicates a highly skewed distribution where a large number of entities have relatively few relationships. This is evident from the long flat sections of the graph at lower relationship counts. The point where the CDF reaches 1 indicates the maximum number of any relationship type there exist (i.e., ”Mentioned In Publication”). This provides a sense of the upper limits of connectivity within the graph.

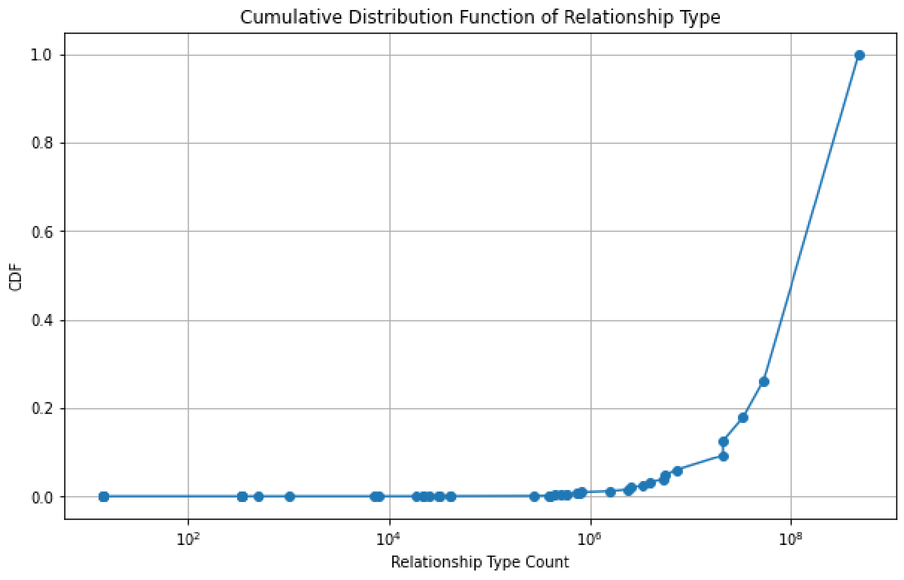


Figure 5.6: Cumulative Distribution Function plot based on all Edges, the x-axis represents the relationship counts, scaled logarithmically. This scale is used to manage a wide range of values, allowing for a clearer visualization of data when relationship counts vary over several orders of magnitude. The slow initial growth of the CDF at lower relationship counts implies that many entities in the CKG are sparsely connected.

The section 5.1 we learned that the highest node count belongs to ”Known Variant”, the lowest node count belonged to ”User”. Edge types with the highest count is ”Mentioned In Publication” and the lowest count is a share between 4 nodes, i.e., ”Participates In”, ”Is Responsible”, ”Studies Tissue” and ”Studies Disease”. Furthermore, we tried to reason as to why it makes sense to have the specific node and edge types in their particular count category. The node count CDF shows that most nodes have a relatively low count, but some nodes have extremely high counts, indicating a typical long-tail distribution. Similarly, the edge type count CDF reveals that while a majority of edge types have fewer abundance, a small number of edge types have a disproportionately high number of occurrences, suggesting the presence of influential edge types.

5.2 Findings of Abstraction Level 2

In abstraction level 2, we assume that the clinical knowledge graph is undirected, this means that we assume for simplification that is no specific incoming or outgoing edges from each nodes, rather it is directionless. Node Type pairs are constructed to analyze which of the nodes’ combination are most dominant. This specific use case helps to determine the link between two nodes and their dependencies on each other.

First, we visualized Node Type pair edge type in a log scaled clustered heat-map (Fig. 5.7). This shows that over all we have very few Node Type pairs that have very high occurrences, i.e., Disease and Publication. This is because, Disease and publication can have one edge type which is either high occurring or several edge types leading them to transpire together. We can also see that most of the values are 0, this is because most of the nodes don't have an edge type connecting them. This leads to fact that those connections are not dependent or related. An logical reasoning for this can be that a Disease can not posses an Amino Acid Sequence, hence, they are not connected.

The image 5.7 is a clustered heat-map, which shows the what two combinations of nodes reside together the most and least often.

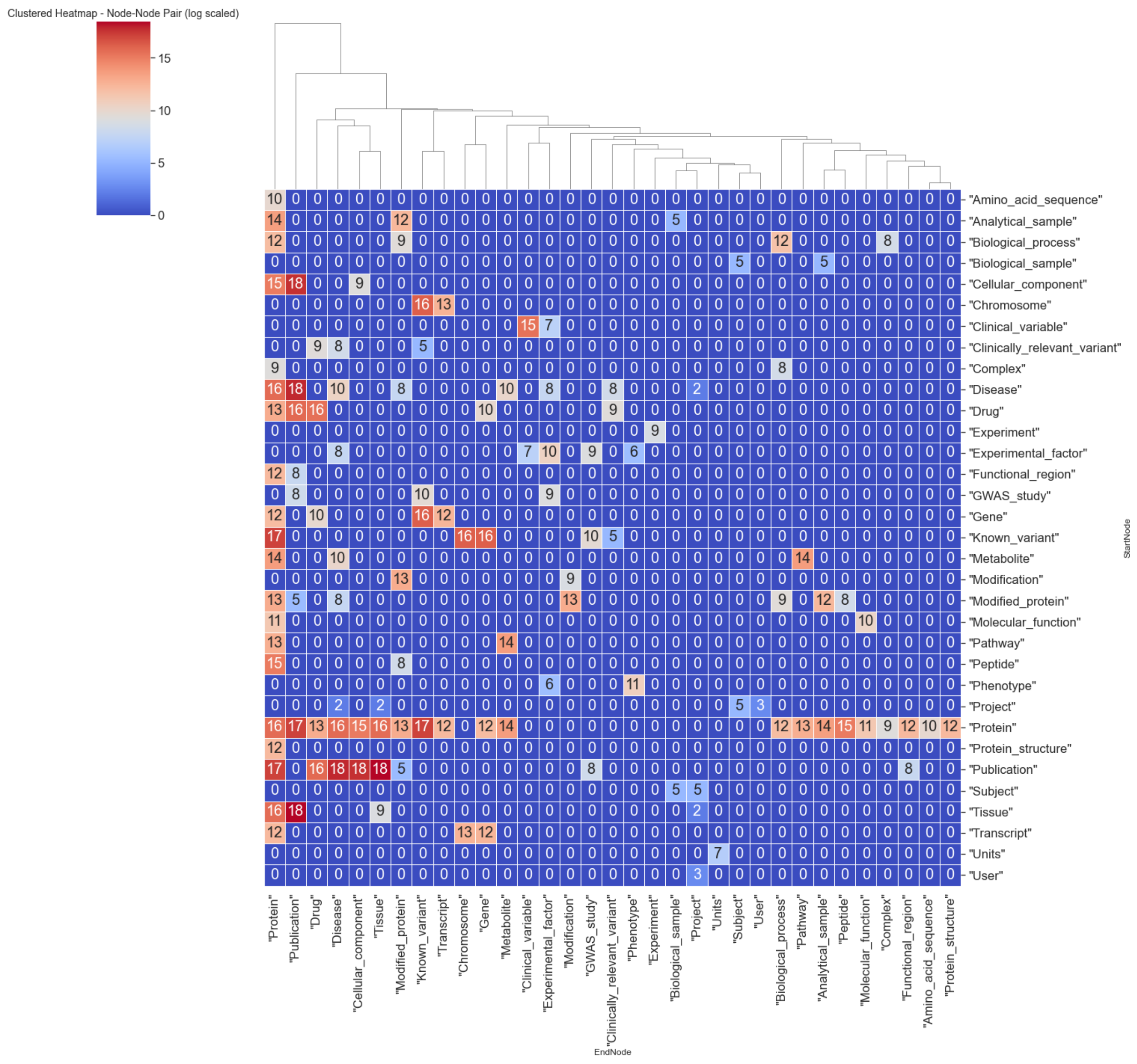
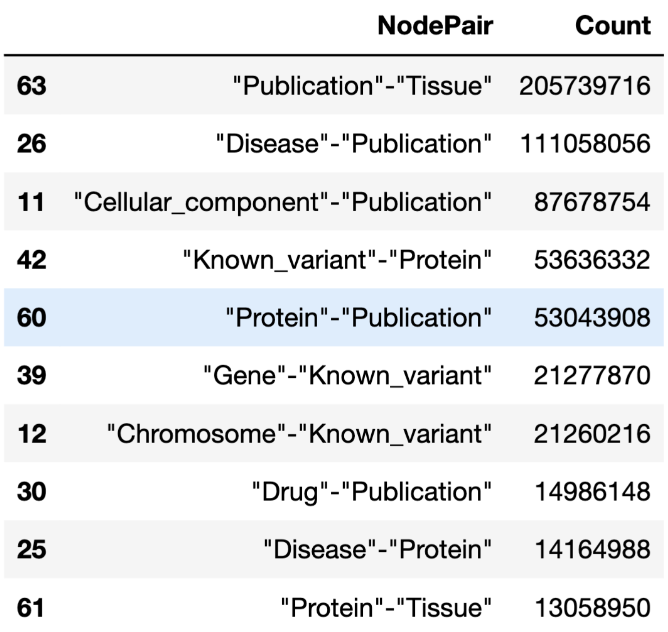


Figure 5.7: Clustered Heat-map based on Euclidean distance for all Node Type pair where we see that most of the connected nodes have 0 values, this is because most of the nodes are not connected to each other. Moreover, we have high connection for Drug, Disease and Modified Protein to Publication.

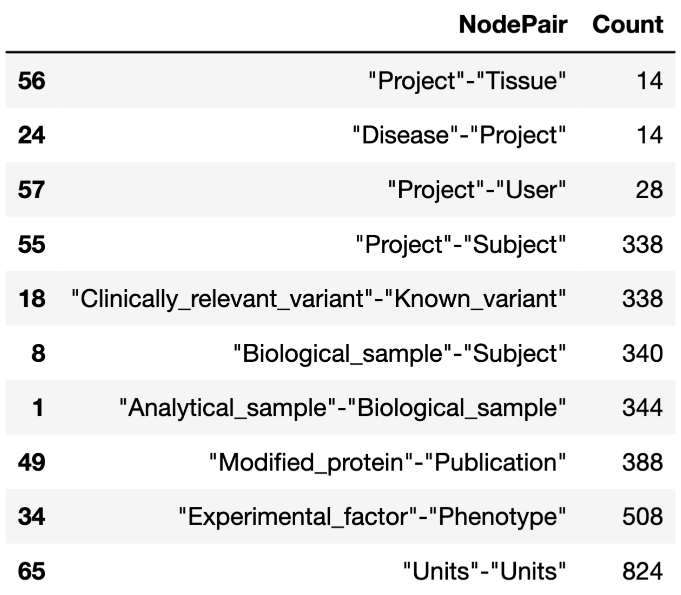
Figures (5.8 and 5.9) provide a detailed look into the extremes of our dataset: the most and least connected Node Type pairs, respectively. Subsequently, to comprehensively assess the interaction dynamics within our system, we also examine the cumulative distribution of Node type pairs, as shown in Figure 5.10.

The figure (5.8) shows that the top most abundant node type pairs are ”Publication-Tissue” (205739716) and ”Disease-Publication” (111058056), while figure (5.9)) illustrates that the node type pairs having lowest abundance are ”Project-Tissue” and ”Disease-Project” (14) each. From the analysis we understand that without the edge type, we have a connection between these specific nodes and we can expect the nodes like Tissue and Disease can be important the to graph.



	NodePair	Count
63	"Publication"-"Tissue"	205739716
26	"Disease"-"Publication"	111058056
11	"Cellular_component"-"Publication"	87678754
42	"Known_variant"-"Protein"	53636332
60	"Protein"-"Publication"	53043908
39	"Gene"-"Known_variant"	21277870
12	"Chromosome"-"Known_variant"	21260216
30	"Drug"-"Publication"	14986148
25	"Disease"-"Protein"	14164988
61	"Protein"-"Tissue"	13058950

Figure 5.8: The image represents the highest occurring nodes type pairs, i.e., ”Publication-Tissue” (205739716) and ”Disease-Publication” (111058056).



	NodePair	Count
56	"Project"-"Tissue"	14
24	"Disease"-"Project"	14
57	"Project"-"User"	28
55	"Project"-"Subject"	338
18	"Clinically_relevant_variant"-"Known_variant"	338
8	"Biological_sample"-"Subject"	340
1	"Analytical_sample"-"Biological_sample"	344
49	"Modified_protein"-"Publication"	388
34	"Experimental_factor"-"Phenotype"	508
65	"Units"-"Units"	824

Figure 5.9: The image represents the lowest occurring nodes type pairs, i.e.,"Project-Tissue" and "Disease-Project" (14) each.

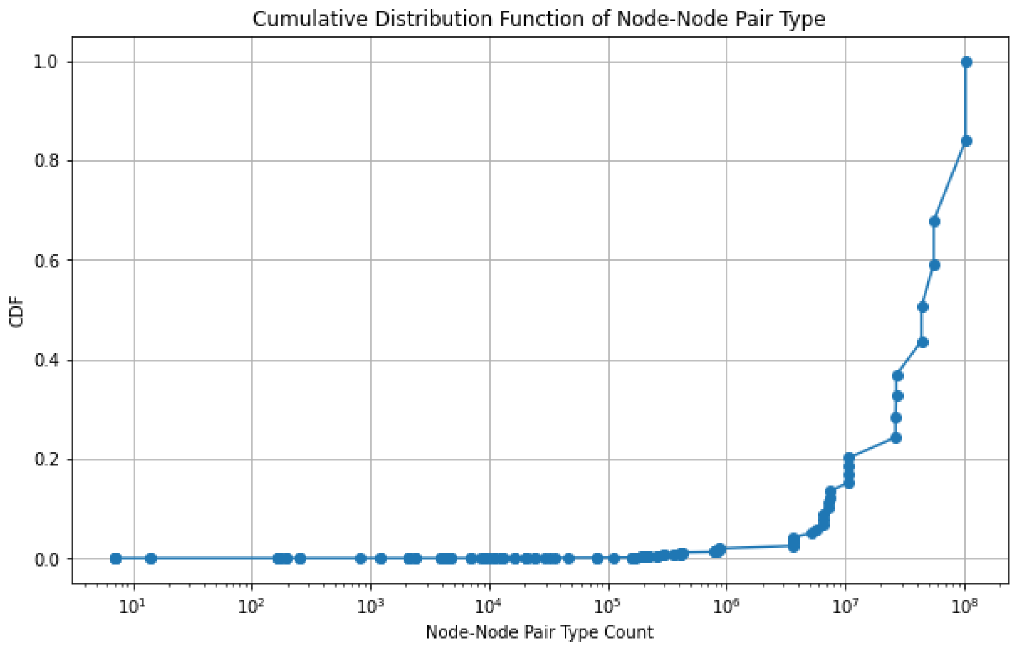


Figure 5.10: Cumulative Distribution Function Plot for all Node types pair, it shows how the CDF is distributed amongst the node type pairs.The graph covers a wide range of values on the x-axis, indicating significant variation in the dataset. The flat, almost horizontal section at the lower part of the x-axis suggests that a substantial proportion of the dataset has very low counts. This means that many entities or observations have minimal occurrences. The steep ascent later in the graph indicates that there is a rapid increase in cumulative frequency as the counts grow. The sharp vertical rise near the end of the x-axis indicates that the maximum count or highest value of the dataset is reached abruptly.

The reason for excluding nodes types is because the variation and spread of those nodes are very limited and hence no knowledge can be inferred from them. In the figure 5.11, let's take the node "Subject" as an example. It is very clear that very few nodes have high connections and majority of the nodes are having very few connections. This is a typical example of a highly skewed graph and this particular node adds little or no value to the research.

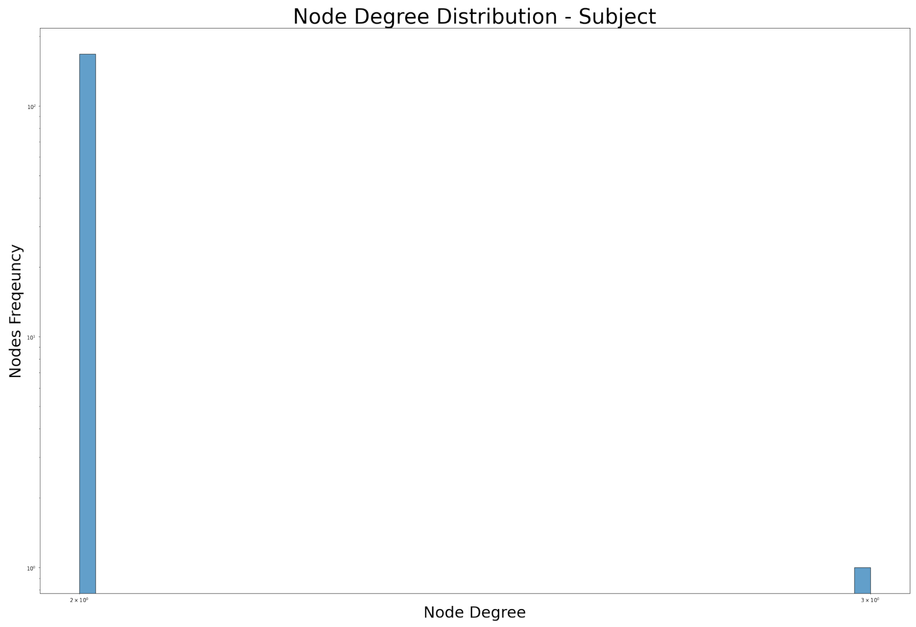


Figure 5.11: Node Degree Distribution for the Node "Subject", it show clearly that it contains low or no information that could contribute to our research. This the reason why we are focusing only interesting node types.

The second focus for this section is calculating and inspecting the frequency of node degrees for each node type. Nodes of interested are selected to have a better perspective of the nodes, since we have plenty of nodes but as discussed in figure (5.7), not all of the nodes contain interesting information. Furthermore, the analysis discussed in figure (5.12) is based on nodes of interest due to the low information in other nodes. Following is a list of interesting nodes:

- Gene
- Drug
- Transcript
- Protein
- Disease

Each histogram displays on the x-axis the node degree on a logarithmic scale, indicating the number of connections each node has within the graph. The logartihmically scaled y-axis, represents the frequency of nodes for each degree value, showing how many nodes of the considered node type have a certain number of connections.

The histograms for drugs, proteins, and diseases appear to show a right-skewed distribution, suggesting that a larger number of nodes have fewer connections, and a small number of nodes have a very high degree of connections. The transcript histogram has a different pattern, with several peaks indicating that there might be groups of transcripts that have a similar number of connections. The metabolite distribution seems to be less skewed than the others, suggesting a more uniform distribution of connections among metabolite nodes. There are a few outliers, e.g., in figure (5.12) there is one disease with a high degree. This is potentially because edge type ”Mentions in Publications” has a lot of connections to Disease so it might be some really unspecific disease and could be treated as an outlier.

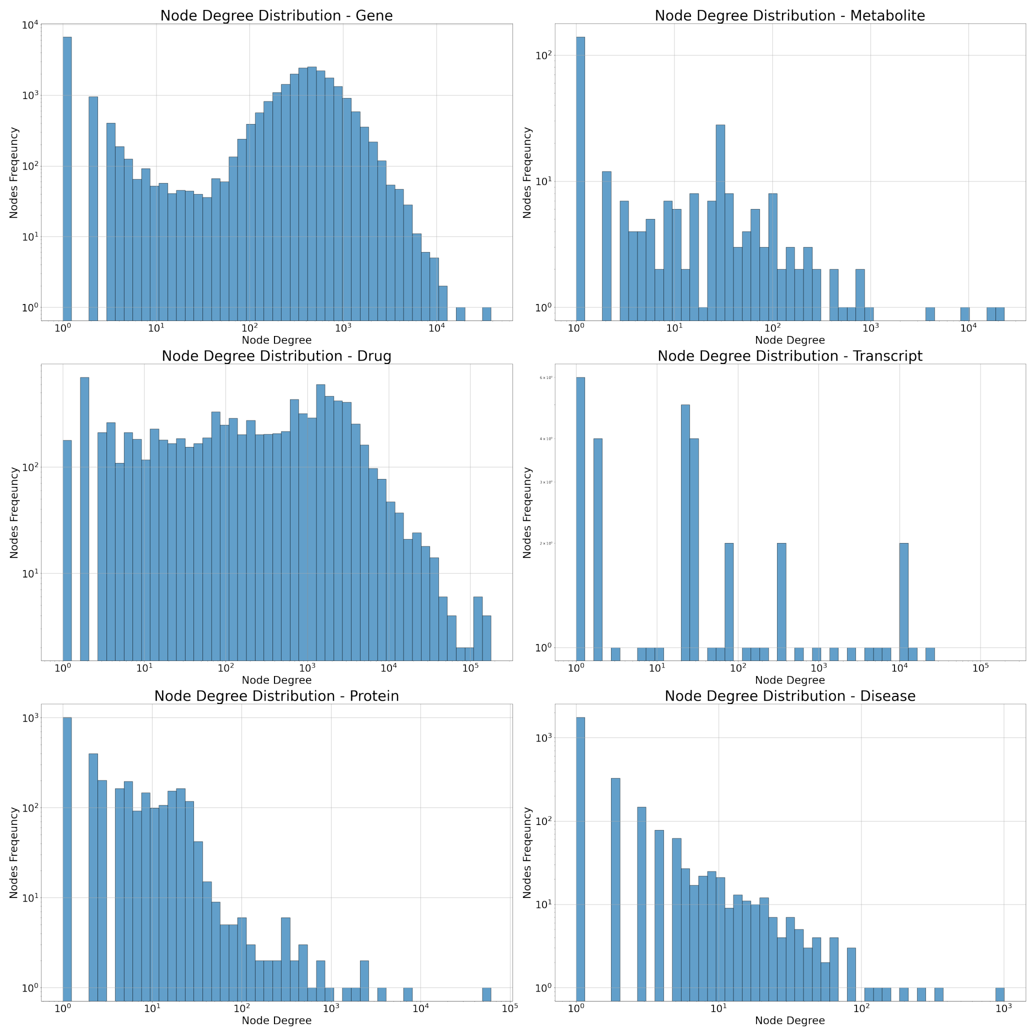


Figure 5.12: Node Degree Distribution for the interesting nodes i.e., Gene, Drug, Transcript, Protein, Disease and special mention Metabolite. All of the graphs show high distribution of node degrees, with Disease having an outlier in the first bar, while Transcript shows an odd distribution of node degree

Overall, in section 5.2, we understood via clustered heat-map that the occurrences of node types pairs like ”Publication Tissue” and ”Publication Disease” occur together quite frequently. Whereas, we highlighted what are the node types we included for a deeper look in terms of node degrees and why we eliminated some of the node types.

5.3 Findings of Abstraction Level 3

This abstraction level considers the graph to be directed, which means that we have to differentiate between incoming and outgoing edges with respect to each node of interest. Intuitively, we can infer that there would be a lot of combinations between each node and it's edges. This section focuses on all combinations where Protein is either start node or end node. The following graphs from (5.13 to 5.16) contains all the depiction of node degree layout.

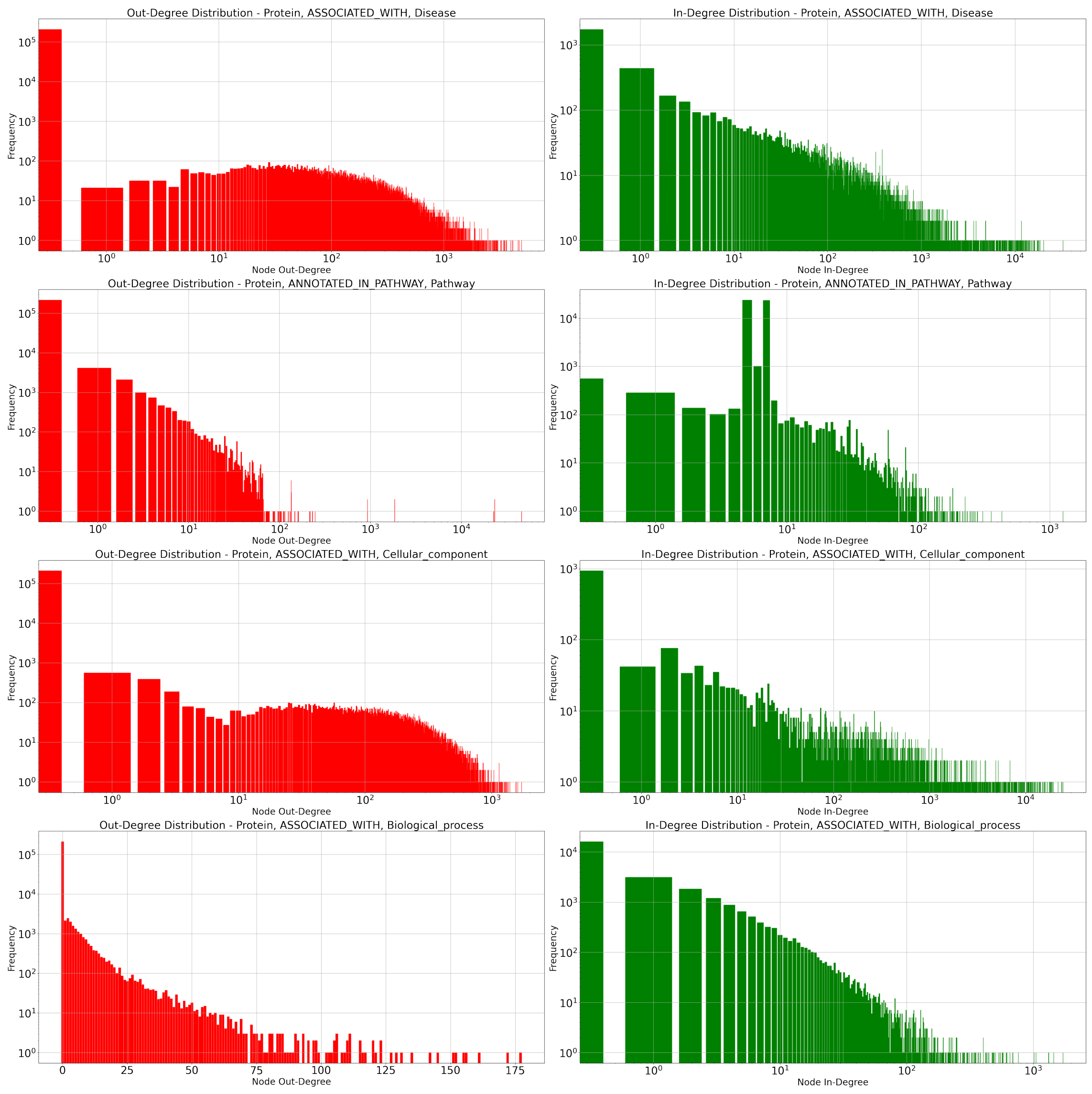


Figure 5.13: Protein In and Out Degree Distribution (1), here we see the node out degree in red and node in green. This figure shows "Protein" - "Associated With" - "Drug, Cellular Component and Biological Process" and "Protein" - "Annotated In" - "Pathway".

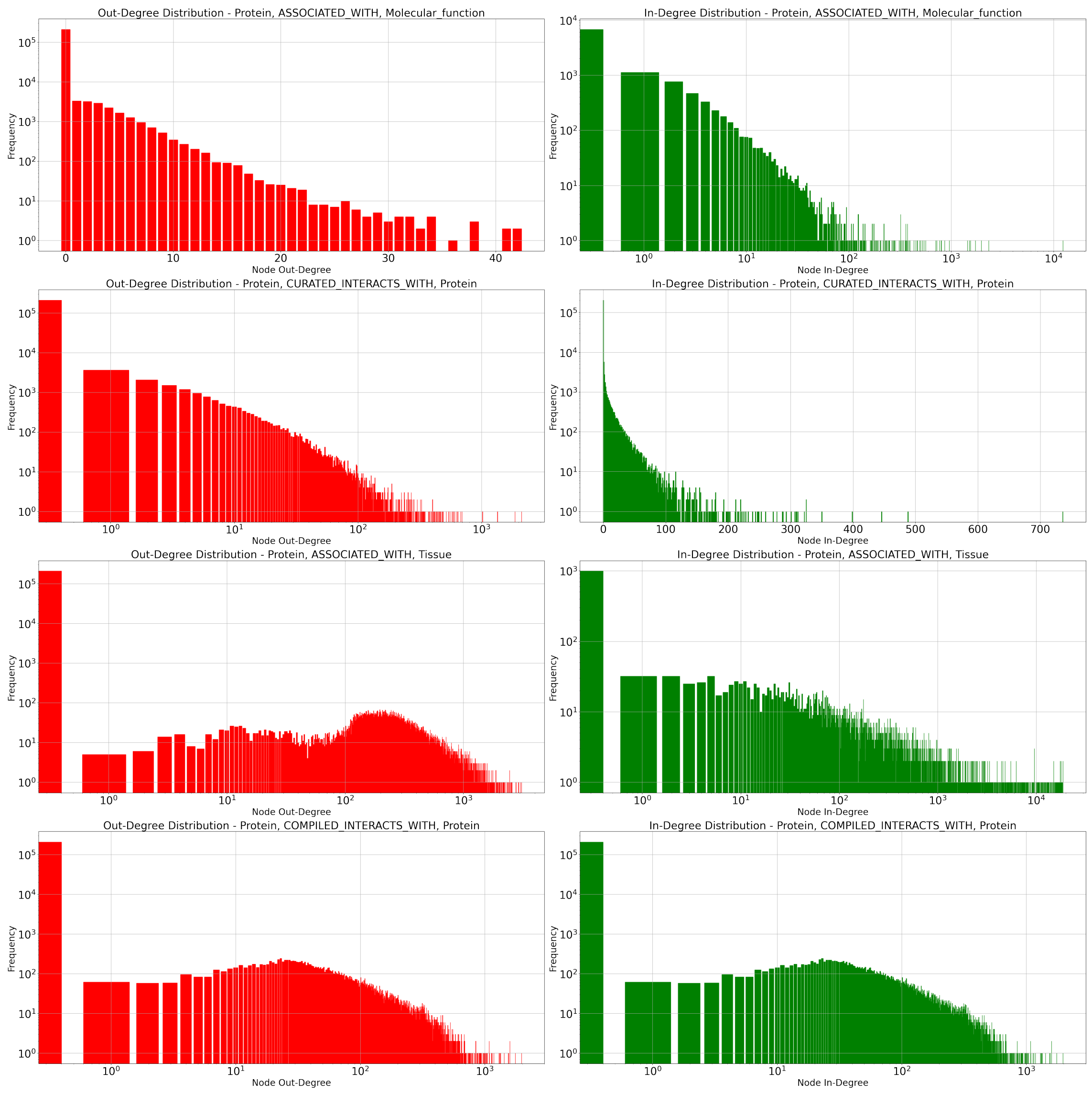


Figure 5.14: Protein In and Out Degree Distribution (2), here we see the node out degree in red and node in green. This figure shows "Protein" - "Associated With" - "Molecular Function and Tissue" abd "Protein" - "Curated Interacts With and Compiled Interacts with" - "Protein".

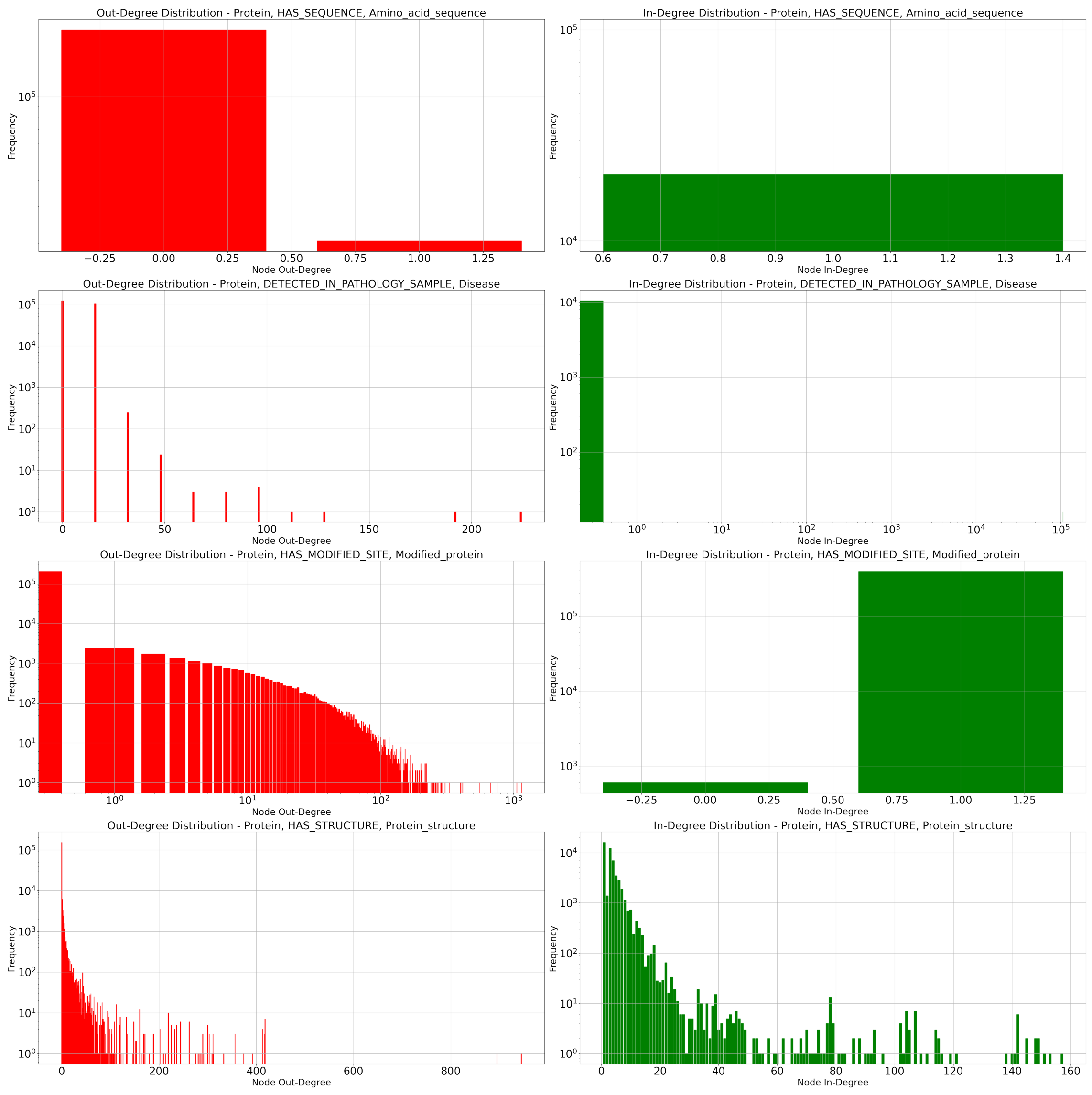


Figure 5.15: Protein In and Out Degree Distribution (3), here we see the node out degree in red and node in green. This figure shows ”Protein” connection to respective node types via certain edge types.

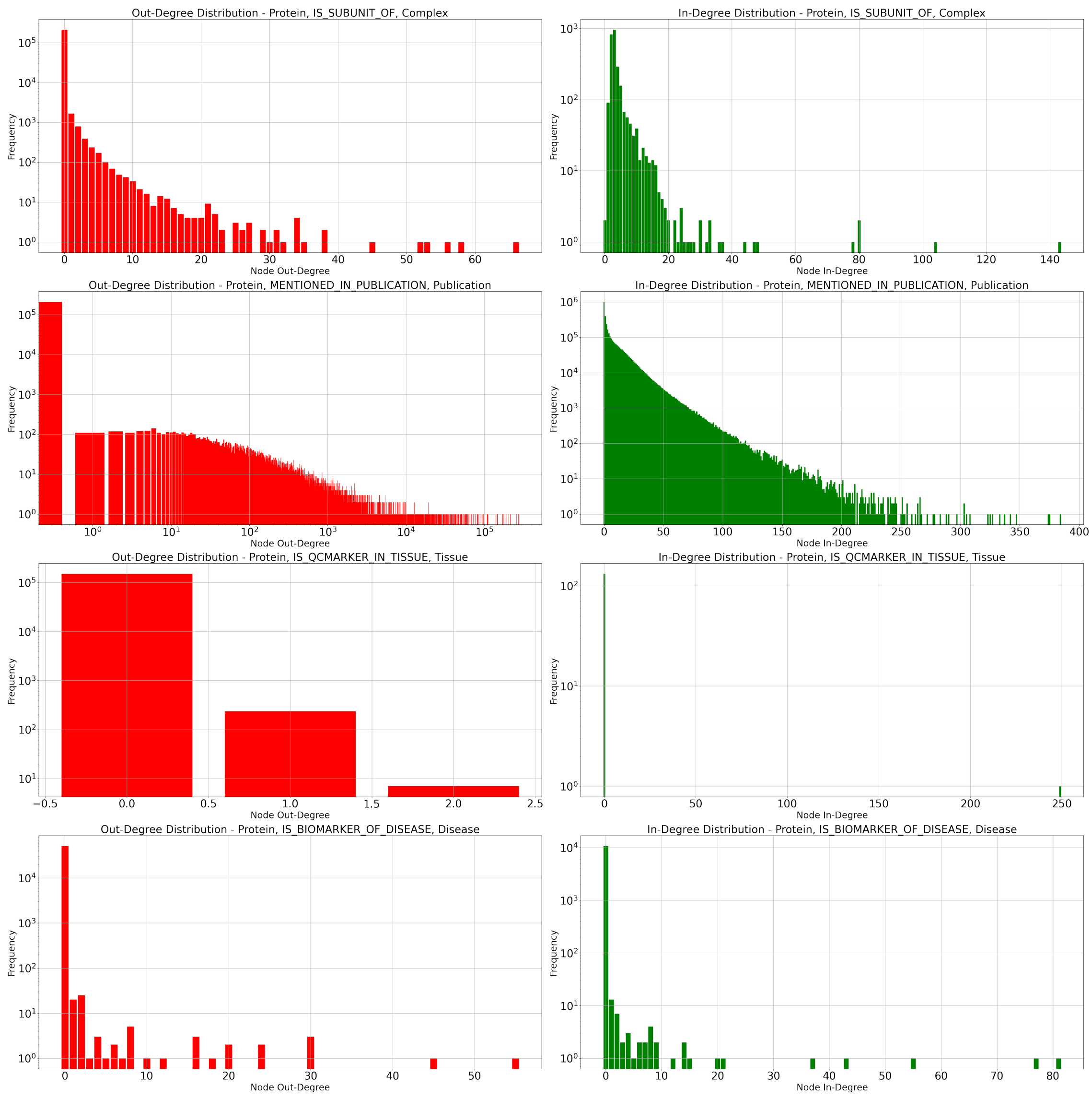


Figure 5.16: Protein In and Out Degree Distribution (5), here we see the node out degree in red and node in green. This figure shows "Protein" connection to respective node types via certain edge types.

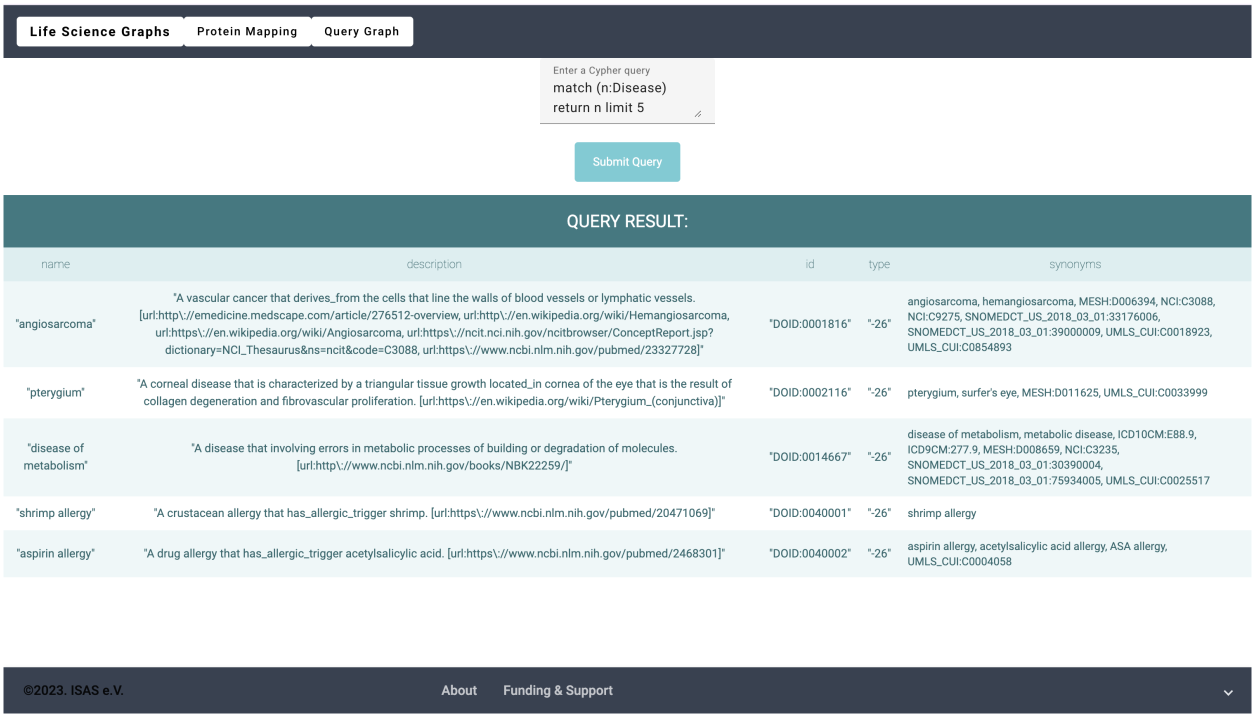
In degrees are the edge types which are coming into a node type, while Out degrees are the edge types which is heading out of a particular node type. The total in degree or out degree is essentially the count of incoming and outgoing edge types respectively. In the figures 5.13, 5.14, 5.15 and 5.16, we can infer that Out Degree for "Protein Associated with Disease", "Protein Associated with Cellular Component" and "Protein Associated with Tissue" have a fairly high magnitudes and the data is populated almost evenly with the growing number of out degrees. On the other hand, if we see the In Degree, ("Protein", "Associated With", "Disease"), ("Protein", "Associated With", "Tissue" and ("Protein", "Complied Interaction With", "Protein") have similar distribution to each other. Which essentially means that a lot of incoming edges are connected to the node Disease, Tissue and Protein.

5.4 Web Application

Web Application is designed for better accessibility of the the clinical knowledge graph. Through this web application, users can easily connect to and query the clinical knowledge graph. In this section, we will go through the frontend web application and its functionalities.

Query Graph Page is the first page where the users can type in a query and retrieve the results. From where a user can navigate to either Query Graph page 5.17 or Protein Mapping page 5.18.

In figure 5.17, a query panel is shown, where a user can type in a Cypher query and retrieve the data from clinical knowledge graph, without the setup or installation of any dedicated environment. This should lower the access barriers especially for users that have less expertise with python.

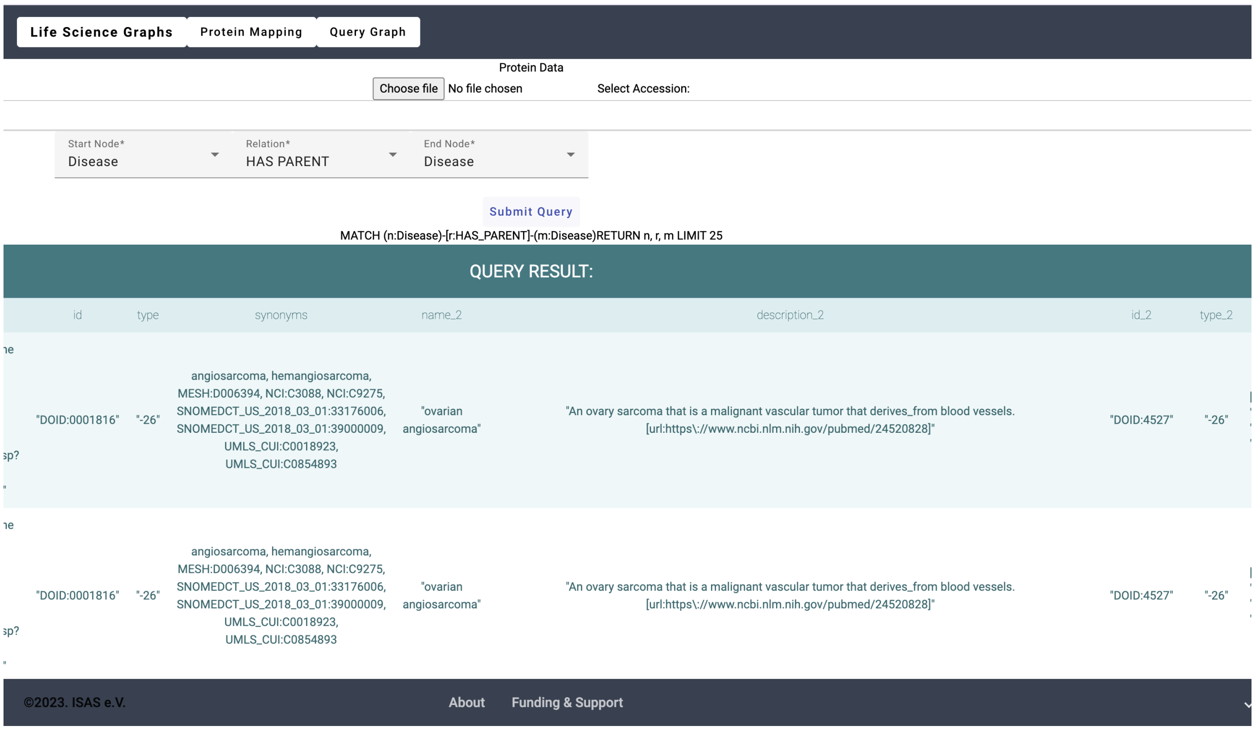


```
match (n:Disease)
return n limit 5
```

name	description	id	type	synonyms
"angiosarcoma"	"A vascular cancer that derives_from the cells that line the walls of blood vessels or lymphatic vessels. [url:http\://emedicine.medscape.com/article/276512-overview, url:http\://en.wikipedia.org/wiki/Hemangiosarcoma, url:https\://en.wikipedia.org/wiki/Angiosarcoma, url:https\://ncit.nci.nih.gov/ncitbrowser/ConceptReport.jsp?dictionary=NCI_Thesaurus&ns=ncit&code=C3088, url:https\://www.ncbi.nlm.nih.gov/pubmed/23327728]"	"DOID:0001816"	"-26"	angiosarcoma, hemangiosarcoma, MESH:D006394, NCI:C3088, NCI:C9275, SNOMEDCT_US_2018_03_01:33176006, SNOMEDCT_US_2018_03_01:39000009, UMLS_CUI:C0018923, UMLS_CUI:C0854893
"pterygium"	"A corneal disease that is characterized by a triangular tissue growth located_in cornea of the eye that is the result of collagen degeneration and fibrovascular proliferation. [url:https\://en.wikipedia.org/wiki/Pterygium_(conjunctiva)]"	"DOID:0002116"	"-26"	pterygium, surfer's eye, MESH:D011625, UMLS_CUI:C0033999
"disease of metabolism"	"A disease that involving errors in metabolic processes of building or degradation of molecules. [url:http\://www.ncbi.nlm.nih.gov/books/NBK22259/]"	"DOID:0014667"	"-26"	disease of metabolism, metabolic disease, ICD10CM:E88.9, ICD9CM:277.9, MESH:D008659, NCI:C3235, SNOMEDCT_US_2018_03_01:30390004, SNOMEDCT_US_2018_03_01:75934005, UMLS_CUI:C0025517
"shrimp allergy"	"A crustacean allergy that has_allergic_trigger shrimp. [url:https\://www.ncbi.nlm.nih.gov/pubmed/20471069]"	"DOID:0040001"	"-26"	shrimp allergy
"aspirin allergy"	"A drug allergy that has_allergic_trigger acetylsalicylic acid. [url:https\://www.ncbi.nlm.nih.gov/pubmed/2468301]"	"DOID:0040002"	"-26"	aspirin allergy, acetylsalicylic acid allergy, ASA allergy, UMLS_CUI:C0004058

Figure 5.17: Clinical Knowledge Graph - Query Panel of Web Application, the user have the space to type in a Cypher query and retrieve the results. This is the primary functionality of the Web Application.

The user can select node types and edge types and retrieve results. Figure 5.18 shows the drop down node type and edge type select utility. This functionality is designed for users who are not familiar with query languages, specifically, Cypher. The users have options to select either all, or any combination between start node type, edge type and end node type and the web application will return the respective results. For now, we have limited node types, however, incorporation of all nodes are possible and is scoped in future work. This should lower the access barriers especially for users that have less with Neo4j.



Protein Data

Choose file No file chosen Select Accession:

Start Node*: Disease

Relation*: HAS PARENT

End Node*: Disease

```
MATCH (n:Disease)-[r:HAS_PARENT]-(m:Disease)RETURN n, r, m LIMIT 25
```

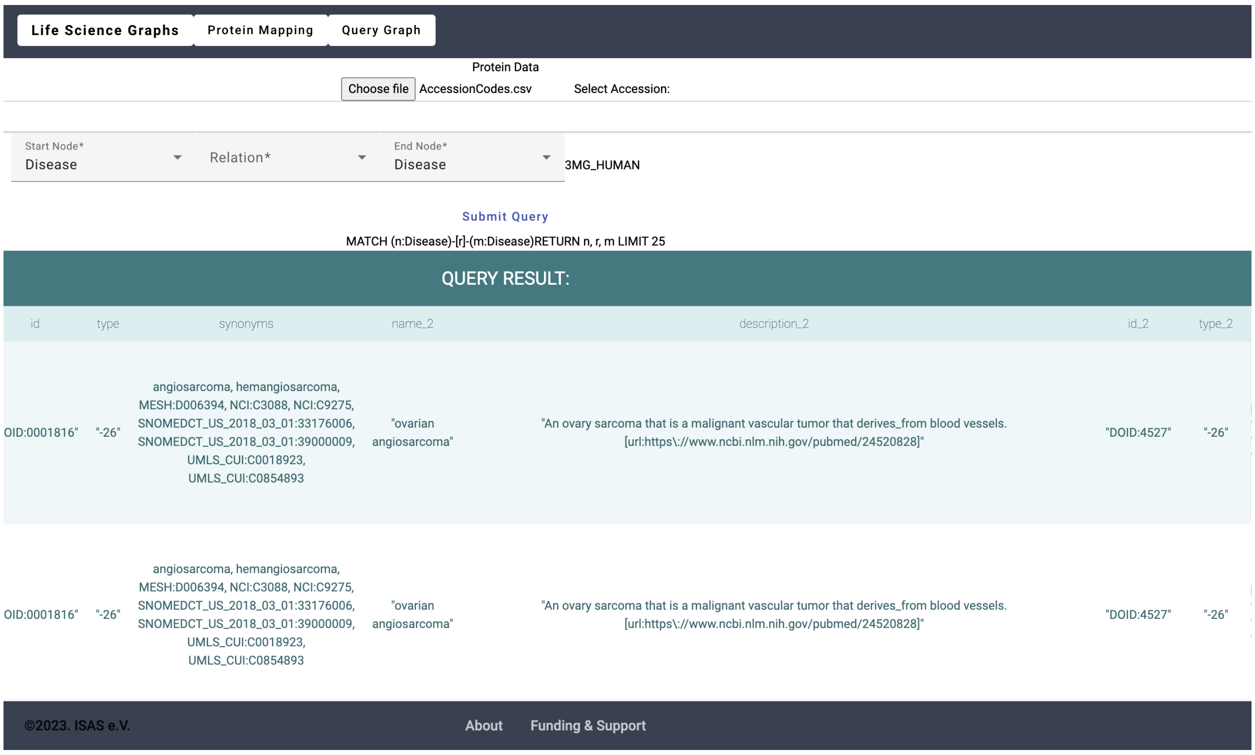
QUERY RESULT:

id	type	synonyms	name_2	description_2	id_2	type_2
"DOID:0001816"	"-26"	angiosarcoma, hemangiosarcoma, MESH:D006394, NCI:C3088, NCI:C9275, SNOMEDCT_US_2018_03_01:33176006, SNOMEDCT_US_2018_03_01:39000009, UMLS_CUI:C0018923, UMLS_CUI:C0854893	"ovarian angiosarcoma"	"An ovary sarcoma that is a malignant vascular tumor that derives_from blood vessels. [url:https\://www.ncbi.nlm.nih.gov/pubmed/24520828]"	"DOID:4527"	"-26"
"DOID:0001816"	"-26"	angiosarcoma, hemangiosarcoma, MESH:D006394, NCI:C3088, NCI:C9275, SNOMEDCT_US_2018_03_01:33176006, SNOMEDCT_US_2018_03_01:39000009, UMLS_CUI:C0018923, UMLS_CUI:C0854893	"ovarian angiosarcoma"	"An ovary sarcoma that is a malignant vascular tumor that derives_from blood vessels. [url:https\://www.ncbi.nlm.nih.gov/pubmed/24520828]"	"DOID:4527"	"-26"

©2023. ISAS e.V. About Funding & Support

Figure 5.18: Clinical Knowledge Graph - Drop down functionality, in this view users can use the drop down option to select nodes types and edge types. They result in tabular manner are then displayed for selected query. We can also see the Cypher query generated when a respective node type and edge type is selected.

Often researchers have lists of accession numbers of proteins that they are interested in (e.g., differentially expressed proteins). We provide a functionality to fill this gap. Figure 5.19 presents the capacity of the web application to input the desired list of Accession codes and select one or more codes from the list. The application then incorporates the selected codes into the query and dedicated results are displayed.



AccessionCodes.csv

Start Node*: Disease
Relation*
End Node*: Disease

3MG_HUMAN

MATCH (n:Disease)-[r]-(m:Disease)RETURN n, r, m LIMIT 25

QUERY RESULT:

id	type	synonyms	name_2	description_2	id_2	type_2
OID:0001816"	"-26"	angiosarcoma, hemangiosarcoma, MESH:D006394, NCI:C3088, NCI:C9275, SNOMEDCT_US_2018_03_01:33176006, SNOMEDCT_US_2018_03_01:39000009, UMLS_CUI:C0018923, UMLS_CUI:C0854893	"ovarian angiosarcoma"	"An ovary sarcoma that is a malignant vascular tumor that derives_from blood vessels. [url:https\://www.ncbi.nlm.nih.gov/pubmed/24520828]"	"DOID:4527"	"-26"
OID:0001816"	"-26"	angiosarcoma, hemangiosarcoma, NCI:C3088, NCI:C9275, MESH:D006394, SNOMEDCT_US_2018_03_01:33176006, SNOMEDCT_US_2018_03_01:39000009, UMLS_CUI:C0018923, UMLS_CUI:C0854893	"ovarian angiosarcoma"	"An ovary sarcoma that is a malignant vascular tumor that derives_from blood vessels. [url:https\://www.ncbi.nlm.nih.gov/pubmed/24520828]"	"DOID:4527"	"-26"

©2023. ISAS e.V.

Figure 5.19: Clinical Knowledge Graph - Accession List Plugin, is the figure, we see an option of inserting a custom accession file, which can be used to filter specific proteins. This is aimed for researchers with dedicated uses cases like drug re-purposing.

The web application has a room for further extension, it is discussed in the Chapter (6). The use cases like finding biomarkers, finding associations between individual omics layers like genomics, transcriptomics and proteomics, identifying a disease based on differentially expressed proteins, finding drugs for treatment (drug reprurposing), can we optimistically solved via the web application.

5.5 Discussion of Research Questions

In Section 5.5 we answer the Research Questions which were proposed in Section 1.2.

Research Question 1: How can we visualize different graph properties in terms in of Nodes, Edges and their structure?

Answer: The visualisation of Nodes and Edges are done in abstraction level 1 (5.1), where the most occurring nodes and least occurring nodes are highlighted. Furthermore, The Cumulative Distribution Function (CDF) graphs for node and edge type counts provide insights into the distribution of connections within a dataset. The node count CDF reveals that while the majority of nodes have relatively low counts, a long-tail distribution is indicated by some nodes having exceptionally high counts. Comparably, the edges count CDF shows that some edges have disproportionately, indicating the presence of important nodes within the data, whereas the bulk of edges have less abundance.

Research Question 2: What can we learn from different abstraction levels of the Clinical Knowledge Graph?

Answer: In section (5.2 and 5.3), I have answered the learning from different abstraction levels. It is clear that the most frequent occurring node types pairs are (”Disease and Publication”, ”Cellular Component and Publication” and ”Tissue and Publication”). This is due to the node ”Publication” having high relation with these nodes. Moreover, the nodes of interests are identified to be ”Protein”, ”Drug”, ”Disease”, ”Gene” and ”Transcript” and their node degree suggest a very excessive connections to these nodes. Finally, we scoped the abstraction level 3, where the graph is assumed to be directed, we see that the node Protein has an intense in-degree and out-degree connection, specially, (”Protein”, ”Associates With”, ”Tissue”) and (”Protein”, ”Associates With”, ”Disease”).

Research Question 3: What are user-friendly methods for querying data from a Knowledge Graph via a front-end interface?

Answer: In Section (5.4), the methods of querying data from Knowledge Graph via a front-end interface is discussed. The Web Application has 3 main functionalities, a query panel to type in Cypher query and retrieve results, which takes no installations and downloads for any knowledge graphs. The second feature which eases the user to use drop down instead of writing queries manually, it helps non-technical users to query the clinical knowledge graph. An option of inserting a list of accession codes for use case specific researchers, is included. This is mainly for researchers finding Drugs for Treatments etc.

6. Conclusion

This work highlights on how a knowledge graph can be seamlessly be part of an analysis and querying system. The web application provides a fundamental proof of principle for the users who are involved with exploring knowledge graphs. The web application allows the users write queries, select node and edges types and also allowing custom accession codes for deeper understanding of the clinical knowledge graph.

The aim of the study was to explore and understand how the clinical knowledge graph is structured, how can we extract meaning with different abstraction levels of the data and finally how to make the clinical knowledge graph more accessible. It enables users to write custom queries and select node and edges types for querying with out knowing a scripting language like Cypher, with the additional functionality of using custom accession codes to tailor results with specific proteins. This adaptability improves the accessibility of clinical knowledge graph.

The analysis currently facilitates three levels of data abstraction, each providing a deeper insight than the last. The first level offers distribution of nodes (i.e. highest and lowest node and relationship counts), the second level treats the graph as undirected focusing on node degrees and node type combinations, while the third level assumes a directed graph approach to compute node type in-degrees and out-degrees. Additionally, basic visualization techniques like tables, commutative distributive function and bar graphs have been employed to manage and interpret the extensive data. The use of logarithmic scaling and various binning methods has been crucial in representing the distribution of node and edge type distributions effectively. Through these visualizations, we identified that Gene, Transcript, Protein, Disease, Drug nodes are particularly significant within the graph.

6.1 Scope of Future Work

I advise exploring into the following research areas to enhance the EDA of knowledge graphs based on data from metaomics:

- **Conducting User Survey for Web Application**: A user study for evaluating the user-friendliness of the web application, which could improve the quality, usability and allow more users to engage .
- **Additions to current Frontend**: The addition of graph visualizations and the ability to return graph output along with tabular data. At present, the frontend has the limitation of retrieving only few hundred rows, in the future, pagination could be applied for massive result outputs.
- **Integration of different Knowledge Graphs**: Currently, we have only the Clinical Knowledge Graph 2.2.7 integrated, however, we have all the necessary tools to incorporate other knowledge graphs which also exist like Biomedical Knowledge Graphs [Nicholson and Greene, 2020] and Medical Knowledge Graphs [Shi et al., 2017] which can bring in more information in terms of different node types and relationships from different knowledge graphs.
- **Text to Query Language using Large Language Models**: Superseding from writing queries and selecting respective nodes to query, we can have a Text to Query model ingested e.g. LangChain [Topsakal and Akinci, 2023] and Neodash [Schäfer et al., 2022]. The user can write to the database without delving into the technical aspect of querying the knowledge graph.
- **Using dedicated Embedding Techniques**: It is very progressive to apply dedicated graph learning and embedding techniques. Extending from exploratory data analysis, abstractions and visualisations, it can provide clear insights in terms of graph learning tasks like link prediction (e.g., drug re-purposing), node classification and regression (e.g., finding missing properties of nodes. Exploring graph embedding strategies like Node2Vec 2.2.8.3 (for homogeneous graphs), Verse [Tsitsulin et al., 2018], and Metapath2Vec 2.2.8.4 (for heterogeneous graphs) to advance our understanding and applications of the clinical knowledge graph. These improvements and extensions can help significantly broaden the application's utility and effectiveness in clinical knowledge exploration. ULTRA is a method for generating transferable, uniform, and learnable graph representations. It makes use of the fundamental interactions and invariances present in the relationship graph. In order to derive relative relational representations, it uses conditional message forwarding. Using pre-trained ULTRA for heterogeneous graphs reason could be educational.
- **Plugins for Visualisations**: The enhancement of the application's capabilities by integrating advanced graph visualization libraries such as iGraph [Ognyanova, 2019], and Gephi [Jacomy et al., 2014] can increase the frontend's features. These features will further enrich the user experience and provide with detailed node bases visualizations.

Bibliography

Sami Abu-El-Haija, Bryan Perozzi, Rami Al-Rfou, and Alexander A Alemi. Watch your step: Learning node embeddings via graph attention. *Advances in neural information processing systems*, 31, 2018. (cited on Page 22)

Bilal Abu-Salih. Domain-specific knowledge graphs: A survey. *Journal of Network and Computer Applications*, 185:103076, 2021. (cited on Page 19)

Bruce Alberts. *Molecular biology of the cell*. Garland science, 2017. (cited on Page 7)

Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1): 81–100, 1993. (cited on Page 11)

Leigh Anderson and Jeff Seilhamer. A comparison of selected mrna and protein abundances in human liver. *Electrophoresis*, 18(3-4):533–537, 1997. (cited on Page 8)

Renzo Angles and Claudio Gutierrez. Survey of graph database models. *ACM Computing Surveys (CSUR)*, 40(1):1–39, 2008. (cited on Page ix and 12)

Dmitry Anikin, Oleg Borisenko, and Yaroslav Nedumov. Labeled property graphs: Sql or nosql? In *2019 Ivannikov Memorial Workshop (IVMEM)*, pages 7–13. IEEE, 2019. (cited on Page 13)

Euan A Ashley. Towards precision medicine. *Nature Reviews Genetics*, 17(9):507–522, 2016. (cited on Page 7)

Krystyna Balińska, Dragoš Cvetković, Zoran Radosavljević, S Simić, and Dragan Stevanović. A survey on integral graphs. *Publikacije Elektrotehničkog fakulteta. Serija Matematika*, pages 42–65, 2002. (cited on Page 11)

Greg Barnes and Uriel Feige. Short random walks on graphs. In *Proceedings of the Twenty-Fifth annual ACM symposium on Theory of computing*, pages 728–737, 1993. (cited on Page 22)

Niko Bernaola, Mario Michiels, Pedro Larrañaga, and Concha Bielza. Learning massive interpretable gene regulatory networks of the human brain by merging bayesian networks. *PLOS Computational Biology*, 19(12):e1011443, 2023. (cited on Page ix and 9)

Maciej Besta, Robert Gerstenberger, Emanuel Peter, Marc Fischer, Michał Podstawski, Claude Barthels, Gustavo Alonso, and Torsten Hoefler. Demystifying graph databases: Analysis and taxonomy of data organization, system designs, and graph queries. *ACM Computing Surveys*, 56(2):1–40, 2023. (cited on Page 12)

Gloria Bondel, Andre Landgraf, and Florian Matthes. Api management patterns for public, partner, and group web api initiatives with a focus on collaboration. In *26th European Conference on Pattern Languages of Programs*, pages 1–17, 2021. (cited on Page ix and 26)

P Buning and J Steger. Graphics and flow visualization in computational fluid dynamics. In *7th Computational Physics Conference*, page 1507, 1985. (cited on Page 10)

Philip Burnard. A method of analysing interview transcripts in qualitative research. *Nurse education today*, 11(6):461–466, 1991. (cited on Page 6)

Carlos D Bustamante, Francisco M De La Vega, and Esteban G Burchard. Genomics for the world. *Nature*, 475(7355):163–165, 2011. (cited on Page 1 and 8)

Surajit Chaudhuri and Umeshwar Dayal. An overview of data warehousing and olap technology. *ACM Sigmod record*, 26(1):65–74, 1997. (cited on Page 13)

Karim Khanmohammadi Chenab, Beheshteh Sohrabi, and Atyeh Rahmanzadeh. Superhydrophobicity: advanced biological and biomedical applications. *Biomaterials science*, 7(8):3110–3137, 2019. (cited on Page 8)

Frederic Commoner, Anatol W. Holt, Shimon Even, and Amir Pnueli. Marked directed graphs. *Journal of Computer and System Sciences*, 5(5):511–523, 1971. (cited on Page 11)

Randy Connolly. Facing backwards while stumbling forwards: The future of teaching web development. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 518–523, 2019. (cited on Page 25)

Allan Peter Davis, Cynthia J Grondin, Robin J Johnson, Daniela Sciaky, Jolene Wiegers, Thomas C Wiegers, and Carolyn J Mattingly. Comparative toxicogenomics database (ctd): update 2021. *Nucleic acids research*, 49(D1):D1138–D1143, 2021. (cited on Page 29)

Davide Di Pierro, Stefano Ferilli, and Domenico Redavid. Lpg-based knowledge graphs: A survey, a proposal and current trends. *Information*, 14(3), 2023. ISSN 2078-2489. doi: 10.3390/info14030154. URL https://www.mdpi.com/2078-2489/14/3/154. (cited on Page 14)

Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 135–144, 2017. (cited on Page ix, 23, and 24)

Eli Eisenberg and Erez Y Levanon. Human housekeeping genes, revisited. *TRENDS in Genetics*, 29(10):569–574, 2013. (cited on Page 6)

Yakov Fain and Anton Moiseev. *Angular 2 Development with TypeScript*, volume 239. Manning Publications Company, 2017. (cited on Page 27)

Oliver Fiehn. Metabolomics–the link between genotypes and phenotypes. *Plant molecular biology*, 48:155–171, 2002. (cited on Page 7)

Mikhail Galkin, Sören Auer, Maria-Esther Vidal, and Simon Scerri. Enterprise knowledge graphs: A semantic approach for knowledge management in the next generation of enterprise information systems. In *International Conference on Enterprise Information Systems*, volume 2, pages 88–98. SCITEPRESS, 2017. (cited on Page 19)

Jesse James Garrett et al. Ajax: A new approach to web applications. 2005. (cited on Page 24)

Viara Grantcharova, Eric J Alm, David Baker, and Arthur L Horwich. Mechanisms of protein folding. *Current opinion in structural biology*, 11(1):70–82, 2001. (cited on Page 6)

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016. (cited on Page 23)

Manish Kumar Gupta and Krishna Misra. A holistic approach for integration of biological systems and usage in drug discovery. *Network modeling analysis in health informatics and bioinformatics*, 5:1–12, 2016. (cited on Page 7)

Stavros Harizopoulos, Daniel J Abadi, Samuel Madden, and Michael Stonebraker. Oltp through the looking glass, and what we found there. In *Making Databases Work: the Pragmatic Wisdom of Michael Stonebraker*, pages 409–439. 2018. (cited on Page 13)

Xing He, Rui Zhang, Rubina Rizvi, Jake Vasilakes, Xi Yang, Yi Guo, Zhe He, Mattia Prosperi, and Jiang Bian. Prototyping an interactive visualization of dietary supplement knowledge graph. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1649–1652. IEEE, 2018. (cited on Page 30)

Xing He, Rui Zhang, Rubina Rizvi, Jake Vasilakes, Xi Yang, Yi Guo, Zhe He, Mattia Prosperi, Jinhai Huo, Jordan Alpert, et al. Aloha: developing an interactive graph-based visualization for dietary supplement knowledge graph through user-centered design. *BMC medical informatics and decision making*, 19:1–18, 2019. (cited on Page 30)

John R Hepler and Alfred G Gilman. G proteins. *Trends in biochemical sciences*, 17 (10):383–387, 1992. (cited on Page 6)

Robert Heyer, Kay Schallert, Annegret Büdel, Roman Zoun, Sophia Dorl, Alexander Behne, Fabian Kohrs, Sebastian Püttker, Christoph Siewert, Thilo Muth, Grit Saake, Udo Reichl, and Dirk Benndorf. A robust and universal metaproteomics workflow for research studies and routine diagnostics within 24 h using phenol extraction, fasp digest, and the metaproteomeanalyzer. *Frontiers in Microbiology*, 10:1883, 2019. doi: 10.3389/fmicb.2019.01883. (cited on Page 10)

Koki Horikoshi. Alkaliphiles: some applications of their products for biotechnology. *Microbiology and molecular biology reviews*, 63(4):735–750, 1999. (cited on Page 8)

Simon Huang. Load time optimization of javascript web applications, 2019. (cited on Page 27)

Zhisheng Huang, Jie Yang, Frank van Harmelen, and Qing Hu. Constructing knowledge graphs of depression. In *Health Information Science: 6th International Conference, HIS 2017, Moscow, Russia, October 7-9, 2017, Proceedings 6*, pages 149–161. Springer, 2017. (cited on Page 19)

James RA Hutchins. What's that gene (or protein)? online resources for exploring functions of genes, transcripts, and proteins. *Molecular Biology of the Cell*, 25(8): 1187–1201, 2014. (cited on Page 6)

Mathieu Jacomy, Tommaso Venturini, Sebastien Heymann, and Mathieu Bastian. Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software. *PloS one*, 9(6):e98679, 2014. (cited on Page 64)

Matthias Jarke and Jurgen Koch. Query optimization in database systems. *ACM Computing surveys (CsUR)*, 16(2):111–152, 1984. (cited on Page 12)

Weiwei Jiang. Graph-based deep learning for communication networks: A survey. *Computer Communications*, 185:40–54, 2022. (cited on Page 10)

Salim Jouili and Valentin Vansteenberghe. An empirical comparison of graph databases. In *2013 International Conference on Social Computing*, pages 708–715, 2013. doi: 10.1109/SocialCom.2013.106. (cited on Page 13)

Tomihisa Kamada, Satoru Kawai, et al. An algorithm for drawing general undirected graphs. *Information processing letters*, 31(1):7–15, 1989. (cited on Page 11)

Konrad J. Karczewski and Michael P. Snyder. Integrative omics for health and disease. *Nature Reviews Genetics*, 19:299–310, 2018. doi: 10.1038/nrg.2018.4. URL https://doi.org/10.1038/nrg.2018.4. (cited on Page 9)

Roland Kellner. Proteomics. concepts and perspectives. *Fresenius' journal of analytical chemistry*, 366:517–524, 2000. (cited on Page 1 and 8)

C. Königs, M. Friedrichs, and T. Dietrich. The heterogeneous pharmacological medical biochemical network pharmebinet. *Scientific Data*, 9(1):393, 2022. doi: 10.1038/s41597-022-01510-3. URL https://doi.org/10.1038/s41597-022-01510-3. (cited on Page 29)

Jan Krützfeldt and Markus Stoffel. Micrornas: a new class of regulatory genes affecting metabolism. *Cell metabolism*, 4(1):9–12, 2006. (cited on Page 6)

Melissa J Landrum, Jennifer M Lee, Mark Benson, Garth Brown, Chen Chao, Shanmuga Chitipiralla, Baoshan Gu, Jennifer Hart, Douglas Hoffman, Jeffrey Hoover, et al. Clinvar: public archive of interpretations of clinically relevant variants. *Nucleic acids research*, 44(D1):D862–D868, 2016. (cited on Page 29)

Juan Alfonso Lara, David Lizcano, María Aurora Martínez, and Juan Pazos. Developing front-end web 2.0 technologies to access services, content and things in the future internet. *Future Generation Computer Systems*, 29(5):1184–1195, 2013. (cited on Page 25)

Malgorzata Lazarska and Olga Siedlecka-Lamch. Comparative study of relational and graph databases. In *2019 IEEE 15th International Scientific Conference on Informatics*, pages 000363–000370. IEEE, 2019. (cited on Page xiii and 15)

Jinjiao Lin, Yanze Zhao, Weiyuan Huang, Chunfang Liu, and Haitao Pu. Domain knowledge graph-based research progress of knowledge representation. *Neural Computing and Applications*, 33:681–690, 2021. (cited on Page 19)

László Lovász. Random walks on graphs. *Combinatorics, Paul erdos is eighty*, 2 (1-46):4, 1993. (cited on Page 22)

Rohan Lowe, Neil Shirley, Mark Bleackley, Stephen Dolan, and Thomas Shafee. Transcriptomics technologies. *PLoS computational biology*, 13(5):e1005457, 2017. (cited on Page 1 and 8)

Tim A Majchrzak, Andreas Biørn-Hansen, and Tor-Morten Grønli. Progressive web apps: the definite approach to cross-platform development? 2018. (cited on Page 27)

Himel Mallick, Ali Rahnavard, Lauren J McIver, Siyuan Ma, Yancong Zhang, Long H Nguyen, Timothy L Tickle, George Weingart, Boyu Ren, Emma H Schwager, et al. Multivariable association discovery in population-scale meta-omics studies. *PLoS computational biology*, 17(11):e1009442, 2021. (cited on Page 9)

Tom Maniatis, Ross C Hardison, Elizabeth Lacy, Joyce Lauer, Catherine O'Connell, Diana Quon, Gek Kee Sim, and Argiris Efstratiadis. The isolation of structural genes from libraries of eucaryotic dna. *Cell*, 15(2):687–701, 1978. (cited on Page 6)

Pierre-Alain Maron, Lionel Ranjard, Christophe Mougel, and Philippe Lemanceau. Metaproteomics: a new approach for studying functional microbial ecology. *Microbial ecology*, 53:486–493, 2007. (cited on Page 10)

Dennis McCarthy and Umeshwar Dayal. The architecture of an active database management system. *ACM Sigmod Record*, 18(2):215–224, 1989. (cited on Page 12)

Justin J Miller. Graph database applications and concepts with neo4j. In *Proceedings of the southern association for information systems conference, Atlanta, GA, USA*, volume 2324, pages 141–147, 2013. (cited on Page 17)

Wilnellys Moore and Sarah Frye. Review of hipaa, part 2: limitations, rights, violations, and role for the imaging technologist. *Journal of nuclear medicine technology*, 48(1):17–23, 2020. (cited on Page 20)

David N Nicholson and Casey S Greene. Constructing knowledge graphs and their biomedical applications. *Computational and structural biotechnology journal*, 18: 1414–1428, 2020. (cited on Page 20 and 64)

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104 (1):11–33, 2015. (cited on Page 18)

Katherine Ognyanova. Network visualization with r. *Network*, 1:T2, 2019. (cited on Page 64)

Michael Olivier, Reto Asmis, Gregory A Hawkins, Timothy D Howard, and Laura A Cox. The need for multi-omics biomarker signatures in precision medicine. *International journal of molecular sciences*, 20(19):4781, 2019. (cited on Page 7)

Christopher Olston, Amit Manjhi, Charles Garrod, Anastassia Ailamaki, Bruce M Maggs, and Todd C Mowry. A scalability service for dynamic web applications. 2005. (cited on Page 27)

Jeff Z Pan. Resource description framework. In *Handbook on ontologies*, pages 71–90. Springer, 2009. (cited on Page 13)

Kim D Pruitt, Tatiana Tatusova, and Donna R Maglott. Ncbi reference sequences (refseq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic acids research*, 35(suppl_1):D61–D65, 2007. (cited on Page 5)

Sumit Purohit, Nhuy Van, and George Chin. Semantic property graph for scalable knowledge graph analytics. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 2672–2677, 2021. doi: 10.1109/BigData52589.2021.9671547. (cited on Page 14)

Ian Robinson, Jim Webber, and Emil Eifrem. *Graph databases: new opportunities for connected data.* " O'Reilly Media, Inc.", 2015. (cited on Page 1)

Ute Roessner and Jairus Bowne. What is metabolomics all about? *Biotechniques*, 46(5):363–365, 2009. (cited on Page 8)

Loïc Royer, Matthias Reimann, Bill Andreopoulos, and Michael Schroeder. Unraveling protein networks with power graph analysis. *PLoS computational biology*, 4(7): e1000108, 2008. (cited on Page 21)

Alberto Santos, Ana R Colaço, Annelaura B Nielsen, Lili Niu, Maximilian Strauss, Philipp E Geyer, Fabian Coscia, Nicolai J Wewer Albrechtsen, Filip Mundt, Lars Juhl Jensen, et al. A knowledge graph to interpret clinical proteomics data. *Nature biotechnology*, 40(5):692–702, 2022. (cited on Page 36)

Jero Schäfer, Ming Tang, Danny Luu, Anke Katharina Bergmann, and Lena Wiese. Graph4med: a web application and a graph database for visualizing and analyzing medical databases. *BMC bioinformatics*, 23(1):537, 2022. (cited on Page 64)

Maria V. Schneider and Sandra Orchard. *Omics Technologies, Data and Bioinformatics Principles*, pages 3–30. Humana Press, Totowa, NJ, 2011. ISBN 978-1-61779-027-0. doi: 10.1007/978-1-61779-027-0_1. URL https://doi.org/10.1007/978-1-61779-027-0_1. (cited on Page 5)

Emmit A Scott Jr. *SPA Design and Architecture: Understanding single-page web applications*. Simon and Schuster, 2015. (cited on Page 27)

Longxiang Shi, Shijian Li, Xiaoran Yang, Jiaheng Qi, Gang Pan, Binbin Zhou, et al. Semantic health knowledge graph: semantic integration of heterogeneous medical knowledge and services. *BioMed research international*, 2017, 2017. (cited on Page 64)

Seiji Tanaka and Harold A Scheraga. Medium-and long-range interaction parameters between amino acids for predicting three-dimensional structures of proteins. *Macromolecules*, 9(6):945–950, 1976. (cited on Page 6)

Robert Endre Tarjan. *Data structures and network algorithms*. SIAM, 1983. (cited on Page 10)

Oguzhan Topsakal and Tahir Cetin Akinci. Creating large language model applications utilizing langchain: A primer on developing llm apps fast. In *International Conference on Applied Engineering and Natural Sciences*, volume 1, pages 1050–1056, 2023. (cited on Page 64)

Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. Verse: Versatile graph embeddings from similarity measures. In *Proceedings of the 2018 world wide web conference*, pages 539–548, 2018. (cited on Page 64)

Chad Vicknair, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen, and Dawn Wilkins. A comparison of a graph database and a relational database: a data provenance perspective. In *Proceedings of the 48th annual Southeast regional conference*, pages 1–6, 2010. (cited on Page 14)

Daniel Walke, Daniel Micheel, Kay Schallert, Thilo Muth, David Broneske, Gunter Saake, and Robert Heyer. The importance of graph databases and graph learning for clinical applications. *Database*, 2023:baad045, 07 2023. ISSN 1758-0463. doi: 10.1093/database/baad045. URL https://doi.org/10.1093/database/baad045. (cited on Page 15)

Zhong Wang, Mark Gerstein, and Michael Snyder. Rna-seq: a revolutionary tool for transcriptomics. *Nature reviews genetics*, 10(1):57–63, 2009. (cited on Page 8)

Paul Warren, Cornelia Boldyreff, and Malcolm Munro. The evolution of websites. In *Proceedings Seventh International Workshop on Program Comprehension*, pages 178–185. IEEE, 1999. (cited on Page 26)

Gerhard Weikum and Hans-Jörg Schek. Concepts and applications of multilevel transactions and open nested transactions, 1992. (cited on Page 13)

David S Wishart, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, et al. Drugbank

5.0: a major update to the drugbank database for 2018. *Nucleic acids research*, 46 (D1):D1074–D1082, 2018. (cited on Page 29)

Nicholas C Wormald et al. Models of random regular graphs. *London mathematical society lecture note series*, pages 239–298, 1999. (cited on Page 11)

Zhilin Yang, Jie Tang, and William Cohen. Multi-modal bayesian embeddings for learning social knowledge graphs. *arXiv preprint arXiv:1508.00715*, 2015. (cited on Page 19)

Hongming Zhang, Daniel Khashabi, Yangqiu Song, and Dan Roth. Transomcs: From linguistic graphs to commonsense knowledge. *arXiv preprint arXiv:2005.00206*, 2020. (cited on Page 19)

Rongqing Zhang, Xiang Cheng, Liuqing Yang, and Bingli Jiao. Interference-aware graph based resource sharing for device-to-device communications underlaying cellular networks. In *2013 IEEE wireless communications and networking conference (WCNC)*, pages 140–145. IEEE, 2013. (cited on Page 10)

Rui Zhang, Rubina Rizvi, and Jake Vasilakes. integrated dietary supplement knowledge base (idisk). 2018. (cited on Page 30)

Wei Zhou, Li Li, Min Luo, and Wu Chou. Rest api design patterns for sdn northbound api. In *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, pages 358–365, 2014. doi: 10.1109/WAINA.2014.153. (cited on Page 26)

I herewith assure that I wrote the present thesis independently, that the thesis has not been partially or fully submitted as graded academic work and that I have used no other means than the ones indicated. I have indicated all parts of the work in which sources are used according to their wording or to their meaning.

Magdeburg, 8th May 2024