



# A framework for evaluating clinical data using graph-based link prediction

by

Himil Parshotam

Thesis presented in partial fulfilment of the requirements for the degree of  
**Master of Engineering (Industrial Engineering)**  
in the Faculty of Engineering at Stellenbosch University

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

Supervisor: Dr. GS Nel

March 2024



---

# Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: March 2024



---

# Abstract

Clinical data repositories, the most prevalent of which are *electronic health records*, typically comprise a broad range of information pertaining to various fields within the medical domain. Clinical data fields encapsulate distinct notions and are typically expressed in various formats — indicative of the specialised nature of the clinical domain. Data range from complex medical imaging data employed for diagnosis, therapy planning, intraoperative navigation, and post-operative monitoring, to clinical summaries derived from information collected during doctor-patient consultations, such as conditions, medications, and patient demographics. These data sources may be characterised by a notable degree of interconnectedness due to complex interactions and relationships that are embedded in respect of the various medical concepts.

Actionable insight may be derived from clinical data if abstracted and analysed by means of an appropriate modelling approach. One such approach is a so-called *knowledge graph* which represents an effective approach towards abstracting complex, interconnected data *via* a specialised data structure in which information is expressed by means of a mathematical construct known as *graphs*. Various algorithmic techniques may be applied to knowledge graphs in order to identify and leverage inferential relationships in a systematic manner — this may then form the basis of clinical decision support. *Link prediction* represents a popular approach towards inferring insight from graph-based data. The application of link prediction techniques from the realms of *network analysis* and *machine learning* warrants consideration in a clinical context due to their notable algorithmic utility in respect of extracting actionable insight from data — their utility can be further enhanced when applied to clinical data that are abstracted by means of knowledge graphs. There are, however, various complexities involved with constructing a knowledge graph and subsequently deriving insight therefrom.

In this thesis, a generic framework is proposed for constructing and analysing a knowledge graph derived from clinical data. The proposed framework conceptually represents a unified pipeline (or architecture) that may be employed towards transforming raw clinical data into an appropriate knowledge graph representation, and subsequently performing graph-based analysis, *i.e.* link prediction. The proposed framework comprises three functional components, each of which addresses an integral step in the overarching process. Two main types of use cases may be realised by means of the framework's implementation, the first of which relates to the prediction of medical conditions in order to identify misdiagnosed conditions, while the second use case pertains to the prediction (*i.e.* suggestion) of medication for prescription purposes.

Due to the challenges associated with access to real-world clinical data (attributable to privacy and confidentiality), reputable synthetic data are considered in order to demonstrate the methodological utility of the proposed framework. More specifically, two different data sets are considered, each of which differs in respect of the underlying clinical context and in terms of complexity. Three computerised instantiations are carried out, each of which varies in respect of different data sets that are subjected to the modelling pipeline and/or the clinical use case under

consideration. An algorithmic verification study is carried out prior to these instantiations in order to verify the functional correctness of the sixteen link prediction algorithms considered. The three main paradigms of link prediction algorithms considered are *common neighbour*-based algorithms, *machine learning* classifiers, and *graph neural networks*. Hyperparameter tuning is also carried out in order to identify suitable algorithmic configurations in respect of the different link prediction algorithms. During each computerised instantiation, algorithmic performance is evaluated both quantitatively (including statistical analyses) and qualitatively by means of appropriate visualisations and contextual reflections. The algorithmic output is also further contextualised in respect of practical insight pertaining to the aim of the respective use cases. Furthermore, the framework is validated by means of a subject matter expert, during which the methodological utility of the proposed framework and its applicability to real-world operations is corroborated.

---

# Opsomming

Kliniese databewaarplekke, waarvan *elektroniese gesondheidsrekords* die mees algemeen is, bevat tipies 'n wye reeks inligting wat betrekking het op verskeie aspekte binne die mediese veld. Datavelde omsluit verskillende begrippe en word tipies in verskeie formate uitgedruk — aanduidend van die gespesialiseerde aard van die kliniese veld. Data wissel van komplekse mediese beeldingdata wat gebruik word vir diagnose, terapiebeplanning, intraoperatiewe navigasie en post-operatiewe monitering, tot kliniese opsommings afgelei van inligting wat ingesamel is tydens dokter-pasiënt konsultasies, soos toestande, medikasie, en pasiëntdemografie. Hierdie databronne kan gekenmerk word deur 'n noemenswaardige mate van onderlinge verbondenheid as gevolg van komplekse interaksies en verwantskappe wat ingebed is ten opsigte van die verskillende mediese konsepte.

Praktiese uitvoerbare insig kan vanaf kliniese data afgelei word indien dit deur middel van 'n toepaslike modelleringsbenadering uitgedruk en ontleed word. Een so 'n benadering is 'n sogenaamde *kennisgrafiek* wat 'n effektiewe benadering tot die modellering van komplekse, onderling gekoppelde data verteenwoordig deur middel van 'n gespesialiseerde datastruktuur waarin inligting uitgedruk word deur middel van 'n wiskundige konsep bekend as *grafieke*. Verskeie algoritmiese tegnieke kan op kennisgrafieke toegepas word om inferensiële verhoudings op 'n sistematiese wyse te identifiseer en te benut — dit kan dan die basis vorm van kliniese besluitsteun. *Skakelvoorspelling* verteenwoordig 'n gewilde benadering tot die afleiding van insig uit grafiek-gebaseerde data. Die toepassing van skakelvoorspellingstegnieke uit die gebiede van *netwerkanalise* en *masjienleer* regverdig oorweging in 'n kliniese konteks as gevolg van hul noemenswaardige algoritmiese nut ten opsigte van die onttrekking van uitvoerbare insig uit data — hul nut kan verder verbeter word wanneer dit toegepas word op kliniese data wat deur middel van kennisgrafieke gemodelleer word. Daar is egter verskeie komplekse oorwegings betrokke tydens die opstel van 'n kennisgrafiek en die verkryging van insig daaruit.

In hierdie tesis word 'n generiese raamwerk voorgestel vir die ontwikkeling en ontleding van 'n kennisgrafiek wat afgelei is van kliniese data. Die voorgestelde raamwerk verteenwoordig konseptueel 'n verenigde pyplyn (of argitektuur) wat aangewend kan word om rou kliniese data in 'n toepaslike kennisgrafiekvoorstelling te transformeer, en daarna grafiekgebaseerde analise, *i.e.* skakelvoorspelling, uit te voer. Die voorgestelde raamwerk bestaan uit drie funksionele komponente, wat elk 'n integrale stap in die oorkoepelende proses aanspreek. Twee hoofstudies kan deur middel van die raamwerk se implementering gerealiseer word, waarvan die eerste verband hou met die voorspelling van mediese toestande ten einde verkeerd gediagnoseerde toestande te identifiseer, terwyl die tweede gevallestudie fokus op die voorspelling van medikasie vir voorskrifdoeleindes.

As gevolg van die uitdagings wat verband hou met toegang tot werklike kliniese data (toeskryfbaar aan privaatheid en vertroulikheid), word betroubare sintetiese data oorweeg om die metodologiese nut van die voorgestelde raamwerk te demonstreer. Meer spesifiek word twee verskillende

datastelle oorweeg, wat elkeen verskil ten opsigte van die onderliggende kliniese konteks en ten opsigte van kompleksiteit. Drie gerekenariseerde instansiasies word uitgevoer, wat elk verskil ten opsigte van verskillende datastelle wat aan die modelleringspyplyn en/of die kliniese gebruik geval onder oorweging onderwerp word. 'n Algoritmiese verifikasiestudie word voor hierdie instansiasies uitgevoer om die funksionele korrektheid van die sestien skakelvoorspellingsalgoritmes wat oorweeg is, te verifieer. Die drie hoofparadigmas van skakelvoorspellingsalgoritmes wat oorweeg word, is algemene buur-gebaseerde algoritmes, masjienleerklassifiseerders, en grafiekneurale netwerke.

Hiperparameterinstelling word ook uitgevoer om geskikte algoritmiese konfigurasies ten opsigte van die verskillende skakelvoorspellingsalgoritmes te identifiseer. Tydens elke gerekenariseerde instansiasie word algoritmiese prestasie beide kwantitatief (insluitend statistiese ontledings) en kwalitatief deur middel van gepaste visualiserings en kontekstuele refleksies geëvalueer. Die algoritmiese uitset word ook verder gekontekstualiseer ten opsigte van praktiese insig met betrekking tot die doel van die onderskeie gevallestudie. Verder word die raamwerk deur middel van 'n vakkundige gevalideer, waartydens die metodologiese bruikbaarheid van die voorgestelde raamwerk en die toepaslikheid daarvan op werklike bedrywighede gestaaf word.

---

# Acknowledgements

The author wishes to acknowledge the following people and institutions for their various contributions towards the completion of this work:

- My mother, father, and sister for their unwavering support, knowledge, and guidance throughout my academic career. Thank you for providing me with the opportunity to pursue my postgraduate studies.
- My supervisor, Dr Stephan Nel, for your constant support throughout the duration of this thesis. Thank you for your dedication to excellence and the valuable knowledge and research techniques that you have shared with me.
- The *Stellenbosch Unit for Operations Research and Engineering* (SUnORE) and the Department of Industrial Engineering, for the privilege to utilise their computational facilities and office space, and for promoting a research environment geared towards the pursuit of excellence.
- The NRF and SUnORE for their generous financial support.





---

# Table of Contents

<b>Abstract</b>	<b>iii</b>
<b>Opsomming</b>	<b>v</b>
<b>Acknowledgements</b>	<b>vii</b>
<b>List of Acronyms</b>	<b>xv</b>
<b>List of Figures</b>	<b>xvii</b>
<b>List of Tables</b>	<b>xxi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	1
1.2 Problem statement . . . . .	3
1.3 Objectives and scope . . . . .	3
1.4 Thesis organisation . . . . .	5
<b>I Literature review</b>	<b>7</b>
<b>2 Graph, statistical, and clinical prerequisites</b>	<b>9</b>
2.1 Graph preliminaries . . . . .	10
2.1.1 What is a graph? . . . . .	10
2.1.2 Graph types . . . . .	11
2.1.3 Representing graphs on computers . . . . .	13
2.1.4 Graph connectedness . . . . .	13
2.1.5 Graph databases . . . . .	15
2.1.6 Graph visualisations . . . . .	16
2.1.7 Important graph terminology . . . . .	17

2.2	Statistical testing preliminaries . . . . .	18
2.2.1	Inferential statistical testing . . . . .	18
2.2.2	Friedman test . . . . .	19
2.3	Clinical preliminaries . . . . .	19
2.3.1	Electronic health records . . . . .	19
2.3.2	Synthetic data generation . . . . .	20
2.3.3	Medical ontologies . . . . .	23
2.4	Clinical knowledge graphs . . . . .	24
2.5	Chapter summary . . . . .	28
<b>3</b>	<b>Link prediction</b>	<b>31</b>
3.1	Overview of link prediction . . . . .	32
3.1.1	Graph-based link prediction . . . . .	32
3.1.2	CRISP-DM . . . . .	33
3.1.3	Taxonomy of link prediction algorithmic approaches . . . . .	34
3.2	Common-neighbour based link prediction . . . . .	34
3.3	Path-based algorithms . . . . .	36
3.4	Classifier-based algorithms . . . . .	37
3.4.1	Machine learning paradigms . . . . .	37
3.4.2	Supervised learning link prediction . . . . .	39
3.5	Embedding-based link prediction . . . . .	42
3.5.1	Graph representation learning . . . . .	43
3.5.2	Graph embeddings . . . . .	44
3.5.3	Multi-relational data . . . . .	49
3.5.4	Graph neural networks . . . . .	51
3.6	Link prediction on bipartite graphs . . . . .	59
3.7	Link prediction on knowledge graphs . . . . .	60
3.8	Performance evaluation . . . . .	64
3.8.1	Evaluation metrics . . . . .	65
3.8.2	Graph partitioning . . . . .	67
3.9	Chapter summary . . . . .	70
<b>II</b>	<b>Framework</b>	<b>73</b>
<b>4</b>	<b>Framework discussion</b>	<b>75</b>
4.1	Framework development process . . . . .	76

Table of Contents	xi
4.2 Data flow diagrams . . . . .	77
4.3 A generic data science paradigm . . . . .	77
4.4 Quality assurance . . . . .	79
4.4.1 Verification . . . . .	79
4.4.2 Validation . . . . .	80
4.5 Related frameworks . . . . .	81
4.6 The MEDIKAL framework . . . . .	84
4.6.1 Database component . . . . .	85
4.6.2 The Processing component . . . . .	86
4.6.3 The KG construction component . . . . .	90
4.6.4 The Analysis component . . . . .	94
4.7 Design verification . . . . .	99
4.8 Chapter summary . . . . .	100
<b>5 Framework implementation</b>	<b>101</b>
5.1 Algorithmic verification . . . . .	102
5.2 Clinical data set background . . . . .	103
5.3 First framework instantiation . . . . .	104
5.3.1 Processing component . . . . .	105
5.3.2 KG construction component . . . . .	106
5.3.3 Analysis component . . . . .	108
5.4 Second framework instantiation . . . . .	121
5.4.1 Processing component . . . . .	122
5.4.2 KG construction component . . . . .	123
5.4.3 Analysis component . . . . .	123
5.5 Third framework instantiation . . . . .	128
5.5.1 Processing component . . . . .	129
5.5.2 KG construction component . . . . .	130
5.5.3 Analysis component . . . . .	130
5.6 Reflection . . . . .	135
5.7 Methodological utility of MEDIKAL framework . . . . .	136
5.8 Chapter summary . . . . .	137
<b>III Conclusion</b>	<b>139</b>
<b>6 Conclusion</b>	<b>141</b>

6.1 Thesis summary . . . . .	141
6.2 Appraisal of thesis contributions . . . . .	142
6.3 Suggestions for future work . . . . .	144
<b>References</b>	<b>147</b>
<b>A Numerical results</b>	<b>169</b>





---

# List of Acronyms

**AA:** Adamic Adar

**AI:** Artificial Intelligence

**ANOVA:** Analysis of Variance

**AUC:** Area Under a Curve

**AUPRC:** Area Under the Precision-Recall curve

**AUROC:** Area Under the Receiver Operating Characteristic Curve

**BERT:** Bidirectional Encoder Representations from Transformers

**BFS:** Breadth-First Search

**CDSS:** Clinical Decision Support Systems

**CKG:** Clinical Knowledge Graph

**CN:** Common Neighbours

**CRISP-DM:** Cross-Industry Standard Process for Data Mining

**DFD:** Data Flow Diagram

**DFS:** Depth-First Search

**DT:** Decision Tree

**EHR:** Electronic Health Record

**FN:** False Negative

**FP:** False Positive

**GAT:** Graph Attention Network

**GCN:** Graph Convolution Network

**GDL:** Geometric Deep Learning

**GNN:** Graph Neural Network

**GOF:** Goodness-Of-Fit

**GPCR:** G-Protein-Coupled Receptor



- GRL:** Graph Representation Learning
- GT:** Graph Transformer
- GUI:** Graphical User Interface
- i.i.d:** Independent and Identically Distributed
- JC:** Jaccard Coefficient
- KG:** Knowledge Graph
- kNN:**  $k$ -Nearest Neighbours
- LCL:** Local Community Links
- LR:** Logistic Regression
- LSTM:** Long Short-Term Memory
- MediKAL:** Medical Knowledge graph Analysis through Link prediction
- MEDDRA:** Medical Dictionary for Regulatory Activities Terminology
- ML:** Machine Learning
- MLP:** Multi-Layer Perceptron
- MR-GCN:** Multi-Relational Graph Convolutional Network
- NB:** Naïve Bayes
- NER:** Named Entity Recognition
- NLP:** Natural Language Processing
- NR:** Nuclear Receptor
- PA:** Preferential Attachment
- PALM:** Patient-centered Analytic Learning Machine
- PRA:** Path-based Resource Allocation
- PyG:** PyTorch Geometric
- SNOMED-CT:** Systematised Nomenclature of Medicine-Clinical Terms
- RA:** Resource Allocation
- RCGN:** Relational Graph Convolutional Network
- RF:** Random Forest
- ROC:** Receiver Operating Characteristic
- RS-EHR:** Realistic Synthetic Electronic Health Records
- TN:** True Negative
- TP:** True Positive
- UniMP:** Unified Message Passaging Model

---

# List of Figures

1.1	A visual representation of a KG . . . . .	2
2.1	A visual representation of a graph . . . . .	10
2.2	Examples of types of graphs . . . . .	11
2.3	A visual representation of a bipartite graph . . . . .	12
2.4	A visual representation of a complete bipartite graph . . . . .	12
2.5	A digraph and its underlying graph . . . . .	12
2.6	Computer representations of graphs . . . . .	13
2.7	An example of a walk . . . . .	14
2.8	A graph data model . . . . .	16
2.9	A force-based algorithm . . . . .	17
2.10	The PADARSER framework . . . . .	23
2.11	A clinical KG . . . . .	25
2.12	A taxonomy of clinical KGs . . . . .	26
3.1	The link prediction problem . . . . .	32
3.2	CRISP-DM life cycle . . . . .	33
3.3	Link prediction taxonomic overview . . . . .	35
3.4	Encoder-decoder approach . . . . .	45
3.5	Random walk procedure in node2vec . . . . .	49
3.6	The TransE model . . . . .	50
3.7	GNN pipeline . . . . .	52
3.8	Neural message passing . . . . .	54
3.9	Triadic closure . . . . .	59
3.10	Quadratic closure . . . . .	59
3.11	A depiction of $k$ -fold cross-validation . . . . .	64
3.12	A graphical illustration of the bias-variance trade-off . . . . .	65
3.13	Confusion matrix . . . . .	65

---

3.14	ROC curves . . . . .	67
3.15	Message-passing and supervision edges . . . . .	69
3.16	Inductive link prediction data split . . . . .	70
3.17	Transductive link prediction data split . . . . .	70
4.1	Framework development process . . . . .	76
4.2	The four symbols of a DFD . . . . .	77
4.3	Generic data science paradigm . . . . .	78
4.4	Methodology for system verification . . . . .	80
4.5	Pipeline for extracting medical concepts from unstructured text data . . . . .	81
4.6	Stroke KG framework construction . . . . .	82
4.7	CKG data model . . . . .	83
4.8	MEDIKAL framework . . . . .	85
4.9	Level-one DFD of the processing component . . . . .	87
4.10	Clinical data model . . . . .	87
4.11	Medical history presented in a .json format . . . . .	90
4.12	Level-two DFD of the Processing component . . . . .	91
4.13	Level-one DFD of the KG construction component . . . . .	92
4.14	Level-two DFD of the KG construction component . . . . .	93
4.15	Level-one DFD of the Analysis component . . . . .	95
4.16	Level-two DFD of the Analysis component . . . . .	96
5.1	Graph data model of first framework implementation . . . . .	105
5.2	KG pertaining to the 1K patient data set . . . . .	109
5.3	KG pertaining to the 10K patient data set . . . . .	110
5.4	Degree centrality of the condition vertices . . . . .	111
5.5	AUPRC box plots of the best performing link prediction algorithms . . . . .	116
5.6	Precision-recall curves of the first instantiation . . . . .	119
5.7	Confusion matrices of the first instantiation . . . . .	119
5.8	Historical versus predicted conditions (1K data set) . . . . .	120
5.9	Historical versus predicted conditions (10K data set) . . . . .	121
5.10	Data model of second framework implementation . . . . .	122
5.11	KG constructed in the second instantiation . . . . .	124
5.12	Degree centrality of condition and medication vertices . . . . .	125
5.13	Box plots of all algorithms (AUPRC second instantiation) . . . . .	126
5.14	Precision-recall curve and confusion matrix of the second instantiation . . . . .	128

---

5.15	Historical versus predicted medications (second instantiation) . . . . .	128
5.16	Data model of third framework implementation . . . . .	129
5.17	KG constructed in the third instantiation . . . . .	131
5.18	Degree centrality score of end vertices of ISA relation . . . . .	132
5.19	Box plots of all algorithms (AUPRC third instantiation) . . . . .	133
5.20	Precision-recall curve and confusion matrix of the third instantiation . . . . .	134
5.21	Historical versus predicted conditions (third instantiation) . . . . .	135
A.1	Box plots of all algorithms (AUROC 1K data set) . . . . .	174
A.2	Box plots of all algorithms (AUPRC 1K data set) . . . . .	175
A.3	Box plots of all algorithms (AUROC 10K data set) . . . . .	176
A.4	Box plots of all algorithms (AUPRC 10K data set) . . . . .	177
A.5	Box plots of all algorithms (AUROC second instantiation) . . . . .	177
A.6	Box plots of all algorithms (AUROC third instantiation) . . . . .	178



---

## List of Tables

2.1	Popular international medical terminology systems . . . . .	20
4.1	An example of patient data represented in a tabular <code>.csv</code> format . . . . .	88
4.2	An example of condition data represented in a tabular <code>.csv</code> format . . . . .	89
4.3	An example of HAS relationship data represented in a tabular <code>.csv</code> format . . . . .	89
4.4	An example of PRESCRIBED relationship data represented in a tabular <code>.csv</code> format . . . . .	89
4.5	Hyperparameters of link prediction approaches . . . . .	98
5.1	The structural properties pertaining to four benchmark data sets . . . . .	102
5.2	AUROC scores obtained by Aziz <i>et al.</i> . . . . .	103
5.3	AUROC scores obtained during the verification study . . . . .	103
5.4	An extract of the <code>conditions1K.csv</code> file . . . . .	105
5.5	An extract of the <code>has1K.csv</code> file . . . . .	106
5.6	An extract of the data contained within <code>condition_nodes10K.csv</code> . . . . .	107
5.7	An extract of the <code>has10k.csv</code> file . . . . .	108
5.8	A summary of the graph features (first instantiation) . . . . .	108
5.9	A list of the algorithms considered in Module 9.1 for the first instantiation . . . . .	112
5.10	Hyperparameter value ranges for link prediction algorithms . . . . .	113
5.11	A summary of the link prediction algorithm performance (1K data set) . . . . .	114
5.12	A summary of the link prediction algorithm performance (10K data set) . . . . .	115
5.13	The $p$ -values obtained in respect of each evaluation metric and data set . . . . .	117
5.14	The best performing link prediction algorithms (1K data set) . . . . .	117
5.15	The best performing link prediction algorithms (10K data set) . . . . .	118
5.16	An extract of the derived clinical insights . . . . .	121
5.17	An extract of the <code>medications.csv</code> file . . . . .	122
5.18	A summary of the graph features (second instantiation) . . . . .	123
5.19	A summary of the link prediction algorithm performance (second instantiation) . . . . .	125
5.20	Nemenyi test $p$ -values for AUROC values (second instantiation) . . . . .	126

---

5.21	Nemenyi test $p$ -values for AUPRC values (second instantiation) . . . . .	126
5.22	The best performing link prediction algorithms (second instantiation) . . . . .	127
5.23	An extract of the derived clinical insights (second instantiation) . . . . .	127
5.24	A summary of the graph features (third instantiation) . . . . .	130
5.25	A summary if the link prediction algorithm performance (third instantiation) . .	132
5.26	Nemenyi test $p$ -values for AUROC values (third instantiation) . . . . .	133
5.27	Nemenyi test $p$ -values for AUPRC values (third instantiation) . . . . .	133
5.28	The best performing link prediction algorithms (third instantiation) . . . . .	134
5.29	An extract of derived clinical insights (third instantiation) . . . . .	135
A.1	Hyperparameter tuning results for classifier-based algorithms (1K data set) . . .	169
A.1	Hyperparameter tuning results for classifier-based algorithms (1K data set) . . .	170
A.2	Hyperparameter tuning results for classifier-based algorithms (10K data set) . . .	170
A.2	Hyperparameter tuning results for classifier-based algorithms (10K data set) . . .	171
A.2	Hyperparameter tuning results for classifier-based algorithms (10K data set) . . .	172
A.3	Hyperparameter tuning results for the GNN architectures (1K data set) . . . . .	172
A.4	Hyperparameter tuning results for the GNN architectures (10K data set) . . . . .	173
A.5	Nemenyi procedure $p$ -values for AUROC values (1K data set) . . . . .	179
A.6	Nemenyi procedure $p$ -values for AUPRC values (1K data set) . . . . .	180
A.7	Nemenyi procedure $p$ -values for AUROC values (10K data set) . . . . .	181
A.8	Nemenyi procedure $p$ -values for AUPRC values (10K data set) . . . . .	182







---

---

## CHAPTER 1

---

# Introduction

### Contents

1.1	Background . . . . .	1
1.2	Problem statement . . . . .	3
1.3	Objectives and scope . . . . .	3
1.4	Thesis organisation . . . . .	5

## 1.1 Background

The scope of human healthcare is markedly vast and multifaceted, comprising a variety of factors such as genetic predispositions, environmental factors, and personal lifestyle choices, to name but a few [200]. Towards representing a patient’s clinical history effectively, clinical data repositories such as *electronic health records* (EHRs) ought to encapsulate a broad range of information pertaining to various facets within the medical domain. The various sub-domains within the realm of healthcare, however, comprise distinct data elements, formats, and abstractions which is indicative of the complex and specialised nature of the medical domain and its constituent fields. Accordingly, data may range from complex medical imaging data employed for diagnosis, therapy planning, intraoperative navigation, and post-operative monitoring, to clinical summaries (comprising structured and unstructured information) generated during doctor-patient consultations [225]. These data sources are typically characterised by a notable degree of interconnectedness due to complex interactions and relationships that are embedded in respect of the various medical concepts.

The notably complex and diverse nature of medical data renders it particularly important to integrate, abstract, analyse, and synthesise the various interconnected data sources by means of an appropriate data representation (*i.e.* model) so as to facilitate systematic and reliable decision support. Towards this end, so-called *knowledge graphs* (KGs) represent an effective approach, according to which complex, information-rich data can be expressed by means of a powerful mathematical construct, called *graphs* [239]. A graph may be defined simply as a collection of *vertices* that are joined by means of *edges* [86]. A vertex may be regarded as a representation of a real-world object which may, in turn, be connected to other real world objects — *i.e.* two vertices are *joined via* an edge denoting some contextual relationship [108]. KGs represent extensions of graphs in respect of their computational capacity to express at more detailed levels of abstraction the information (*i.e.* features) pertaining to both the real-world objects and their

relationships [120]. A visual representation of a simple graph comprising a collection of vertices (depicted by means of points) and edges (depicted by means of line segments) is presented in Figure 1.1.

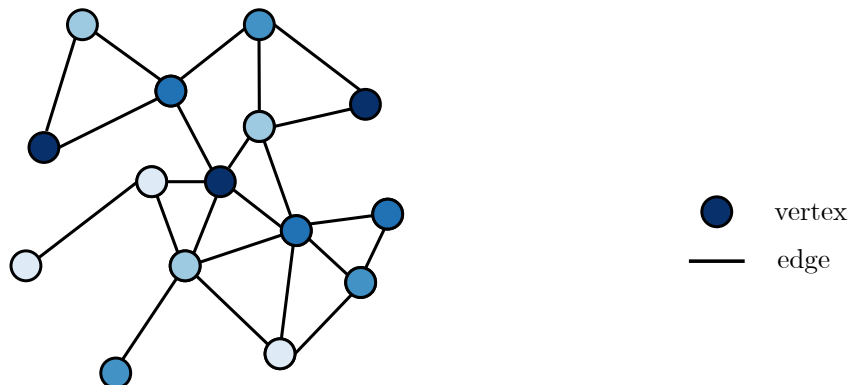


FIGURE 1.1: A visual representation of a graph comprising a collection of vertices and edges.

In a clinical context, the vertices constituting a KG may be employed towards abstracting entities such as patients, symptoms, conditions, and medications (including various features pertaining to each), while the edges may be employed towards expressing the interrelations between these entities (together with features pertaining to the respective relationships) [207]. The utility of KGs may be ascribed to their representational capacity, according to which complex relationships embedded within information sources may be modelled — such capacity notably exceeds those of conventional computational approaches [120]. Appropriately, the successful application of KGs in the healthcare domain has been reported widely [186, 207, 289, 307]. KGs derived from medical data may be further enriched by means of the integration of contextual information derived from dedicated *medical ontologies*<sup>1</sup>. Information gleaned from medical ontologies can supplement data-driven approaches towards inferring actionable insight from KGs due to the enrichment beyond the knowledge embedded within some sample (patient) data under consideration [259]. Consequently, improved decision support may be facilitated by means of ontology-supplemented KGs.

An appropriate analytical approach ought to be adopted in order to realise the analytical potential of KGs. One notable approach towards extracting inferential patterns from graphs is by means of *link prediction* — the aim of which is to predict the likelihood of a connection (*i.e.* edge) between two vertices in a graph based on *local* features pertaining to the vertices under consideration and *global* features pertaining to the graph as a whole [179]. In a clinical context, link prediction may be employed towards, for example, predicting (*i.e.* diagnosing) medical conditions, or suggesting (*i.e.* prescribing) medication. Traditional methods for conducting link prediction in graphs stem from simpler, heuristic-based approaches, such as *common neighbour* (CN)-based methods which are based on the principle that two vertices in a graph are more likely to have an edge joining them if they share a reasonably large number of adjacent vertices [168]. The domain of *machine learning* (ML) comprises a large number of powerful techniques that are amenable to the task of link prediction [108]. Such techniques are, however, susceptible to the intricacies involved with non-Euclidean data representations, such as graphs [39]. Fortunately, the nascent approach of *graph neural networks* (GNNs) [249] represents an effective approach towards harnessing the computational efficacy of ML-based techniques (more specifically, *deep learning*) by appropriately addressing non-Euclidean, graph-based data structures [108]. The

<sup>1</sup>A medical ontology may be defined as a structured vocabulary of medical-related data which comprise standardised definitions and descriptions of medical concepts and their relationships [149].

application of GNNs involve generating appropriate representations (called *embeddings*) of the considered graph which encapsulate information at various levels of abstraction, *i.e.* vertex- and edge-specific features, as well as *structural properties* of the graph (in respect of the nature according to vertices are connected within the graph) [44]. The advent of GNNs has been accompanied by promising developments towards analysing graph data structures, as showcased by the numerous advancements in the research community [38, 110, 147, 282].

KGs, together with algorithmic link prediction approaches, certainly represent notable utility in respect of clinical decision support. There are, however, various intricacies involved with the construction of KGs (based on clinical data) and the subsequent application of link prediction algorithms so as to derive decision support. Considerations range from graph data preprocessing, KG construction, medical ontology enrichment, algorithmic configuration as well as implementation, and the analysis of resulting outputs. A unified approach towards graph-based link prediction that encapsulates each of the aforementioned facets has not yet (to the best of the author's knowledge) been addressed in the literature. Focus is either placed exclusively on the construction of a KG [53, 164, 248] or exclusively on the analysis of pre-defined KGs [176, 223]. An important opportunity therefore manifests.

## 1.2 Problem statement

The aim in this thesis is to propose a data-driven framework — called the *Medical Knowledge graph Analysis through Link prediction* (MEDIKAL) framework — that can be employed towards processing clinical data in order to derive an appropriate KG abstraction. Furthermore, the proposed framework should facilitate the configuration and application of algorithmic techniques in order to extract actionable insights by means of graph-based link prediction in respect of different clinical use cases. The design and documentation of the framework ought to align with the fundamental facets that constitute link prediction analyses, *i.e.* ingesting, as input, raw clinical data, processing these data appropriately in order to construct a KG, after which various link prediction algorithms may be applied in order to derive decision support in respect of the resulting algorithmic output.

The proposed framework ought to comprise different functional components relating to the main tasks to be executed, such as data processing, KG construction, medical ontology enrichment, algorithmic configuration, implementation, and output analysis. The framework proffered in this thesis should also facilitate the execution of two prominent clinical use cases, the first of which relates to condition diagnosis (*i.e.* to predict misdiagnosed diseases), while the second use case pertains to medication prescription (*i.e.* to predict medication to prescribe). The framework is to be verified in respect of both its conceptual design (facilitated by means of detailed documentation accompanied by diagrammatic exposition) and algorithmic working (facilitated by means of reproducibility studies). Furthermore, the framework's utility in respect of its real-world applicability is to be validated by means of an appropriate subject matter expert.

## 1.3 Objectives and scope

The following seven objectives are pursued in this thesis:

I To *conduct* a thorough review of the literature related to:

(a) Appropriate graph, statistical, and clinical prerequisites,

- (b) Graph-based analytical techniques in respect of:
  - i. Link prediction fundamentals and its constituent algorithmic paradigms,
  - ii. link prediction in bipartite graphs and KGs, and
  - iii. performance evaluation of link prediction algorithmic approaches.
- II To *design*, based on the literature review of Objective I, a generic framework that may be employed towards constructing a medical KG, from which insight may be gleaned by means of different link prediction approaches. This framework should therefore facilitate:
  - (a) The preprocessing of raw patient-related data,
  - (b) the construction of a medical KG (possibly enriched by a medical ontology),
  - (c) the application and evaluation of different link prediction algorithmic approaches, and
  - (d) the analysis and contextualisation of algorithmic results in order to extract actionable insight.
- III To *verify* the computerised implementations of certain algorithmic approaches by means of the framework's (partial) instantiation in respect of benchmark data.
- IV To *implement* computerised instantiations of the designed framework in respect of the two considered use cases, *i.e.* diagnosis prediction and medication prescription.
- V To *conduct* an algorithmic performance analysis (including inferential statistical testing) of various link prediction algorithms so as to assess performance dynamics in respect of different hyperparameter values and different data considerations.
- VI To *validate* the methodological utility of the framework by means of a subject matter expert.
- VII To *recommend* sensible follow-up work related to the research carried out during the thesis which may be pursued in future.

In order to narrow down the scope of the problem considered in this thesis, the following delimitations are employed:

**Data:** Real-world patient data contain markedly sensitive personal and medical information. Consequently, access to such data sources is often accompanied by various privacy and ethical challenges. Towards addressing these challenges, synthetically generated patient data are employed in order to demonstrate the utility of the MEDIKAL framework. The data considered in this thesis are synthetic EHR data generated by means of *Synthea* [284], an open-source software library capable of generating high-quality, realistic, and historical-based clinical (patient) records. The medical ontological information employed in this study is derived from established ontologies, namely: The *systematised nomenclature of medicine-clinical terms* (SNOMED-CT) [31] as well as RXNORM [208].

**Computational environment:** The computational experiments conducted in this thesis are executed in respect of a 2.3GHz quad-core Intel i5 processor together with 8GB of RAM. The macOS Ventura 13.4 operating system was employed.

**Graph-based analysis:** Various graph-based analytical tasks have been proposed in the literature, namely: Node classification, link prediction, graph classification, and clustering [108]. Link prediction is selected as the principal analytical approach adopted in this thesis as

it represents an intuitive approach towards inferring clinical decision support. Furthermore, studies in the literature do not address the unified aim of the proposed MEDIKAL framework — therefore an opportunity arises in respect of the task of link prediction. Three of the most popular link prediction paradigms (together with the most prevalent algorithms therein) are considered, namely: CN-based algorithms, supervised ML classifiers, and GNNs (within the embedding-based paradigm) [298]. *Matrix factorisation* and *path-based techniques* are omitted due to their comparatively limited popularity in the literature.

## 1.4 Thesis organisation

In addition to this introductory chapter, this thesis comprises an additional five chapters (together with a bibliography and an appendix), partitioned into three parts. Part I comprises two chapters and is dedicated to a thorough review of relevant preliminaries and the literature pertaining to relevant topics addressed in this thesis. In Chapter 2, fundamental graph, statistical, and clinical prerequisites that are relevant to the work conducted in this thesis are presented. First, an overview of graph preliminaries is presented which encapsulates technical background in respect of the underlying mathematical principles of graphs and certain elementary notions. Thereafter, statistical testing preliminaries are addressed which includes a brief discussion on inferential statistical testing and a specific set of *non-parametric* tests. Lastly, an overview of relevant clinical prerequisites is presented which pertains to important contextual information within the clinical domain. A discussion on EHR data is then presented which is followed by a discourse pertaining to synthetic clinical data generation as well as medical ontology systems. The notion of clinical KGs is subsequently delineated which includes a discussion on relevant work within the KG literature. A discussion on formative topics pertaining to ML within a clinical context are also discussed.

The aim in Chapter 3 is to present detailed discussions on the fundamental concepts pertaining to the field of link prediction. The chapter opens with an introduction to the contextualisation of link prediction in a graph-based context which is followed by an elucidation of various link prediction algorithmic approaches, namely: CN, classifier-, and embedding-based algorithms. The focus then shifts to a discussion on link prediction in respect of bipartite graphs and KGs, during which the intricate challenges associated with applying link prediction algorithms to complex graph structures are discussed. Performance evaluation is then addressed which includes a discussion on important evaluation metrics and graph data partitioning considerations.

Part II of this thesis pertains to the proposed framework and comprises two chapters, *i.e.* Chapters 4 and 5. In Chapter 4, the notion of a framework and its developmental process are first described. A discussion pertaining to *data flow diagrams* (DFDs) is then presented, followed by a discourse on the generic data science paradigm which represents the methodological foundation upon which the proposed framework is constructed. Related work within the literature is then presented in respect of similar (albeit slightly tangential) frameworks. Finally, a high-level overview of the proposed framework is presented, followed by a detailed documentation of the MEDIKAL framework's design in respect of its main functional components and constituent modules.

In Chapter 5, three distinct computerised instantiations of the proposed framework are addressed during which the functional working and utility of the framework are demonstrated. The chapter opens with a discourse pertaining to computational verification in order to establish the functional correctness of the algorithmic approaches employed. Background information pertaining

to the main clinical (synthetic) data sets under consideration is then presented. Thereafter, the framework's implementation is discussed in detail with respect to each component and its constituent modules. Three different computerised instantiations are carried out, each of which differs in respect of the data and/or the clinical use case considered. A discourse pertaining to the framework's validation by a subject matter expert is then presented.

The final part of this thesis, *i.e.* Part III, comprises Chapter 6, the bibliography, and an appendix. Chapter 6 contains a summary of the content covered in this thesis as well as a reflection on its contributions. Furthermore, a number of suggestions are discussed with respect to possible follow-up work that may be conducted in order to extend upon the aforementioned contributions. The bibliography then follows. Finally, in the appendix, additional numerical results are presented by means of various visualisations.

**Part I**

**Literature review**





---

---

## CHAPTER 2

---

# Graph, statistical, and clinical prerequisites

### Contents

2.1	Graph preliminaries . . . . .	10
2.1.1	<i>What is a graph?</i> . . . . .	10
2.1.2	<i>Graph types</i> . . . . .	11
2.1.3	<i>Representing graphs on computers</i> . . . . .	13
2.1.4	<i>Graph connectedness</i> . . . . .	13
2.1.5	<i>Graph databases</i> . . . . .	15
2.1.6	<i>Graph visualisations</i> . . . . .	16
2.1.7	<i>Important graph terminology</i> . . . . .	17
2.2	Statistical testing preliminaries . . . . .	18
2.2.1	<i>Inferential statistical testing</i> . . . . .	18
2.2.2	<i>Friedman test</i> . . . . .	19
2.3	Clinical preliminaries . . . . .	19
2.3.1	<i>Electronic health records</i> . . . . .	19
2.3.2	<i>Synthetic data generation</i> . . . . .	20
2.3.3	<i>Medical ontologies</i> . . . . .	23
2.4	Clinical knowledge graphs . . . . .	24
2.5	Chapter summary . . . . .	28

The second chapter in this thesis is dedicated to providing the reader with an understanding of the graph, statistical, and clinical preliminaries that are foundational to this thesis. First, an overview of graph preliminaries is presented which provides technical insight into the underlying mathematical principles of various topics pertaining to graphs, such as graph types, graph representations, graph connectedness, and graph visualisations. The notions of graph databases and multi-relational graphs are also elucidated. Thereafter, statistical testing preliminaries are presented in order to expound on its application with respect to the analysis conducted in this thesis. Lastly, an overview of relevant clinical prerequisites is provided in order to equip the reader with sufficient domain knowledge. The chapter concludes with a brief summary of its contents.

## 2.1 Graph preliminaries

Graph theory, a prominent sub-discipline of mathematics, is currently an active field of research, whilst its application-orientated counterpart, *i.e.* network analysis, is also afforded interest, especially so in the prior two decades (at the time of writing). The increased popularity of graph-based approaches may largely be attributed to its applicability to many important use cases in key scientific domains, *e.g.* physics, chemistry, psychology, and sociology. Furthermore, certain problems within the realm of theoretical computer science may also be addressed by means of graph-theoretic formulations and their subsequent analyses [19, 68]. The representational capacity of graphs as mathematical constructs is therefore of significant value, as showcased by its rich history together with nascent advances. The aim in the following sections is to present a background on the foundational concepts pertaining to graphs, during which important terminology and mathematical notation are elucidated.

### 2.1.1 What is a graph?

A graph  $\mathcal{G}$  represents a finite (non-empty) set of objects called vertices, denoted by  $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$  and a (possibly empty) set of objects called edges, denoted by  $\mathcal{E}$  [279]. The cardinality of the vertex set  $\mathcal{V}$  is called the *order* of  $\mathcal{G}$  and is denoted by  $|\mathcal{V}| = n$ . Each edge is associated with an unordered pair of vertices, denoted by  $\{v_i, v_j\}$ , which are called the *end vertices* of this edge — an edge therefore joins two vertices, denoted by  $e_k = \{v_i, v_j\}$ , and the vertices  $v_i$  and  $v_j$  are *incident* to edge  $e_k$ . Furthermore, the cardinality of the edge set  $\mathcal{E} = \{e_1, e_2, \dots, e_m\}$  is called the *size* of  $\mathcal{G}$  and is denoted by  $|\mathcal{E}| = m$ . Two distinct vertices sharing the same edge are said to be *adjacent*; similarly, two edges sharing the same vertex are also adjacent. Furthermore, two distinct vertices  $v_i$  and  $v_j$  are neighbours if  $\{v_i, v_j\} \in \mathcal{E}$  [19]. The set of adjacent vertices of a vertex  $v_i$  in graph  $\mathcal{G}$  is referred to as the *open neighbourhood* of  $v_i$  and is denoted by  $\mathcal{N}_{\mathcal{G}}(v_i)$ . The *closed neighbourhood* of  $v_i$  corresponds to the set of adjacent vertices of  $v_i$  as well as the vertex  $v_i$ , denoted by  $\mathcal{N}_{\mathcal{G}}[v_i] = \mathcal{N}_{\mathcal{G}}(v_i) \cup \{v_i\}$ . Each subsequent reference to the notion of a neighbourhood refers to an open neighbourhood unless stated otherwise. Moreover, a graph  $\mathcal{H}$  is a *subgraph* of graph  $\mathcal{G}$  if  $\mathcal{V}(\mathcal{H}) \subseteq \mathcal{V}(\mathcal{G})$  and  $\mathcal{E}(\mathcal{H}) \subseteq \mathcal{E}(\mathcal{G})$  [68].

Graphs are typically depicted as a collection of points, representing vertices, joined together by line segments, representing edges. An example of such a representation may be observed in Figure 2.1. An edge with vertex pair  $\{v_i, v_i\}$ , *i.e.* the same vertex at both ends of the edge, is called a (*self*)-*loop*. Moreover, a pair of vertices  $\{v_i, v_j\}$  may be joined by more than one edge. Such edges are called *parallel edges* [68]. In Figure 2.1, edge  $e_6$  is a loop, whereas edges  $e_4$  and  $e_5$  are parallel edges. A graph comprising loops and/or parallel edges is referred to as a *pseudograph*. If a pseudograph only contains parallel edges, it is called a *multigraph*. A *simple graph* is defined as a graph containing neither self-loops nor parallel edges [68]. Due to the scope of this project, only simple graphs are considered.

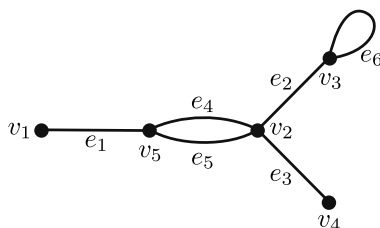


FIGURE 2.1: A visual representation of a graph comprising a set of vertices and edges, as depicted by points and line segments, respectively [19].

The number of edges incident on vertex  $v_i$  is referred to as the *degree* of the vertex  $v_i$  and is denoted by  $d(v_i)$  [68]. A graph is called *r-regular* if the degree of each vertex in the graph equates to  $r$  [279]. In Figure 2.1,  $d(v_2) = 4$ . Due to each edge being *incident* with two vertices, *i.e.* each edge contributes two degrees, the sum of the degrees in respect of all vertices in a graph equates to twice the number of its edges. The average degree of a vertex set is denoted by  $\bar{d}(\mathcal{V})$ .

### 2.1.2 Graph types

Graphs can assume a variety of forms, each possessing distinct properties. The simplest manifestation of a graph is a so-called *trivial graph* which is of order  $n = 1$ , as seen in Figure 2.2(a). A *complete graph* of order  $n$ , denoted by  $\mathcal{K}_n$ , is a graph in which each pair of distinct vertices is connected by an edge. A complete graph of order  $n = 5$  is presented in Figure 2.2(b). A graph comprising an empty edge set, *i.e.*  $\mathcal{E} = \emptyset$ , is referred to as an *empty graph* and is denoted by  $\bar{\mathcal{K}}_n$ . In Figure 2.2(c), an example of an empty graph is depicted. It is also possible for the edges within a graph to contain direction.

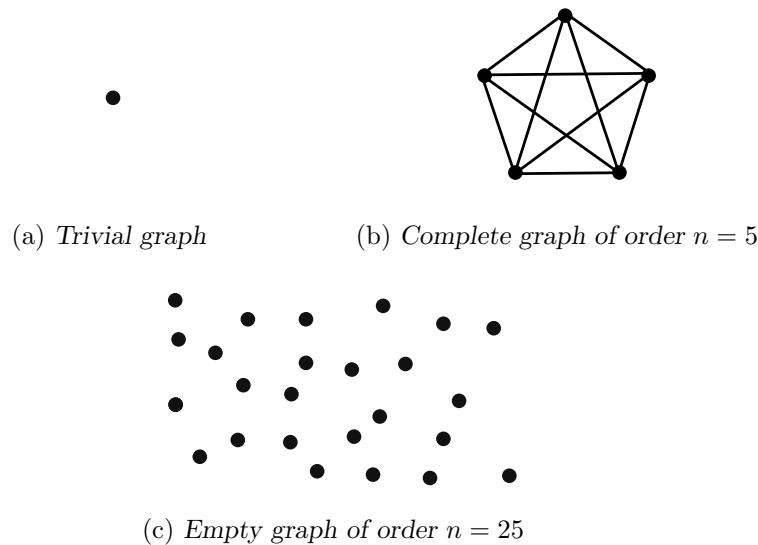


FIGURE 2.2: Examples of a (a) trivial, (b) complete, and (c) empty graph.

A *bipartite graph* is a graph whose vertex set  $\mathcal{V}$  can be partitioned into two *partite sets*  $\mathcal{V}_1$  and  $\mathcal{V}_2$ , according to which each edge within the graph joins a vertex of  $\mathcal{V}_1$  to a vertex of  $\mathcal{V}_2$ , as seen in Figure 2.3. There is therefore no edge between vertices in the same partite set and therefore the maximum possible degree of a vertex is the number of vertices in the opposing vertex set [34, 279]. The degree centrality of a vertex is a measure of the number of edges incident to a vertex. Accordingly, in the case of a bipartite graph, the degree centrality of a vertex  $v_i$  within  $\mathcal{V}_1$ , denoted by  $d_c(v_i)$ , may be expressed as

$$d_c(v_i) = \frac{d(v_i)}{|\mathcal{V}_2|}, \text{ for } v_i \in \mathcal{V}_1$$

and similarly for a vertex  $v_j$  in  $\mathcal{V}_2$ , that is

$$d_c(v_j) = \frac{d(v_j)}{|\mathcal{V}_1|}, \text{ for } v_j \in \mathcal{V}_2.$$

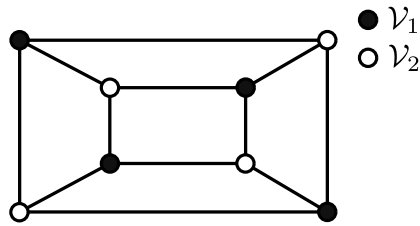


FIGURE 2.3: A visual representation of a bipartite graph comprising partite sets  $\mathcal{V}_1$  and  $\mathcal{V}_2$  [279].

A bipartite graph in which each vertex of  $\mathcal{V}_1$  is adjacent to each vertex of  $\mathcal{V}_2$  is called a *complete bipartite graph*. If  $|\mathcal{V}_1| = r$  and  $|\mathcal{V}_2| = s$ , then a complete bipartite graph may be denoted by  $\mathcal{K}_{r,s}$ . Moreover, a complete bipartite graph of the form  $\mathcal{K}_{n,n}$  is referred to as an *n-biclique*. In Figure 2.4, a  $\mathcal{K}_{3,3}$  complete bipartite graph (also referred to as a 3-biclique) is presented [279].

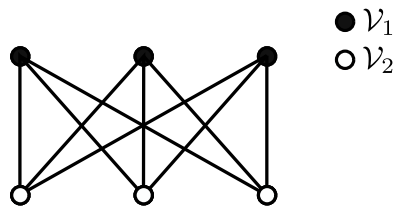


FIGURE 2.4: A visual representation of a complete bipartite graph  $\mathcal{K}_{3,3}$ , also called a 3-biclique [279].

Another important classification of graphs is a so-called directed graph (or *digraph*), denoted by  $\mathcal{D}$ , which comprises a finite nonempty set of vertices and a set of *ordered* pairs of distinct vertices, called *arcs* or *directed edges*. The vertex set of  $\mathcal{D}$  is denoted by  $\mathcal{V}(\mathcal{D})$ , while the arc set of  $\mathcal{D}$  is denoted by  $\mathcal{E}(\mathcal{D})$ , such that  $\mathcal{D} = (\mathcal{V}, \mathcal{E})$ . Digraphs may be represented diagrammatically in a similar manner to Figure 2.1 but with the addition of arrowheads indicating directionality. For example, a line segment with an arrowhead from vertex  $v_1$  to vertex  $v_2$  corresponds to the arc  $(v_1, v_2)$ . An example of a digraph may be observed in Figure 2.5(a) which is contrasted by its undirected counterpart in Figure 2.5(b).

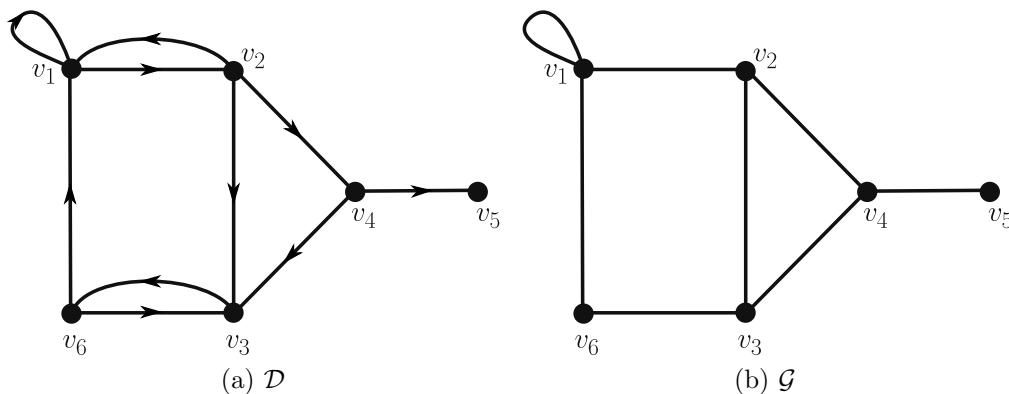


FIGURE 2.5: A digraph  $\mathcal{D}$  (left) and its undirected counterpart  $\mathcal{G}$  (right)[19].

### 2.1.3 Representing graphs on computers

An effective approach towards representing graphs on a computer is a so-called adjacency matrix. The *adjacency matrix* of a graph  $\mathcal{G}$  of order  $n$ , is an  $n \times n$  binary matrix denoted by  $\mathbf{A}(\mathcal{G})$  whose  $(i, j)$ -th element equates to one if  $\{v_i, v_j\} \in \mathcal{E}(\mathcal{G})$  i.e.  $\mathbf{A}(\mathcal{G})_{v_i, v_j} = 1$ , or zero otherwise, for  $i, j \in \{1, \dots, n\}$ . Consider the example graph  $\mathcal{G}$  graphically illustrated in Figure 2.6(a). The corresponding adjacency matrix of  $\mathcal{G}$  is shown in Figure 2.6(b). In the discussions hereafter, the adjacency matrix is simply denoted by  $\mathbf{A}$  (unless the context is unclear). An alternative approach towards representing graphs computationally is a so-called *edge list* in which each edge within a graph is represented by its associated vertex pair, i.e.  $\{v_i, v_j\}$ . The edge list of a graph may therefore be represented as a column vector containing all adjacent vertex pairs [8]. For example, the edge list corresponding to the graph in Figure 2.6(a) is presented in Figure 2.6(c).

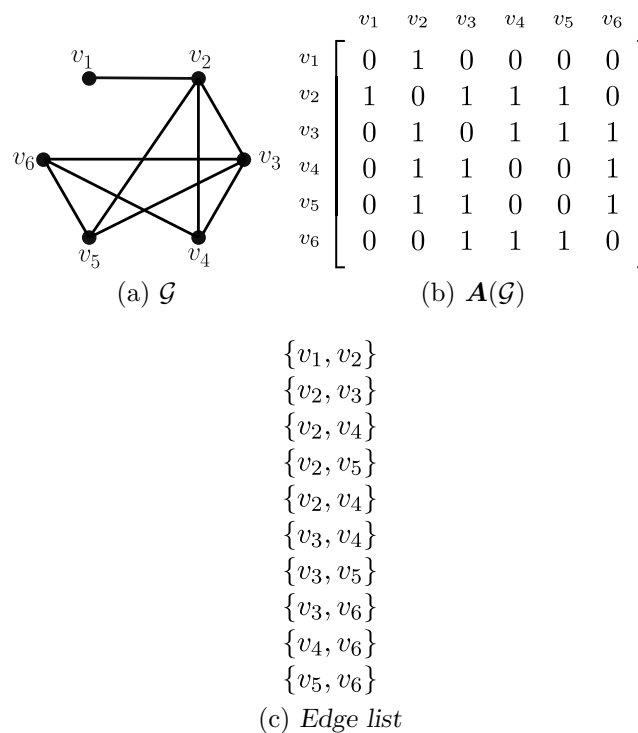


FIGURE 2.6: A graphical illustration of the different approaches towards representing the graph in (a), i.e. an adjacency matrix (b) and an edge list (c) [279].

### 2.1.4 Graph connectedness

An important consideration in graphs relate to the extent to which vertices are (directly or indirectly) joined by means of interconnecting edges [279]. Practical insight may be inferred based on the underlying graph's so-called connectedness. For example, a graph comprising many interconnecting edges typically corresponds to improved levels of redundancy (within some network) but also increased cost [279]. Foundational concepts pertaining to graph connectedness are introduced hereafter.

### Connected graphs

A  $v_i$ - $v_j$  walk in a graph is defined as a finite, alternating sequence of vertices and edges that starts at the vertex  $v_i$  and ends at the vertex  $v_j$  such that each edge is incident with the vertices preceding and following it within the sequence [279]. It should be noted that a walk can start and end at the same vertex. Furthermore, it is not necessary for all edges and vertices in a walk to be distinct. The number of edges in a walk is called the *length* of the walk [296]. For example, consider the graph  $\mathcal{G}$  in Figure 2.7. A  $v_3$ - $v_4$  walk can be represented by the sequence

$$v_3, v_3v_2, v_2, v_2v_6, v_6, v_6v_3, v_3, v_3v_4, v_4, v_4v_5, v_5, v_5v_4, v_4.$$

This walk may be expressed in a more compact form (in the case of simple graphs) by omitting the edges and commas, yielding  $v_3 v_2 v_6 v_3 v_4 v_5 v_4$  [279].

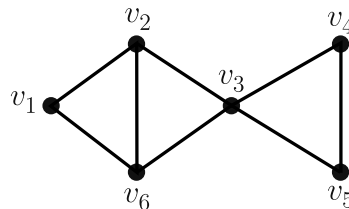


FIGURE 2.7: A simple graph  $\mathcal{G}$  used to express the notion of a walk [279].

A walk containing only distinct edges is referred to as a *trail*. If, in addition, all the vertices within the sequence are also distinct, then the trail is called a *path*. The aforementioned walk  $v_3$ - $v_4$  is not a trail as the edge  $\{v_4, v_5\}$  occurs twice in the sequence. The sequence  $v_3 v_2 v_6 v_3 v_4$  is a  $v_3$ - $v_4$  trail in the graph  $\mathcal{G}$ , whereas the trail  $v_3 v_5 v_4$  is a  $v_3$ - $v_4$  path in graph  $\mathcal{G}$  [279]. A walk is referred to as *closed* if it starts and ends at the same vertex, otherwise it is referred to as *open*. Two notable types of paths in graph analysis are the *Eulerian path* and the *Hamiltonian path*. In an Eulerian path, each edge is visited exactly once, while in the case of a Hamiltonian path, each vertex is visited exactly once. A path can be both Eulerian and Hamiltonian [121]. Furthermore, a closed walk in which all the edges are different is called a *closed trail*. A *circuit* is a closed trail containing at least three vertices. Moreover, a circuit in which no vertices, other than the first and last, are repeated is called a *cycle*. Based on the aforementioned concepts and terminology, a graph  $\mathcal{G}$  is *connected* if there exists a path in  $\mathcal{G}$  between any two of its vertices, otherwise it is *disconnected*. Disconnected graphs may be partitioned into a number of connected subgraphs called *components*, where the number of components of  $\mathcal{G}$  is denoted by  $k(\mathcal{G})$  [68].

For a connected graph  $\mathcal{G}$ , the *distance* between two vertices  $v_i$  and  $v_j$ , denoted by  $d_{\mathcal{G}}(v_i, v_j)$ , is defined as the minimum of the lengths of the  $v_i$ - $v_j$  paths of  $\mathcal{G}$ . If  $\mathcal{G}$  is a disconnected graph, then the distance between two vertices  $v_i$  and  $v_j$  belonging to the same component of  $\mathcal{G}$  is defined similarly. If the vertices  $v_i$  and  $v_j$  do not belong to the same component of  $\mathcal{G}$ , then  $d_{\mathcal{G}}(v_i, v_j)$  is undefined which may be expressed as  $d_{\mathcal{G}}(v_i, v_j) = \infty$  [279]. Distance may be further contextualised by means of so-called edge weights which are applicable to weighted<sup>1</sup> graphs. Prominent path finding algorithms include Dijkstra's algorithm [71] and the Bellman-Ford algorithm [26].

<sup>1</sup>A weighted graph may be regarded as a generalisation — *i.e.* if weights are omitted, each edge may be assumed to have a weight of one.

### 2.1.5 Graph databases

A graph database management system, also known as a graph database, diverges from traditional relational database management systems which utilise data tables. Instead, it employs a graph data model (*i.e.* schema<sup>2</sup>) to store data in the form of vertices and edges, alternatively referred to as nodes and relationships (or links). A graph data model may be defined as a high-level graph-based abstraction of the considered problem which defines the type of vertices and edges that constitute the problem context [239]. Graph databases are typically constructed in the context of *online transactional processing* systems which employ *create, read, update, and delete* methodologies to execute transactions in respect of the graph database, typically in real-time [239]. A graph database comprises a single data structure, unlike relational database management systems which store data in heterogeneous tables. Due to each vertex (or edge) containing a direct reference to its adjacent vertex (or edge), there is no need for further “join” operations<sup>3</sup>, as the data structure is already “joined” by the edges that define it. A significant advantage that arises from this notion is that there is a constant compute time requirement involved when retrieving an adjacent vertex or edge — *i.e.* the time required to carry out a local read operation at a vertex or edge remains constant irrespective of the size of the graph, rendering graph databases markedly efficient for analyses [241].

### Multi-relational graphs

The field of graphs (or networks) is largely concerned with the development of computational approaches pertaining to *single-relational* graphs, in which all edges are homogenous in nature. Real-world systems are, however, often more complex, comprising multiple relationship types and therefore require a more appropriate representation. One such representation is a so-called *multi-relational property graph* which may be defined as a directed, edge-labelled, and attributed multi-graph<sup>4</sup> [241]. Towards representing these heterogeneous relations, the current edge notation is extended to include an edge type, denoted by  $\tau \in \mathcal{R}$ , where  $\mathcal{R}$  denotes the set of relations such that  $\{v_i, \tau, v_j\} \in \mathcal{E}$  which corresponds to an edge of type  $\tau$  between vertex  $v_i$  and vertex  $v_j$ . Furthermore, an adjacency matrix is defined for each edge type  $\tau$ , denoted by  $\mathbf{A}_\tau$ . A multi-relational graph may be summarised by the adjacency tensor<sup>5</sup>  $\mathcal{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{R}| \times |\mathcal{V}|}$  [108]. The vertices of a graph may also be heterogeneous in nature. More specifically, the vertices are partitioned into multiple disjoint sets, that is  $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \dots \cup \mathcal{V}_k$ , where  $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset$  for all  $i \neq j$ , based on inherent characteristic differences [108]. A KG, on the other hand, is a directed graph in which the vertices represent real-world entities and the edges specifically represent *subject-property-object* triplets, expressed as  $\{\text{head entity}, \text{relation}, \text{tail entity}\}$ , *i.e.* there is a relation from the *head* entity to the *tail* entity. A KG is therefore regarded as an instance of a heterogeneous graph [40]. An example graph data model of a KG representing the entities and relations within a film context is presented in Figure 2.8.

<sup>2</sup>A schema can be defined as the conceptual architecture which details the type of entities within a data store and the manner in which they are connected to one another [23].

<sup>3</sup>In the context of conventional relational databases, join operations relate to combining rows from two or more tables based on a related column between them, thereby connecting data instances that reside in separate tables.

<sup>4</sup>A graph containing parallel edges and/or self-loops.

<sup>5</sup>A tensor is a mathematical object that generalises the concepts of scalars, vectors, and matrices



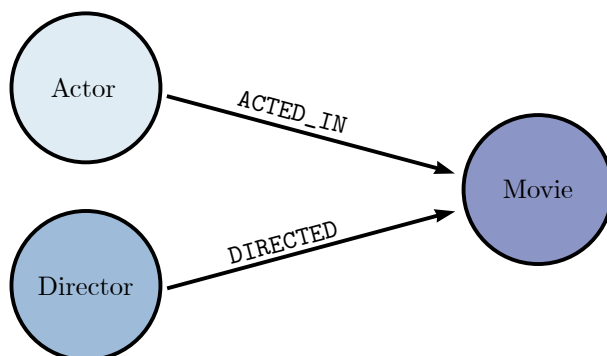


FIGURE 2.8: A graphical illustration of a graph data model representing the entities and relations in the film industry as well as the triplets that these entities and relations convey, *i.e.*  $\{\text{Actor}, \text{ACTED\_IN}, \text{Movie}\}$  and  $\{\text{Director}, \text{DIRECTED}, \text{Movie}\}$ .

### 2.1.6 Graph visualisations

Graph visualisations can provide qualitative insight into various properties of the considered graph, *e.g.* the level (or extent) of connectedness and the presence of communities (or clusters). Visual legibility in respect of graph visualisations may be expressed by means of so-called aesthetic criteria, notable examples of which include minimising edge overlaps, distributing vertices and edges uniformly, preserving symmetry and fixing edge length [70]. Visualising graphs effectively may be formulated as an optimisation problem in which multiple objectives are considered and the formulation of decision variables depends on the adopted visualisation approach [70]. A prominent methodology employed in the domain of graph visualisations is *force-based* (or *force-directed*) layout algorithms which adopt a physics-based approach towards generating graph visualisations [70]. Accordingly, a graph is expressed as a *system of bodies* representing the vertices of a graph, with forces acting on these bodies representing the edges. Graph visualisation is therefore equivalent to finding an equilibrium state for the force-induced system, *i.e.* positioning vertices in a graph such that the total force acting on each vertex is zero.

The *Kamada-Kawai spring layout* algorithm [134] is a popular force-based approach towards generating graph visualisations. Edges are expressed by means of straight lines while the positions of the vertices are algorithmically determined [134]. Kamada and Kawai define their spring model by introducing a dynamic system in which  $n = |\mathcal{V}|$  particles are defined,  $p_i$  denotes a particle corresponding to vertex  $v_i$ , for all  $v_i \in \mathcal{V}$ . By relating the balanced layout of vertices to the dynamically balanced spring system, the degree of imbalance may be formulated as the total energy of springs, that is

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{k_{ij}(|p_i - p_j| - l_{ij})^2}{2},$$

where  $l_{ij}$  denotes the length of the spring between particles  $p_i$  and  $p_j$ , calculated by means of the expression  $l_{ij} = L \times d_{ij}$ , where  $L$  denotes the desirable length of a single edge in the display plane and  $d_{ij}$  denotes the length of the shortest path between vertices  $v_i$  and  $v_j$ . Furthermore,  $k_{ij}$  denotes the strength of the spring in respect of particles  $p_i$  and  $p_j$ , expressed as

$$k_{ij} = \frac{K}{d_{ij}^2},$$

where  $K$  denotes a constant that is specific to the system, *e.g.* a spring constant in a simple mechanical system. A visually legible layout may therefore be obtained by minimising  $E$ .

Another force-based approach is the method employed by GraphViz [83] which is a popular graph visualisation library, especially so for large graphs. GraphViz employs dynamic bins (an extension of Fruchterman and Reingold’s technique [92]) in order to approximate long-distance repulsive forces, consequently mitigating the computational burden [83]. The original methodology proposed by Fruchterman and Reingold is analogous to molecular or planetary simulations, according to which vertices represent subatomic particles (or celestial bodies), exerting attractive and repulsive forces on one another which results in particle movement (or displacement). Each iteration of the algorithm comprises three steps. During the first two steps, the respective effects of attractive forces and repulsive forces in respect of each vertex are calculated. Thereafter, the total displacement of each vertex is constrained by some bound which decreases over time so as to refine the overall layout of the vertices [92]. A graphical illustration of a force-based algorithm employed for graph visualisation is depicted in Figure 2.9.

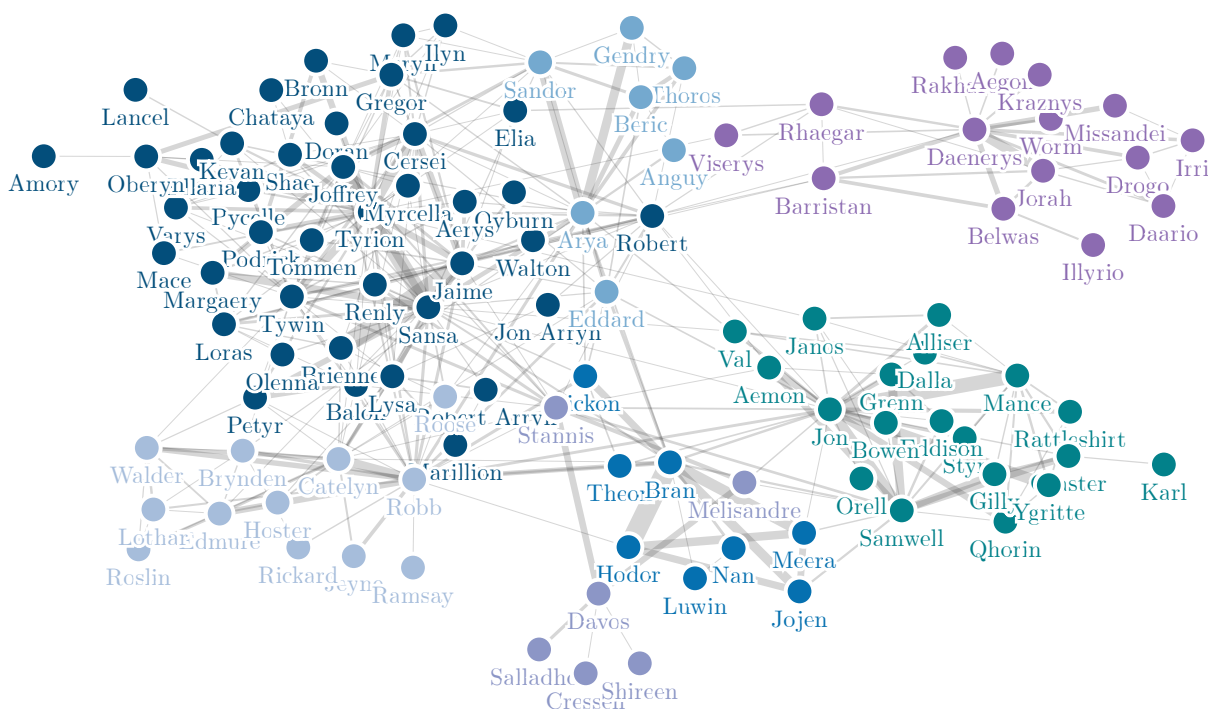


FIGURE 2.9: An example of a graph visualisation generated by a force-based algorithm in the context of a social network. The graph illustrates the interaction patterns among individuals who are segregated into distinct groups, as denoted by their respective colour designations.

### 2.1.7 Important graph terminology

Some key, contextual terminology employed in various graph-related discussions throughout this thesis is now defined.

**Structural properties:** Innate characteristics, such as degree distribution, component structure, cyclical patterns, and connectivity within a graph, which may be leveraged in order to infer underlying patterns and structures that encapsulate the constituent vertices and edges.

**Local information:** Structural properties pertaining to an individual vertex and its neighbouring vertices.

**Global information:** Structural properties pertaining to sub-graphs and/or the entire graph.

**Topology:** The general arrangement of vertices and edges within a graph, from which structural properties may be derived.

## 2.2 Statistical testing preliminaries

Statistical testing involves carrying out systematic (quantitative) analyses with respect to sample data in order to draw meaningful conclusions [10]. The aim in this section is to delineate relevant preliminaries pertaining to the field of inferential statistical testing. In this thesis, inferential statistical testing is employed to determine whether a significant statistical difference exists in respect of algorithmic performance data generated during various investigations and comparative studies.

### 2.2.1 Inferential statistical testing

Inferential statistical testing represents an effective approach towards drawing inference and formulating generalisations with respect to a *population* based on sample data [10]. In order to accomplish this, hypothesis testing is employed which involves defining a *null hypothesis*, denoted by  $H_0$ , and an *alternative hypothesis*, denoted by  $H_1$  [57]. The null hypothesis, conventionally assumed to be true, represents the statement of no effect, *i.e.* no difference between the data samples. The alternative hypothesis states the contrary, *i.e.* the presence of an effect (or a difference) in respect of the data samples. A *level of significance*, denoted by  $\alpha$ , reflects the level of confidence with which to reject  $H_0$  or not. The value of  $\alpha$  is compared with a so-called *p-value* which is computed by conducting some statistical test. The *p-value* can be interpreted conceptually as the probability of being incorrect if the null hypothesis is rejected [57]. Therefore, if  $p < \alpha$ , then  $H_0$  is rejected in favour of  $H_1$  at a significance level of  $\alpha$ .

*Parametric* statistical tests are often employed to conduct inferential statistical testing, however, their application is subject to the assumption that the considered data can be described by means of some specified distribution. Examples of common parametric tests include the *paired t-test* [101] which is employed to determine whether there is a statistically significant difference between the mean of two data samples, while the *analysis of variance* (ANOVA) test [88] represents a generalisation of the paired *t-test* which involves comparing mean differences between three or more samples. These tests mainly depend on three important assumptions, namely: (1) Observations (*i.e.* data instances) must be independent of each other, (2) the residuals (*i.e.* differences) should be approximately normally distributed within each sample, and (3) the variances within each sample should be equal, known as *homoscedasticity* [88]. *Non-parametric* tests, on the other hand, presuppose no assumptions in respect of the underlying distribution. Non-parametric statistical tests are employed in this thesis due to the lack of consensus in the literature regarding assumptions that may be made reliably in respect of link prediction algorithmic performance data. Abbas *et al.* [1] employed the Friedman test [91] to identify whether or not a significant statistical difference exists between the performance of various link prediction algorithms in the drug-discovery domain. Furthermore, Zheng *et al.* [322] employed the Friedman test to determine whether the performance of ML algorithms on imbalanced binary labelled data differs statistically significantly. The task of link prediction (the focal point in this study) may be formulated as a binary classification problem (discussed later) which is typically imbalanced. The Friedman test is therefore selected as the main inferential statistical test considered in this thesis together with an appropriate *post hoc* procedure.

### 2.2.2 Friedman test

The Friedman test [91] is employed when the assumptions pertaining to normality and homogeneity of variance are not necessarily met [123]. The null hypothesis  $H_0$  asserts that the difference between all sample median pairs is zero, *i.e.* they are equal, whereas the alternative hypothesis  $H_1$  asserts that the difference between at least two sample medians is not zero, *i.e.* they are not equal. The Friedman test involves converting the data points into *Friedman ranks* by sorting the data points in ascending order and subsequently ranking the samples [123].

Once a significant statistical difference is identified by the Friedman test, a *post hoc* procedure is employed in order to distinguish between each sample pair. The Nemenyi [209] procedure employs the Friedman ranks in order to perform two-tailed pairwise significant tests in respect of all sample pairs, from which the specific sample pair corresponding to a statistically significant difference can be identified [123]. Zheng *et al.* [322] employed the Nemenyi *post hoc* procedure in order to identify the best performing ML model with respect to imbalanced binary labelled data.

## 2.3 Clinical preliminaries

In this section, the reader is provided with an overview of the clinical prerequisites pertaining to the research conducted in this thesis. First, the reader is introduced to EHRs which include discussions on their content, purpose, and utility towards providing a comprehensive source of clinical data for perusal. Synthetic data generation in the clinical domain is discussed thereafter, followed by an explanation of medical ontologies and their practical value. Lastly, the notion of clinical KGs is elaborated upon.

### 2.3.1 Electronic health records

An EHR may be described as a digital repository in which patient data may be stored, securely exchanged, and accessed by various authorised entities [116]. The data stored within an EHR conventionally comprise retrospective, concurrent, and prospective patient information with the purpose of facilitating a holistic approach to patient care [116]. The increasing popularity of EHRs may be ascribed to an increased availability of (digital) patient data as well as improved practical and financial efficacy associated with integrating EHRs into existing health information systems. EHRs also facilitate a change in the perspective from which healthcare treatment is regarded as a singular doctor-patient relationship to the perspective from which a patient's care is deemed a joint responsibility in respect of multiple healthcare practitioners [103]. Healthcare institutions that typically employ EHRs include hospitals, pharmacies, and general practitioner surgeries [32].

Typically, a patient's EHR is frequently updated during *each* interaction by a healthcare practitioner. Traditional EHR incarnations may be categorised according to time-, source-, and problem-orientated information, each of which are stored separately [271]. More contemporary EHR approaches, however, combine these three categories into a single information system. Time-orientated data are expressed in a chronological format, whereas in the case of problem-orientated medical records, notes specific to each medical problem encountered by a patient are recorded individually. Each problem is characterised by subjective and objective information, relevant medical assessments, and a medical action-plan. Source-orientated records contain information pertaining to the manner in which the information was obtained. Examples may

include, consultation notes, lab tests, X-rays, scans, and blood tests [116]. The data within an EHR are expressed in either a structured format, such as standardised medical codes or nomenclatures, or in an unstructured format, such as natural language (free-text) data. A list of popular international terminologies is presented in Table 2.1 which represents structured data sources for EHRs [116].

TABLE 2.1: An overview of popular international medical terminology systems employed in EHRs [116].

Component	International terminology
Diagnoses	International classification of diseases [14]
Procedures	Current procedural terminology [240]
Medication	Anatomical therapeutic chemical classification index [201]
Pathological findings	Systematised nomenclature of medicine [115]
Nursing problems	International classification of nursing practice [80]

Information constituting an EHR may include patient demographics, clinical progress notes, past diagnoses, symptoms, prescribed medications, chronic illnesses, vital signs, immunisations, laboratory data, and radiology reports, to name but a few [116]. EHRs ought to be regarded as an integral component towards automating and streamlining the work flow of healthcare practitioners as it can facilitate a comprehensive analysis and synthesis of a patient’s medical examination — from which important use cases such as clinical decision support and quality assurance may be derived [304]. Furthermore, Häyrinen *et al.* [116] reported that EHRs can improve the data capturing process of medical information. The scope of their work included the analysis of 55 journal papers pertaining to the practical implementation of EHRs, from which it was reported that these dedicated medical information systems result in more accurate documentation of patient information. The data stored in the EHRs also exhibited greater accuracy [187], comprehensiveness [253], consistency [192], and reliability [132].

Another advantage associated with EHRs is their facilitation of tasks relating to continuously updating and maintaining a patient’s health record which ensures accurate and efficient healthcare delivery. EHRs can also be anonymised which permit greater accessibility and, consequently, improved research in respect of quality improvement [56] and public health surveillance [29]. Moreover, EHRs have been employed in numerous researches studies as a *supplementary* data source due to its ability to provide large-scale, historical data on numerous patients [97]. In this project, the raw data that serves as input for considered case studies represent information that forms part of an EHR — more specifically, information pertaining to conditions, and prescribed medication. As alluded to above, EHRs are particularly suitable to the research aim in this thesis as they encapsulate generic (historical) clinical information rendering it a robust data source for conducting algorithmic analysis. Data captured within EHRs are therefore employed towards demonstrating the utility of constructing a KG based on heterogeneous data sources and subsequently deriving data-driven insights therefrom.

### 2.3.2 Synthetic data generation

Real-world EHRs comprise highly sensitive and confidential patient information and, consequently, access to these data sources is often accompanied by various challenges. A majority of real-world clinical data sets are not suitable for immediate use due to the confidential nature of their contents. The process of distributing such data between the collectors, *i.e.* healthcare institutions, and researchers requires strict cooperation with government regulations, data usage agreements, protocols, and ethics approval guidelines [163]. Due to these challenges, re-

searchers are mostly confined to the analysis of anonymised data sets. Access, however, to anonymised data sets (the exchange of which is facilitated by governmental institutions commercial corporations, and clinical groups [272]) poses challenges due to privacy, confidentiality, and consent considerations. Inadvertent disclosure of anonymised data sets can result in the exposure of individuals to a high risk of identification — multiple cases in which patients have been re-identified from anonymised data sets have been reported [12, 81, 269]. Consequently, the number of anonymised data sets available for research is markedly limited which reduces the scope of research endeavours [284].

In order to circumvent or mitigate the aforementioned issues, researchers have leveraged the notion of synthetic data generation which involves the application of statistical modelling techniques towards generating synthetic data based on the statistical properties pertaining to real-world data [99]. So-called *realistic synthetic* EHRs (RS-EHRs) [284] are considered in this project due their relevance and accessibility. The manifestation of synthetic data may be described by means of the following three different forms, namely: Fully synthetic, partially synthetic, and hybrid [267]. A fully synthetic data set may be described as a data set comprising no affiliation to or association with any real-world data. These data sets adhere to privacy requirements but provide limitations in respect of analyses that can be carried out as a result of information loss [230]. Partially synthetic data, on the other hand, have sensitive data instances replaced with synthetic versions, however, the (original) data remain somewhat susceptible to reidentification in respect of the individuals constituting it [235]. The task of generating hybrid synthetic data involves employing both real-world data and synthetic data — more specifically, it involves randomly selecting a real data instance, matching the real data instance to a synthetic data instance containing similar information, and then synthesising the two instances in order to generate hybrid data. Although this method is more computationally expensive than its counterparts, it generates superior quality data whilst adhering to privacy control standards [267].

Synthetic clinical data have been employed in different investigations, *e.g.* simulation and predictive analytics, algorithmic testing, epidemiological studies, public health research, clinical IT development, as well as education and training [99]. Amoon *et al.* [11] employed synthetic data towards analysing the effect of electromagnetic fields on childhood leukaemia as a result of residential mobility. Synthetic data have also been employed in analyses that include conducting microsimulations for testing policy options [66, 270], evaluating alternative strategies for financing clinical care [118], validating simulation models [265], and improving the accuracy associated with predicting heart disease from various risk factors [9]. Ngufor *et al.* [212] and Dhanapal *et al.* [133] both employed synthetic data to verify the robustness, efficiency, and accuracy of their respective algorithmic approaches, while Chen [51] employed a publicly available synthetic medical insurance claims data set to evaluate a different algorithmic approaches towards phenotype discovery, noise analysis, scalability, and constraints analysis. In the case of epidemiological studies, Cooley *et al.* [58] employed a synthetic data set towards modelling the impact of subway travel in respect of transmitting the influenza virus during an epidemic. Other instances in which synthetic data are employed in the context of epidemiological studies include epidemiological modelling [114, 303], outbreak detection algorithms [96, 274], as well as the simulation of public health events and interventions [84].

Popular methods for generating RS-EHRs include Synthea [284], an open-source software package that generates high-quality RS-EHRs for the purpose of statistical modelling, and MD-Clone's Synthetic Data Engine which is a commercial tool that converts real-world EHR records into synthetic EHRs whilst maintaining the underlying statistical nature of the original data [234]. Although the focus in this thesis is on Synthea, a brief discussion is presented on MD-Clone's Synthetic Data Engine.

In 2020, Reiner *et al.* [234] conducted a study that involved validating the results obtained when analysing synthetically generated data by means of MD-Clone's Synthetic Data Engine. The study was conducted by developing a comprehensive validation process comprising various clinical-related questions which was subsequently applied on a variety of data types in order to assess the accuracy and precision of statistical estimates derived from the synthetic patient data. Synthetic data were generated for five contemporary studies and covered a variety of topics. The results obtained on the synthetic data were then compared with those based on real data for each of the studies. Moreover, the synthetic data sets were generated repeatedly in order to estimate their associated bias and stability. The results obtained from the synthetic data demonstrated utility and generalisation capabilities of techniques that are applied to synthetic data, showcasing that synthetic data represents a suitable alternative to real-world data. MDClone is, however, limited in that it is not open source, not freely available, and necessitates access to real patient data as an input. In this thesis, RS-EHR data generated by Synthea [284] and are employed to construct a clinical KG from which insights can be derived. Synthea is discussed in greater detail below.

## Synthea

In some case, investigations that involve generating clinical synthetic data sets are either too detailed or too general in scope when attempting to produce RS-EHRS across a variety of patient and condition types [202]. For example, certain synthetic clinical data sets are based on markedly specific details which limit their applicability in respect of diverse patient profiles. In other cases, however, these data sets may be too broad in scope and therefore fail to represent patient populations accurately. Walonoski *et al.* [284] stated that the most notable methods for generating RS-EHRS (at their time of publication) were limited due to their dependence on real patient records resulting in reidentification risks. The aforementioned issues contributed towards the development of Synthea [284]. As alluded to earlier, Synthea is an open-source software package capable of generating high-quality, realistic, and historical-based synthetic patient clinical records for the purpose of conducting analyses. The data from which Synthea was originally constructed pertain to a combination of publicly available clinical and census statistics, clinical reports, and clinical guidelines. The framework employed by Synthea is derived from the *Publicly available data approach to the realistic synthetic EHR* (PADARSER) [77] which assumes that access to real EHRs is not viable and/or undesirable — it is therefore based on publicly available data sets for populating the synthetic EHR. The PADARSER framework employs clinical guidelines and protocols in the form of so-called care maps. Synthetic data that have inherent and realistic properties are therefore generated which are appropriate for the context of representing realistic EHRs. The PADARSER framework is presented schematically in Figure 2.10.

PADARSER compiles public data from clinical incident statistics, clinical practice guidelines, and medical coding dictionaries in order to adhere to its underlying principle of privacy protection. The compiled data, to be subsequently extrapolated, serve as input to the generation process which are applied in respect of each synthetic patient. Input by clinicians and clinical practice guidelines are employed to develop the caremaps, while state-transition machines are used to configure temporal models for each patient. Finally, the data are supplemented further by incorporating regionally prevalent aggregate data sets, as well as additional clinician input, and clinical practice guidelines, each of which improves the realism of the generated data. The outcome of the framework, *i.e.* the RS-EHR, is characterised by sufficiently realistic properties in order to represent real EHRs at a large and diverse scale, while mitigating the risks associated with reidentification [284]. While Synthea may be employed for software, testing, validation,

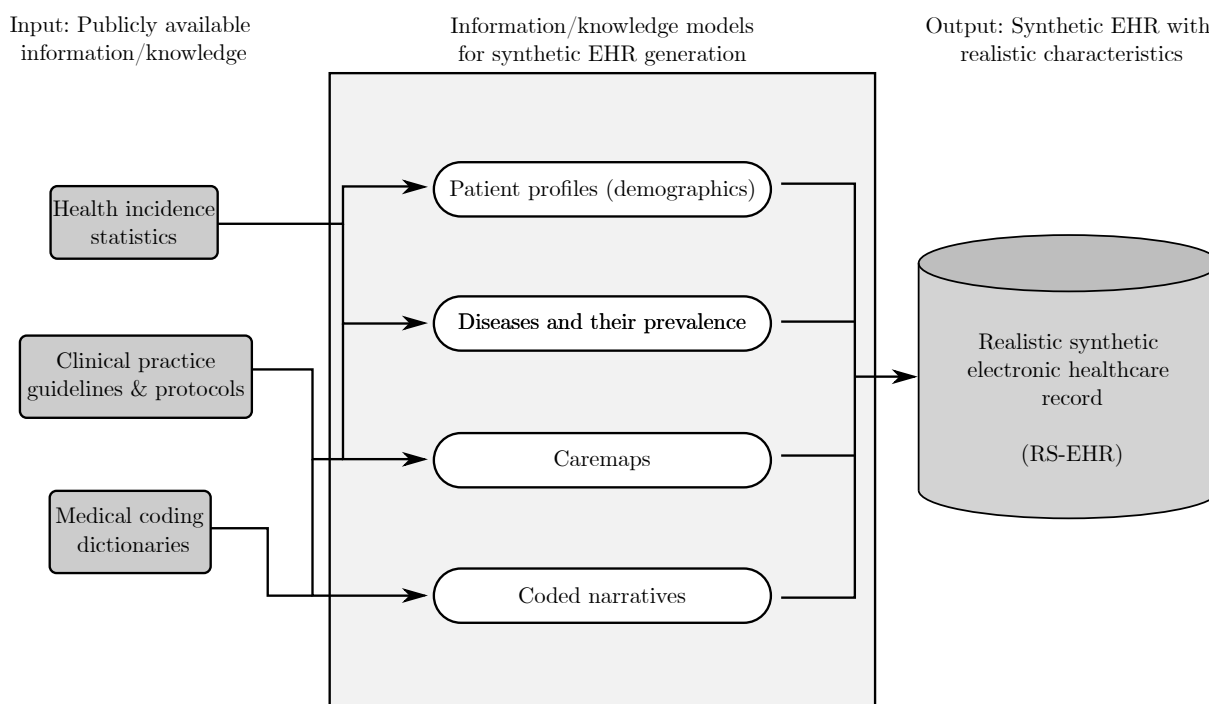


FIGURE 2.10: A graphical illustration of the PADARSER framework employed by Synthea (adapted from [284]).

and algorithm evaluation, it is not yet recommended for fault-less clinical discovery and scientific inference [6], as it does not *currently* account for variations in healthcare delivery and it is limited in respect of its heterogeneous clinical outcomes after major interventions [49, 285]. Its main advantage is to facilitate the task of conducting analyses from which actionable insight may be derived.

### 2.3.3 Medical ontologies

Medical ontologies may be defined as structured vocabularies that facilitate the task of describing the meaning of clinical-related data (*i.e.* its semantics), such that it may be interpreted by both humans and computational approaches [149]. In particular, medical ontologies describe concepts such as diseases, medications, proteins, experimental procedures, and surgical procedures, to name but a few. Medical ontology databases are often represented in a data format called *web ontology language* [188], however, subsets of these databases may be exported in formats such as `.csv`<sup>6</sup>. The need for medical ontologies may also be attributed to the increasing availability of medical data stemming from high-throughput experimental techniques. The extent of medical data is further accompanied by the rapid development of new medications and methods for conducting diagnoses which renders manual data processing markedly challenging. Although computational approaches may be employed to assist with data processing, notable challenges accompany them. Medical records often comprise an abundance of natural language in which different terms are employed to describe the same medical concept, while in other cases, specific medical terms can have differing meanings depending on nuanced contexts. Data formats and structures also vary greatly across departments and institutions which hinders integration. Towards addressing this issue, medical ontologies have been proposed [45].

<sup>6</sup>A common data format comprising comma-separated values.



A notable medical ontology is the SNOMED-CT [31] which comprises a hierarchical structure of clinical terms that may be employed to standardise clinical documentation and reporting. Approximately 300 000 distinct concepts, each of which can be linked to human-readable descriptions, constitute the SNOMED<sup>7</sup> database. Each concept is further linked to synonyms which in turn are linked to additional descriptions. Concepts are also related to one another by means of relationships which encode formal definitions of terms and provide their machine-readable meanings (semantics). A particularly prominent relationship in SNOMED, *i.e.* the *is-a* relation, facilitates the construction of the hierarchy of terminology within the database by linking detailed terms to more general terms (called parents). Medical concepts within SNOMED are categorised as follows: The term *clinical finding/disorder* is used to describe observations and results of patient assessments, *procedure* is used to describe various medical-related activities ranging from administrative procedures to diagnostic and surgical procedures, *body structure* is used to describe bodily locations, *organism* represents a hierarchical taxonomy of pertinent organisms, and, finally *pharmaceutical/biologic product* represents concepts pertaining to medications [149].

Another notable medical ontology is RxNorm [208] which provides standard names for medications with respect to the active ingredient, strength, and dosage, as administered to a particular patient. RxNorm links both branded and generic clinical drugs to their respective active ingredients and related brand names, providing a holistic representation of available clinical drugs. Moreover, RxNorm also links its names to various drug vocabularies commonly employed in pharmacy management and drug interaction software, thereby integrating systems based on different computerised implementations and vocabularies [198].

## 2.4 Clinical knowledge graphs

A so-called KG is a type of data model that represents a nascent approach towards abstracting and expressing clinical-related data, to which various algorithmic techniques may be applied in order to infer insight. As defined in §2.1.5, KGs comprise subject-property-object triplets which may be employed in a clinical context to denote relationships between patients and clinical entities, such as symptoms, diseases, lab results, and medications, as well as relationships between the clinical entities themselves. As an example, consider Figure 2.11 in which numerous vertex types and directed edge types may be employed so as to model the complexities that manifest within a clinical context. Furthermore, the triplets are represented by the head vertex, relation, and tail vertex, for example {Patient, PRESCRIBED, Medication}.

KGs can abstract both data and metadata, as well as complex (and often latent) relationships embedded within the data which facilitates the extraction of actionable insight from the problem domain [24, 170, 309]. Nelson *et al.* [207] showcased the utility of applying KGs to EHR data. Their methodology involved supplementing KG embeddings (discussed later) with clinical data which yielded improved performance in respect of predicting the diagnosis of multiple sclerosis amongst patients using ML. Rotmensch *et al.* [244] constructed a disease-symptom KG from EHR data using probabilistic models. The resulting KG, validated in respect of expert physician opinions, and the analysis thereof showcased that the automated construction of a clinical KG from EHR data *via* rudimentary concept extraction is feasible.

Abu-Salih *et al.* [3] conducted a thorough review of the literature pertaining to KG construction within the clinical domain in which a comprehensive taxonomy for clinical KGs was proposed. Furthermore, an exhaustive evaluation of state-of-the art techniques for KG construction in a

<sup>7</sup>For the sake of simplicity “-CT” is removed.

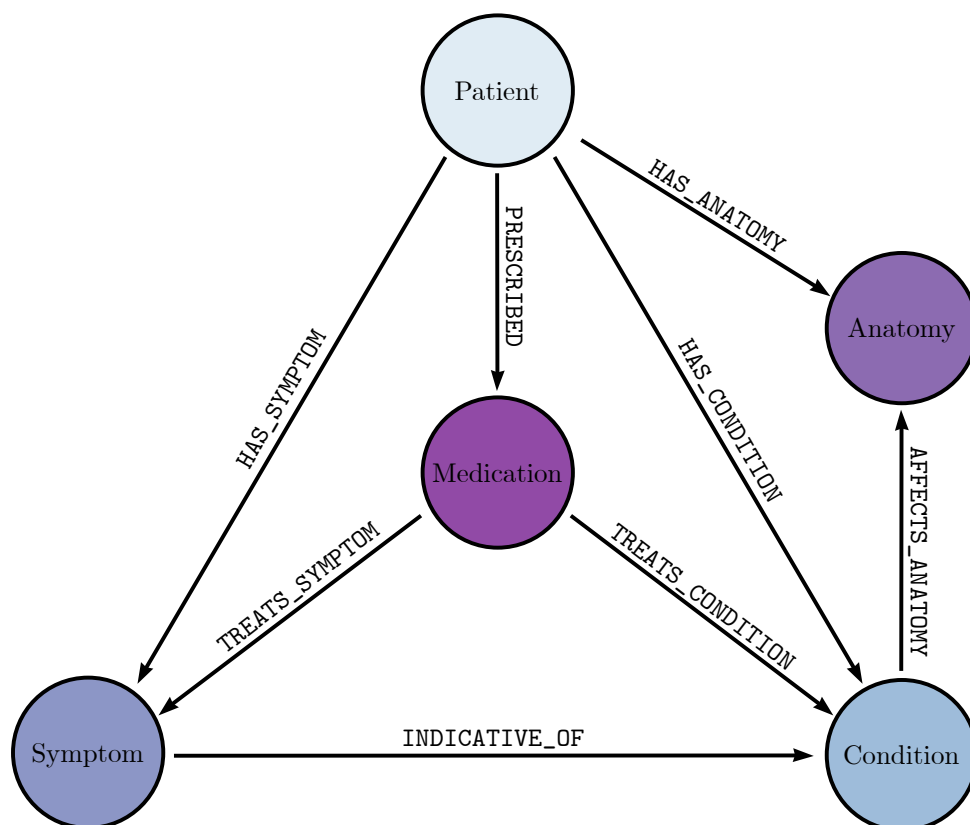


FIGURE 2.11: An example of a clinical KG schema (data model) comprising various vertex types and directed edge types. For example, a patient may be presented with a particular symptom, *i.e.* {Patient, HAS\_SYMPTOM, Symptom}, which is indicative of a specific condition, *i.e.* {Symptom, INDICATIVE\_OF, Condition}, and which may be treated by some medication, *i.e.* {Medication, TREATS\_SYMPTOM, Symptom}.

variety of clinical contexts was included. The proposed taxonomy, provided schematically in Figure 2.12, represented a novel contribution to the literature and was developed to improve the understanding of the emerging field of clinical KGs with respect to its primary components. The taxonomy also provides a general guideline for constructing clinical KGs by ensuring that its primary usage, method of knowledge extraction, type of knowledge base, and knowledge resource, as well as evaluation metrics are each explicitly defined.

In order to construct a KG, entities and relations are first retrieved from the considered data and subsequently employed to express the vertices and edges, respectively, of the KG. This is achieved by means of a process known as *knowledge extraction* — performed at both the entity-level and the relation-level. Three popular methods for performing entity extraction in respect of text data are *named entity recognition* [104] (NER), *named entity disambiguation* [258] (NED), and *named entity linking* (NEL) [257], as reviewed by Al-Moslmi *et al.* [199]. NER is employed to identify instances of entities within text data with respect to their associated “factual names” through either knowledge-based techniques or ML techniques. Knowledge-based techniques rely on domain-specific knowledge to identify entities, while ML-based approaches leverage annotated data, partially annotated data, or the data’s structural distribution. In order to distinguish between ambiguous terms, NEL and its constituent NED process are employed, according to which NEL involves linking an identified entity to an unambiguous instance of the same entity and frame it within a fixed context which is facilitated by a KG [3]. Relation extraction, on the other hand, aims to identify the manner in which two entities (vertices) are related to one

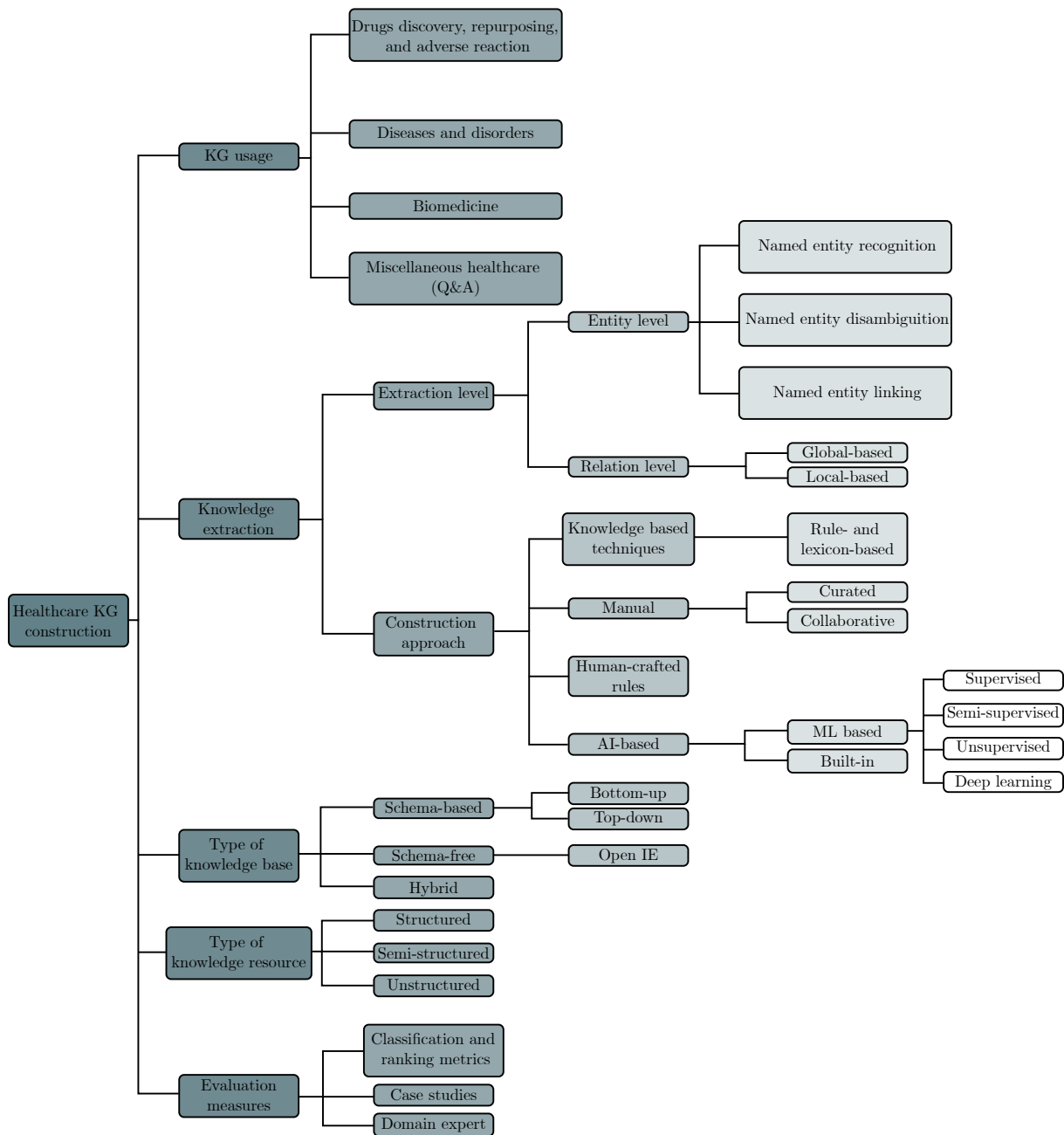


FIGURE 2.12: A taxonomy of KG construction in the clinical domain (adapted from [3]).

another. The two approaches for performing this task are global-based relation extraction which identifies relations across multiple knowledge bases, and local-based relation extraction which identifies frequent instances of relations from short passages of text [3, 146].

The construction of a KG is further governed by the type of knowledge base employed. Nickel *et al.* [214] define two primary knowledge bases, namely: *Schema-based* approaches and *schema-free* approaches. The former type is based on a predetermined ontology schema in which all possible relations are predefined *via* a fixed vocabulary, while the latter type is based on open information extraction strategies that rely on the open access to information (*e.g. via* the internet) and therefore follows no predefined schema, an example of which is OpenIE [85]. Schema-based approaches are further partitioned into bottom-up or top-down approaches. Hybrid approaches

which combine the aforementioned methods may also be employed to incorporate new information into defined ontologies [3, 214]. Abu-Salih *et al.* [3] describe three types of knowledge resources for healthcare, namely: Unstructured data sources (*e.g.* EHRs, medical literature, patient discharge summaries, radiology reports), semi-structured or tree-structured data sources such as mark-up based data, and structured databases such as relational databases that are tabular in nature.

KG evaluation is an important step towards evaluating the quality of the database for a practical setting [48]. An incomplete or inaccurate KG hinders the evaluation process as considerable effort is required to compile all factual data for a particular topic. Consequently, numerous efforts have been made to dynamically update KGs, referred to as KG completion. Furthermore, case studies and domain experts may also be used to perform evaluation [3, 197, 314]

### Related work

KGs are particularly advantageous with respect to processing large amounts of heterogeneous data — a prevalence in a clinical domain. Prominent uses case to which KGs have been applied include clinical drug (*i.e.* medication) studies, medical diseases and disorders, and biomedical studies [3]. With respect to clinical drug studies, in particular drug discovery, Mann *et al.* [186] developed a structured KG containing information in respect of symptoms, drugs, drug interaction, and non-medical information such as drug prices. The aim in the study was to consolidate information from different open-source drug-related databases into a single KG capable of assisting healthcare practitioners with performing complex queries effectively. For example, users would be able to find a potential drug for treating a given list of symptoms or disease as well as retrieve information pertaining to the particular drug's availability at nearby dispensaries by querying the KG. In another study, Ye *et al.* [307] developed a framework for predicting drug-target interactions<sup>8</sup> by combining a KG with a recommender system. The framework learns low-dimensional representations for the entities within the KG and then applies a so-called neural factorisation machine to construct a recommender system for facilitating drug-target discovery. Other notable works that employed KGs for drug discovery include Zhang *et al.* [319] who employed a KG neural network towards predicting potential interactions of various COVID-19 drugs and Zheng *et al.* [315] who proposed several representative embedding models that employ KGs for drug discovery.

KGs have also been employed in tasks that involve investigating the utility of using existing drugs to treat new diseases, such as COVID-19. The purpose of conducting such research, often referred to as drug repurposing, is to reduce the time and costs associated with new drug development [327]. COVID-KG, proposed by Wang *et al.* [290], is a KG constructed from heterogeneous data (sourced from relevant sources in the literature) which was then employed for question-answering and report generation in respect of a drug repurposing case study — clinicians and medical students validated the output highlighting its informative nature. KGs have also been employed towards studying adverse drug interactions. Bean *et al.* [24] constructed a KG from two drug databases upon which a logistic regression-based predictive model was constructed in order to predict new adverse reactions. The so-called Tumor-Biomarker KG proposed by Wang *et al.* [289] facilitates the discovery of new adverse reactions to drugs as well as providing an explanation for these reactions based on a KG derived from scientific biomedical literature.

---

<sup>8</sup>The identification of interactions between chemical compounds and the protein targets responsible for diseases in the human body [246].

A number of studies have employed KG approaches in respect of diseases. A notable example is the Health KG Builder framework that was developed by Zhang *et al.* [320] constructing a cardiovascular-specific KG from multiple clinical sources such as EHRs, medical standards, and domain expert knowledge. Malik *et al.* [184] constructed a KG from both structured and unstructured data sources of 1 025 patients comprising various concepts and different hierarchical and non-hierarchical relationships. A knowledge prediction model was subsequently trained on the KG in order to develop predictive rules for the prediction of subarachnoid haemorrhage. Choi *et al.* [55] studied the impact of genes in human disease by applying a convolutional neural network-based model on a biological<sup>9</sup> KG. They then constructed gene-gene interaction networks for different cancer types and employed graph analysis techniques to identify strongly correlated cancer genes. Ma *et al.* [181] addressed the limitations of previous prediction models in respect of dynamic illness severity data by combining patient status, structured medical knowledge, and drug usage to predict the trend of sequential organ failure assessment scores using a temporal convolutional network. Huang *et al.* [124] constructed a KG by consolidating different computational approaches pertaining to COVID-19 in which ML was employed to predict new treatment methods. A study by Yu *et al.* [310] focussed on chronic disease management and utilised a KG to develop a platform that can be employed for prescribing improved treatment and allocation of clinical resources. Huang *et al.* [125] conducted a study in which they generated a sub-graph from different knowledge bases<sup>10</sup> focusing specifically on major depression disorder. The purpose of the study was to demonstrate the manner according to which multiple large and generic KGs can be employed to construct a smaller disease-specific KG, facilitating the exploration of relationships between different sources.

Zheng *et al.* [323] constructed a KG comprising more than 500 000 individual connections between genes, drugs, and diseases. Each entity within the graph contained heterogeneous domain-specific information such as gene expression, chemical structures, and word embeddings for diseases. The authors further proffered various KG embedding approaches including a GNN-based method which incorporated both the global graph structure as well as heterogeneous domain features. In another study, Zhang *et al.* [321] constructed a KG for the discovery of causal relationships in biomedicine<sup>11</sup>. Their approach involved designing a causal event extraction method using deep learning and then connecting the events to construct a so-called causal knowledge network. Graph embeddings were then employed to determine feature representations of the network which were subsequently used to identify casual events. The study revealed that the casual network is able to discover information pertaining to potential medical causality, thereby facilitating disease diagnosis and treatment. Liu *et al.* [172] conducted a study in which a KG was developed that integrates information pertaining to gut microbiota from medical literature and medical knowledge bases. The developed KG may be employed for detecting possible relationships between gut microbiota, neurotransmitters, and mental disorders.

## 2.5 Chapter summary

This chapter opened with an introduction to the basic concepts and terminology relating to graphs in §2.1. Important notational conventions along with various diagrammatic examples supplemented the discussions. Thereafter, a discussion pertaining to the relevant statistical

---

<sup>9</sup>The term biological is used to describe KGs that represent data pertaining to biology, such as genes, proteins, diseases, and their interrelationships.

<sup>10</sup>A knowledge base (database) is a structured repository of information that captures domain-specific facts, relationships, and rules to facilitate reasoning and knowledge retrieval [213].

<sup>11</sup>The branch of medicine pertaining to the application of biological principles to medical research or practice [175].

---

preliminaries was presented in §2.2. In §2.3, a discourse pertaining to EHR data, synthetic data generation, and medical ontologies was addressed, while in §2.4, clinical KGs were discussed which represents important background and contextual information for assimilating the work carried out in the remainder of this thesis.



---

---

## CHAPTER 3

---

# Link prediction

### Contents

3.1	Overview of link prediction . . . . .	32
3.1.1	<i>Graph-based link prediction</i> . . . . .	32
3.1.2	<i>CRISP-DM</i> . . . . .	33
3.1.3	<i>Taxonomy of link prediction algorithmic approaches</i> . . . . .	34
3.2	Common-neighbour based link prediction . . . . .	34
3.3	Path-based algorithms . . . . .	36
3.4	Classifier-based algorithms . . . . .	37
3.4.1	<i>Machine learning paradigms</i> . . . . .	37
3.4.2	<i>Supervised learning link prediction</i> . . . . .	39
3.5	Embedding-based link prediction . . . . .	42
3.5.1	<i>Graph representation learning</i> . . . . .	43
3.5.2	<i>Graph embeddings</i> . . . . .	44
3.5.3	<i>Multi-relational data</i> . . . . .	49
3.5.4	<i>Graph neural networks</i> . . . . .	51
3.6	Link prediction on bipartite graphs . . . . .	59
3.7	Link prediction on knowledge graphs . . . . .	60
3.8	Performance evaluation . . . . .	64
3.8.1	<i>Evaluation metrics</i> . . . . .	65
3.8.2	<i>Graph partitioning</i> . . . . .	67
3.9	Chapter summary . . . . .	70

The purpose of this chapter is to provide the reader with an understanding of the fundamental concepts pertaining to the field of link prediction. The chapter opens with an introductory discussion on link prediction after which various link prediction algorithms are delineated, namely: CN-, path-, classifier-, and embedding-based algorithms. Thereafter, discussions on link prediction in bipartite graphs and KGs are presented in order to provide further context with respect to the challenges of performing link prediction on complex graphs. Performance evaluation in respect of link prediction is subsequently delineated which includes a discussion on evaluation metrics and graph data partitioning strategies. The chapter concludes with a summary of its contents.



### 3.1 Overview of link prediction

Link prediction is one of many analytical tasks that may be performed in respect of graphs in order to derive insight [179]. It aims to identify missing edges or new (future) edges in respect of non-adjacent vertices within a graph — in the clinical context of this thesis, link prediction may be performed towards supporting decision making in respect of condition diagnosis (*e.g.* to prevent misdiagnosis) or medication prescription (*e.g.* to suggest medication). Furthermore, other (unrelated) applications of link prediction include the analysis of user-user and user-content recommendations [78, 130], the identification of intercity transportation networks [182], predicting future friendships between individuals in a social network [168], as well as predicting the geographical dynamics of transnational terrorism [69]. Link prediction therefore represents a robust foundation from which various analyses may be devised in respect of problem domains that are characterised by interconnectedness. An in-depth discussion on link prediction now follows.

#### 3.1.1 Graph-based link prediction

Consider the undirected, unweighted<sup>1</sup> simple graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ . Let  $\mathcal{U}$  denote a so-called *universal* set which represents the set of all possible edges, therefore a maximum of  $|\mathcal{U}| = \frac{|\mathcal{V}|(|\mathcal{V}|-1)}{2}$  edges can be present, while the set of non-existent edges (*i.e.* non-adjacent vertex-pairs) is given by  $\mathcal{E}' = \mathcal{U} \setminus \mathcal{E}$ . A link prediction task therefore involves predicting the likelihood that a non-existent edge in  $\mathcal{E}'$  ought to be present in  $\mathcal{E}$  [179]. A visual representation of the link prediction problem is presented in Figure 3.1. Consider the graph  $\mathcal{G}$  where  $\mathcal{V} = \{A, B, C, D, E\}$  and  $\mathcal{E} = \{\{A,B\}; \{A,D\}; \{D,C\}; \{D,E\}\}$ . The link prediction task involves predicting whether a non-existent edge, indicated by the dotted lines, is either erroneously missing or is expected to form (due to temporal reasons). The set of non-existent edges may be expressed as

$$\mathcal{E}' = \{\{A,C\}; \{A,E\}; \{B,C\}; \{B,D\}; \{B,E\}; \{C,E\}\}.$$

Typically, a link prediction algorithm computes a score, denoted by  $s(v_i, v_j)$ , for all  $\{v_i, v_j\} \in \mathcal{E}'$  which represents an estimation of the likelihood corresponding to each non-existent edge  $\{v_i, v_j\}$  in respect of its presence. The calculation of such scores is contingent on the extraction of informative features in respect of the considered graph. The data-driven nature of link prediction renders it amenable to various methodological approaches from different computational domains. A discussion on the most popular approach follows.

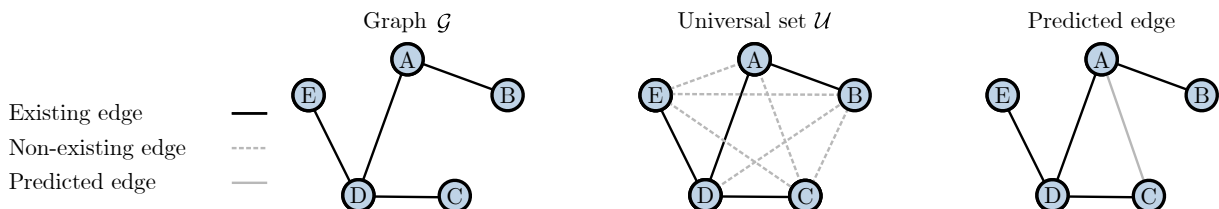


FIGURE 3.1: A visual representation of the link prediction problem according to which the existence likelihood of an edge belonging to the set  $\mathcal{E}'$  is to be estimated.

<sup>1</sup>Unweighted graphs may also be interpreted as weighted graphs for which each edge is assigned a weight of one.

### 3.1.2 CRISP-DM

A typical methodology adopted towards addressing data-driven problems (such as link prediction) involves the six phases of the *cross-industry standard process for data mining* [140] (CRISP-DM) methodology which represents a generalised high-level framework governing the execution of data mining projects (agnostic of industry). The six phases of the CRISP-DM methodology include *business understanding*, *data understanding*, *data preparation*, *modelling*, *evaluation*, and *deployment*. A graphical illustration of the CRISP-DM methodology is presented in Figure 3.2.

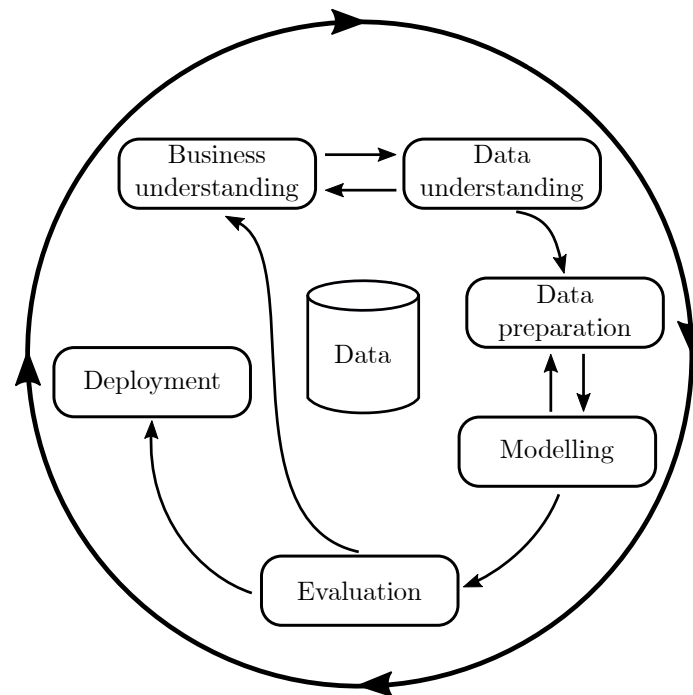


FIGURE 3.2: A graphical illustration of the six-step CRISP-DM lifecycle [330].

The business understanding phase of the CRISP-DM methodology is concerned with identifying and understanding the business objectives in order to devise and subsequently compile an appropriate project plan towards realising these objectives [140]. The next phase, *i.e.* data understanding, involves initial data collection which ought to be relevant to the business objectives. This aim in this phase is to gain a thorough understanding of the data which is accomplished by means of queries and visualisations as well as verifying the quality of the data in terms of both completeness and relevance.

The data preparation phase of the CRISP-DM methodology involves selecting the data to be analysed, cleaning these data, and then formatting the data in order to ensure conformity during the modelling phase [330]. The majority of real-world data sets are often incomplete and contain various flaws and inconsistencies. Such manifestations can relate to missing or unrealistic values (*e.g.* a negative value for a height measurement), corrupt values (*e.g.* incorrect text encoding), inconsistent data schemas, outliers, and other irregularities [286]. Missing values are usually represented as *not-a-number* values, blank entries, or some type of placeholder. These missing values may be addressed by means of a so-called *imputation* method which estimates suitable replacement values in a structured manner [140]. Popular imputation methods include assigning a zero value or some global constant to missing data entries. *Mean* or *median value imputation* may be employed for continuous (numerical) data, while *modal value imputation*

may be employed in the case of categorical data. *Random value imputation* and *heuristic-based imputation* presuppose certain assumptions based on domain knowledge. An important step of the data preparation phase involves partitioning the data into a *training*, *validation*, and *test set*. The training data set is used to train the selected algorithm, whereas the validation set is typically used to assess and tune different model hyperparameters so as to calibrate algorithmic performance. Finally, the test set is used to assess the final performance in an unbiased manner which represents data to which the algorithms have not been exposed [140].

During the modelling phase, one or more link prediction algorithms are selected, after which various computational analyses are conducted in order to analyse the performance. The selection of an appropriate algorithm may be based on the homogeneity of the graph data under consideration as well as the size of the data set in respect of number of instances and dimensionality of features [110]. A taxonomy of link prediction algorithmic approaches that may be employed during the modelling phase is presented later.

Evaluation and deployment represent the last two phases of the CRISP-DM methodology. During the evaluation phase, the results obtained from the modelling phase are analysed according to the objectives defined during the business understanding phase. The entire process undertaken during the first four stages of the CRISP-DM methodology is typically scrutinised in light of findings that stem from the evaluation phase which may aid in identifying potential areas of improvement. After evaluation has been conducted, the project either undergoes additional iterations of the modelling phase before deployment, or the project is deployed without further iterations. The deployment phase entails planning the activities associated with implementation and involves closely monitoring these activities in order to track the impact of the data mining project in a productionised context [330].

### 3.1.3 Taxonomy of link prediction algorithmic approaches

Various link prediction algorithms have been proposed in the literature — the reader is referred to the comprehensive overview of Wu *et al.* [298] for additional perusal. The intuition underpinning a large number of link prediction algorithms is the calculation of a similarity score between two vertices which may be used to derive a probability associated with either a link being missing or a link expected to form between two vertices. These computations are based on feature- and graph-based (*i.e.* structural) properties [298]. Algorithmic approaches range from traditional heuristic-based methods, which involve computing the number of common neighbours shared between two vertices, to more sophisticated techniques, such as supervised ML algorithms and network embedding-based methods [298]. A taxonomic overview of prevalent link-prediction algorithms, which includes CN-, path-, classifier-, and network embedding-based methods, is presented in Figure 3.3. A discussion on each of these algorithmic approaches is presented hereafter.

## 3.2 Common-neighbour based link prediction

CN-based algorithms arguably represent the simplest class of link prediction algorithms. Such approaches involve the calculation of a similarity score between a pair of non-adjacent vertices based on localised properties (or features) relating to the specific vertex-pair and its (immediate) neighbours [155]. Vertex-pairs that have large similarity scores are deemed more likely to have an edge joining them. After scores have been calculated in respect of all non-adjacent vertices, the vertex-pairs may be ranked according to similarity scores (typically in descending order)

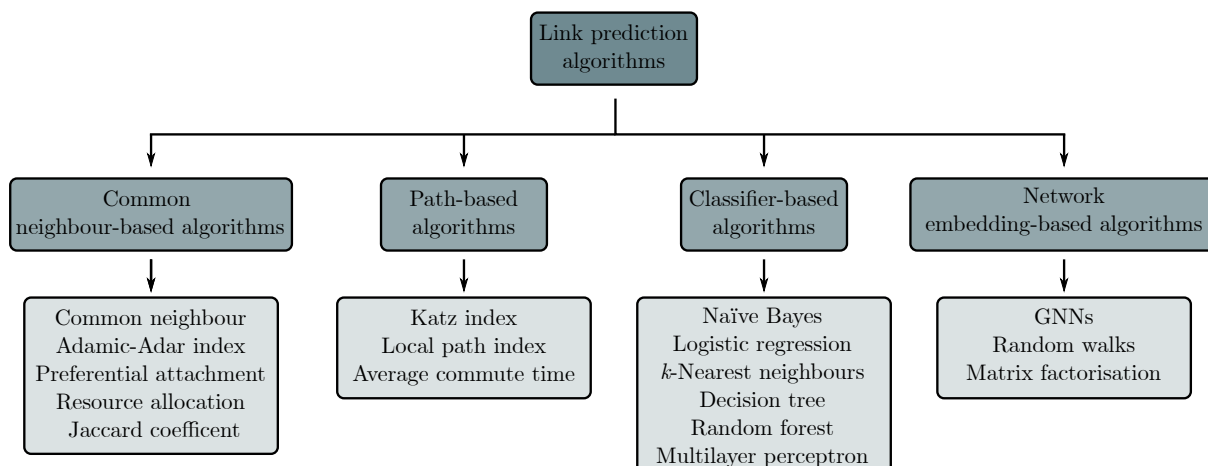


FIGURE 3.3: A taxonomic overview of link prediction algorithms (adapted from [298]).

after which a threshold may be applied to perform link prediction [167]. Similarity scores may be normalised in respect of a problem-specific or user-defined range, if necessary [93].

Popular common neighbour-based algorithms include CNs [211], *Jaccard coefficient* (JC) [127], *Adamic-Adar* (AA) index [4], *preferential attachment* (PA) [21], as well as *resource allocation* (RA) [326]. The number of CNs between a pair of vertices  $v_i$  and  $v_j$  equates to the cardinality of the intersection between the two vertices' neighbourhoods. The corresponding similarity score may be expressed as

$$s^{CN}(v_i, v_j) = |\mathcal{N}(v_i) \cap \mathcal{N}(v_j)|.$$

The likelihood of an edge being present between vertices  $v_i$  and  $v_j$  is therefore directly proportional to the number of shared neighbours between them. Despite its simple working, the method of CNs is reported to be effective in respect of different real-world data sets, outperforming other sophisticated approaches in certain problems [150, 151].

The JC expresses the similarity between vertices  $v_i$  and  $v_j$  in respect of the ratio between the number of CNs and the total number of neighbours in respect of the two vertices. The corresponding expression is

$$s^{JC}(v_i, v_j) = \frac{|\mathcal{N}(v_i) \cap \mathcal{N}(v_j)|}{|\mathcal{N}(v_i) \cup \mathcal{N}(v_j)|}.$$

The JC therefore represents a normalised measure of similarity which increases as the relative number of common neighbours increase. The AA index represents another extension of the CN approach according to which a weight is assigned to each common neighbour. The magnitude of each assigned weight corresponds to the degree of the neighbour, *i.e.* larger weight values are assigned to neighbours that have larger degree values. Accordingly, less connected neighbouring vertices are deemed more informative in respect of inferring potential edges. The AA index may be expressed as

$$s^{AA}(v_i, v_j) = \sum_{v_z \in \mathcal{N}(v_i) \cap \mathcal{N}(v_j)} \frac{1}{\log d(v_z)},$$

where  $d(v_z)$  denotes the degree of a shared neighbour  $v_z \in \mathcal{N}(v_i) \cap \mathcal{N}(v_j)$ .

PA is based on the notion that (new) vertices are more likely to join with (current) vertices that have many connections — *i.e.* the likelihood of link formation increases as the degree of a vertex increases [21]. PA may therefore be expressed as

$$s^{PA}(v_i, v_j) = d(v_i)d(v_j).$$

PA does not require extensive information with respect to the neighbouring vertices and is therefore less computationally burdensome.

The RA index originates from the work of Ou *et al.* [217] which relates to resource allocation dynamics within complex networks, such as international airport networks and electrical power grids. The basic working of the RA index may be expressed as follows: Consider two non-adjacent vertices  $v_i$  and  $v_j$ . Suppose vertex  $v_i$  may send some resources to vertex  $v_j$  via common neighbours of  $v_i$  and  $v_j$ . The similarity between vertices  $v_i$  and  $v_j$  may therefore be computed as the quantity of resources received by vertex  $v_j$  from vertex  $v_i$  (or *vice versa*) which may be expressed as

$$s^{RA}(v_i, v_j) = \sum_{v_z \in \mathcal{N}(v_i) \cap \mathcal{N}(v_j)} \frac{1}{d(v_z)}.$$

Although the RA index is rather similar to the AA index, a subtle distinction may be made. RA assumes that the importance of a common neighbour is inversely related to its number of adjacent vertices, modelling a scenario in which resources are allocated evenly amongst neighbours. The AA index, on the other hand, reduces the contribution of a common neighbour having many connections in a more gradual way when compared with RA index, as accomplished by the log function. The importance of common neighbours having large degrees is therefore scaled accordingly [179].

### 3.3 Path-based algorithms

Path-based algorithms involve computing similarity scores by considering existing paths between vertices. When compared with CN-based approaches, less localised and more global information pertaining to the graph structure is utilised. These methods are, however, associated with greater computational complexity and tend to be intractable in respect of large graphs [155]. Three popular path-based link prediction algorithms are discussed, they are *Katz index* [137], *local path index* [178], and *average commute time* [173].

The Katz index directly aggregates over all paths between vertices  $v_i$  and  $v_j$ , the summation of which is exponentially dampened in order to assign (exponentially) larger weights to shorter paths [179]. The Katz index may be expressed as

$$s(v_i, v_j) = \sum_{l=1}^{\infty} \beta^l \left| \mathcal{P}^l(v_i, v_j) \right|, \quad (3.1)$$

where  $\mathcal{P}^l(v_i, v_j)$  is the set of paths between vertices  $v_i$  and  $v_j$  of length  $l$ , and  $\beta^l$  denotes the damping factor for path-weight assignment. In order to improve convergence of (3.1), the specified damping factor  $\beta$  ought to be less than  $\frac{1}{\lambda_1}$  but greater than zero, where  $\lambda_1$  denotes the maximum eigenvalue of the adjacency matrix [137]. The summation over infinite path lengths corresponds conceptually to identifying all paths up to some practical limit or until paths no longer exist.

The so-called *local path index* [178] reduces the computational complexity associated with the Katz index by only considering paths of lengths two and three which may be expressed as

$$\mathbf{S} = \mathbf{A}^2 + \epsilon \mathbf{A}^3,$$

where  $\epsilon$  denotes another damping factor (similar to  $\beta$ ). For small damping factors, *i.e.*  $\epsilon \approx 0$ , the local path index converges to the aforementioned common-neighbour method.

The *average commute time* [173] in respect of vertices  $v_i$  and  $v_j$  relates to the sum of the average number of steps from  $v_i$  to  $v_j$ , and from  $v_j$  to  $v_i$ , which may be expressed by the pseudo-inverse of the Laplacian matrix. The Laplacian matrix (also called graph Laplacian) may be expressed as

$$\mathbf{L} = \mathbf{D} - \mathbf{A},$$

where  $\mathbf{D}$  denotes the so-called degree matrix which may be expressed as the diagonal matrix, according to which an entry corresponds to

$$D_{v_i, v_j} = \begin{cases} d(v_i), & \text{if } v_i = v_j, \text{ or} \\ 0, & \text{otherwise.} \end{cases}$$

The Laplacian matrix contains important graph properties from which insight may be inferred. The corresponding expression for average commute time is expressed as

$$s(v_i, v_j) = \frac{1}{L_{v_i, v_i} + L_{v_j, v_j} - 2L_{v_i, v_j}}.$$

## 3.4 Classifier-based algorithms

ML algorithms may be applied to the task of link prediction, according to which the likelihood (or score) of a link's presence is predicted base on different informative features of the non-adjacent vertices under consideration. In 1959, Arthur Lee Samuel defined ML as the “field of study that gives computers the ability to learn without being explicitly programmed” [245]. Furthermore, according to the definition by American computer scientist Tom Mitchell, a computer program is said to *learn* from *experience E* with respect to some *class of tasks T*, with an associated *performance measure P*, if its performance  $P$ , in tasks  $T$ , increases with experience  $E$  [194]. For example, a computer program that seeks to *identify images containing people (T)*, may improve its *ability to identify such images correctly (P)*, by analysing and extracting inferential patterns from a *data set comprising images of people (E)*. The remainder of this section includes discussions on the different paradigms of ML with a particular focus on paradigms pertinent to the analyses carried out in this thesis.

### 3.4.1 Machine learning paradigms

The realm of ML may be categorised according to four main paradigms, namely: *Supervised learning*, *unsupervised learning*, *semi-supervised learning*, and *reinforcement learning*. Each ML paradigm is distinct in respect of its methodological working and may be applied to a variety of use cases, depending on the task at hand as well as the nature and extent of the data available [205]. Due to scope delimitations of this research project (discussed in §1.3), the focus in this thesis excludes reinforcement learning, however, a discussion thereon is presented nonetheless.

#### Supervised learning

In the case of supervised learning, training data comprising input features (representing the independent variables) and their corresponding (known) output or target features (representing the dependent variables) are considered [138]. Collectively, the input data together with the

output data are referred to as *labelled data*. The aim in a supervised learning task is to approximate a functional mapping from the independent variables to the dependent variables — a predictive model that predicts correct outputs based on newly presented input data is therefore constructed [159, 205]. Supervised learning may be further partitioned into two main categories, namely: *Classification* and *regression* [18]. In the case of classification problems (which is the focal point in this thesis), the output domain is categorical (*i.e.* countable, finite categories), *e.g.* customer segmentation and spam detection, whereas regression problems are characterised by an output domain that is continuous (*i.e.* real-valued) [126, 205]. Examples of regression problems include sales and production yield forecasting [205]. Popular supervised learning algorithmic approaches include *linear regression*, *logistic regression (LR)* [28], *Poisson regression* [206], *Naïve Bayesian (NB) classifiers* [237], *support vector machines* [280], *k-nearest neighbours (kNNs)* [141], and *decision trees (DTs)* [228], to name but a few.

## Unsupervised learning

Unsupervised learning involves the analysis of *unlabelled data*, *i.e.* data comprising only input features, the aim of which is to uncover the latent structure within the data and infer actionable insight from identified patterns [138, 205]. The three main use cases of unsupervised learning are *clustering*, *dimensionality reduction*, and *density estimation* [205]. Clustering involves categorising input data into relevant subgroups that maximise (or minimise) some similarity (or dissimilarity) measure, without stating any prior hypothesis in respect of the subgroups' characteristics [18]. Clustering methods may provide insight into structural properties of the data so as to facilitate the data's analysis (depending on the task at hand). Examples of popular clustering algorithms include *centroid-based clustering* [135] algorithms, *e.g. k-means clustering* [166], and *distribution-based clustering* [129].

The number of input features, *i.e.* the dimensionality of the feature space, is often markedly large for real-world problem instances, resulting in a large computational burden when attempting to visualise the data or subjecting the data to certain algorithmic approaches — typically referred to as the “curse of dimensionality” [18]. Dimensionality reduction transforms a high-dimensional feature space into one that comprises markedly fewer dimensions (typically two or three, especially if the task at hand is visualisation) while maintaining a majority of the data's information (typically expressed as variance). The removal of uninformative features (or *noise*) can improve the performance of supervised learning algorithms in respect of predictive capability and/or and computational time [18, 205]. Dimensionality reduction may be partitioned into two distinct categories, namely: *Feature selection* and *feature extraction*. The former category involves filtering irrelevant (or redundant) features from the data set, whereas the latter category involves the construction of new, transformed features [154]. Popular dimensionality reduction techniques include *principal component analysis* [2], *t-distributed stochastic neighbour embedding* [119], and the method of *uniform manifold approximation and projection* [189].

The aim of the final unsupervised learning use case, *i.e.* density estimation, involves estimating some latent distribution, *i.e.* an underlying probability function [205]. Notable algorithms within the realm of density estimation include *kernel density estimation* [242], *Gaussian mixture models* [236], and *histogram estimation* [254]. Density estimation algorithms facilitate an improved understanding of the data's inherent structure and characteristic. For example, density estimation is typically employed in anomaly detection for the purpose of identifying data instances that diverge from an estimated distribution [203]. Moreover, density estimation algorithms can be employed to generate new, realistic data instances based on the learnt distribution of an existing data set [232].

### Semi-supervised learning

Semi-supervised learning may be performed in respect of data sets comprising a combination of labelled and unlabelled data. Typically, the extent of unlabelled data markedly exceeds that of labelled data which may simply be ascribed to the nature of data collection in general [159]. A semi-supervised learning algorithm performs either a supervised or unsupervised task. In the case of a supervised learning task, a distinction may be made between *inductive* and *transductive* learning. Inductive learning involves training an algorithm on a combination of labelled and unlabelled data so as to perform predictions on new (unseen) data instances that were not part of the original training dataset [185]. Transductive learning methods, on the other hand, infer the correct labels for the unlabelled data by incorporating the labelled data. Incorporating unlabelled data presupposes the presence of relationships in respect of the latent distribution of the data. Consequently, at least one of the following three assumptions ought to hold true, namely: *Continuity assumption*, *cluster assumption*, and *manifold assumption* [205]. The continuity assumption assumes that similar data points (in respect of the feature space) are likely to share the same label. The cluster assumption assumes that data points corresponding to the same cluster share the same label. It is important to note that the cluster assumption differs from the continuity assumption in respect of its focus on global structure within the data, whereas the continuity assumption pertains to the notion of local smoothness within the data. Finally, the manifold assumption assumes that high-dimensional data can often be expressed by means of a lower-dimensional representation [205].

### Reinforcement learning

Reinforcement learning is a behavioural learning approach that is based on principles similar to those that underpin human learning [126, 138]. Reinforcement learning comprises three main components, namely: An *agent*, an *environment*, and a *set of actions*. During each iteration of the learning process, an agent chooses an action based on the state of the environment, in an attempt to achieve some pre-defined goal [205]. The agent is then *rewarded* (or *punished*) for the chosen action depending on whether the action results in an outcome that is *favourable* (or *unfavourable*) towards the pre-defined goal. The agent then incorporates this feedback in order to adjust its decision strategy (*i.e.* policy) which is repeated until some stopping criteria are met. The learning process employed during reinforcement learning may therefore be described informally as a process of trial-and-error. Prominent reinforcement learning approaches include *policy iteration*, *value iteration*, the *state-action-reward-state-action algorithm* [268], and the *R-Markov average reward technique* [328].

#### 3.4.2 Supervised learning link prediction

Link prediction may be formulated as a classification task to which various algorithmic approaches may be applied, as mentioned in §3.4.1 [113]. Notable examples of classification algorithms that have been applied to link prediction include LR [30], NB [237], *k*NNs [141], DTs [228], random forests (RFs) [36], and *multilayer perceptrons* (MLPs) [195]. More specifically, the link prediction task is expressed as a binary classification problem in which a data instance corresponds to a pair of vertices and the label of the data instance indicates the presence or absence of an edge between that pair. The input data to a link prediction task are represented by a collection of descriptive features for each pair of vertices, denoted by  $\mathcal{X}$ , while the output data are represented by labels for each pair of vertices, denoted by  $\mathcal{Y}$ . Consider the vertex-pairs



$v_i, v_j \in \mathcal{V}$ , each of which has a label  $y_{v_i v_j} \in \mathcal{Y}$ , or simply  $y_{ij}$ , such that

$$y_{ij} = \begin{cases} 0, & \{v_i, v_j\} \notin \mathcal{E} \\ 1, & \{v_i, v_j\} \in \mathcal{E}. \end{cases}$$

Accordingly,  $y_{ij} = 0$  represents a missing edge (negative class) and  $y_{ij} = 1$  represents an existing edge (positive class). Furthermore, each vertex-pair is assigned a feature vector, denoted by  $\mathbf{x}_{v_i v_j} \in \mathcal{X}$  (or simply  $\mathbf{x}_{ij}$ ) representing descriptive graph-based properties which may be categorised as follows: (1) Structural properties of the graph, *e.g.* vertex degree distribution and community structure, and (2) feature-based properties which relate domain-specific information of the real-world problem itself, *e.g.* demographic features of individuals in a social network [27]. Given  $p$  features, the feature vector may be expressed as  $\mathbf{x}_{ij} = [x_{ij}^{(1)} x_{ij}^{(2)} \cdots x_{ij}^{(p)}]$ . Link prediction (formulated as a supervised learning task) involves predicting the label of negative instances based on their corresponding input features. This prediction is governed by the functional mapping

$$f : \mathcal{X} \mapsto \mathcal{Y}$$

which is learnt during training. The manner according to which training is carried out depends on the algorithmic approach adopted. A discussion on popular approaches follows.

### Naïve Bayes

The NB classifier is a probabilistic classifier based on the application of Bayes' theorem — the assumption of feature independence is therefore of particular importance [237]. The application of Bayes' theorem within a binary classification context may be expressed as

$$P(y_{ij}|\mathbf{x}_{ij}) = \frac{P(y_{ij})P(\mathbf{x}_{ij}|y_{ij})}{P(\mathbf{x}_{ij})}, \quad (3.2)$$

where  $P(y_{ij}|\mathbf{x}_{ij})$  denotes the posterior probability of class  $y_{ij}$  given input features  $\mathbf{x}_{ij}$ , and  $P(\mathbf{x}_{ij}|y_{ij})$  denotes the probability of observing the feature vector  $\mathbf{x}_{ij}$  given  $y_{ij}$ . Feature independence, admittedly a naïve assumption of NB, is encapsulated in the following probability calculation

$$P(\mathbf{x}_{ij}|y_{ij}) = \prod_{w=1}^p P(x_{ij}^{(w)}|y_{ij}).$$

One variant of the NB classifier is the Gaussian NB classifier which assumes that the input features follow a Gaussian distribution, expressed as

$$P(x_{ij}^{(w)}|y_{ij}) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_{ij}^{(w)} - \mu_y)^2}{2\sigma_y^2}\right),$$

where  $w \in \{1, \dots, p\}$ , while  $\mu_y$  and  $\sigma_y$  denote the sample mean and sample standard deviation, respectively [128]. A notable paper by Liu *et al.* [174] employed the NB classifier to develop a link prediction model achieving accurate predictions in respect of nine real-world graph-based problem instances, thereby showcasing its suitability to the link prediction task. Furthermore, Raut *et al.* [231] stated that although NB demonstrated reasonable link prediction results on real-world data sets, imbalanced datasets present a challenge.

### Logistic regression

LR is a probabilistic, discriminative classifier that employs the logistic (sigmoid) function to map predicted values to the unit interval. The LR model may be expressed as

$$P(y_{ij}|\mathbf{x}_{ij}) = \frac{1}{1 + \exp\left(-\left(\beta_0 + \beta_1 x_{ij}^{(1)} + \dots + \beta_n x_{ij}^{(p)}\right)\right)}, \quad (3.3)$$

where  $\beta_0, \beta_1, \dots, \beta_p$  are the model coefficients ( $\beta_0$  represents the bias) [131]. Model coefficients are typically derived by means of the *maximum likelihood* method which involves finding coefficient values such that (3.3) is maximised in respect of instances for which  $y_{ij} = 1$  and minimised in respect of instances for which  $y_{ij} = 0$ . Let  $o$  denote the number of training observations. The corresponding likelihood function may therefore be expressed as

$$\prod_{s=1}^o P\left(y_{ij}^{(s)} = 1 | \mathbf{x}_{ij}^{(s)}\right)^{y_{ij}^{(s)}} \left(1 - P\left(y_{ij}^{(s)} = 1 | \mathbf{x}_{ij}^{(s)}\right)\right)^{1 - y_{ij}^{(s)}},$$

according to which each observation in the training data set is regarded as a Bernoulli trial. Upon computing the logarithm the expression

$$\sum_{s=1}^o \log\left(P\left(y_{ij}^{(s)} = 1 | \mathbf{x}_{ij}^{(s)}\right)\right) + \left(1 - y_{ij}^{(s)}\right) \log\left(1 - P\left(y_{ij} = 1 | \mathbf{x}_{ij}^{(s)}\right)\right)$$

is obtained.

### $k$ -nearest neighbours

The  $k$ NN classifier takes as input a positive integer  $k$  and an observation  $\mathbf{x}_{ij}$  and identifies the  $k$  *closest* points to  $\mathbf{x}_{ij}$  according to some adopted distance measure, denoted by  $\mathcal{N}_0$ . The conditional probability of class  $y_{ij}$  may be expressed as the fraction of observations in  $\mathcal{N}_0$  whose response values correspond to  $y_{ij}$ , expressed as

$$P(y_{ij}|\mathbf{x}_{ij}) = \frac{1}{k} \sum_{m \in \mathcal{N}_0} I(y_{ij}),$$

where  $I(y_{ij})$  is an indicator variable, according to which a value of one is assigned if  $y_{ij} = 1$  or zero otherwise [131]. A notable advantage of the  $k$ NN classifier is its simplicity, however, it is less suited to categorical features or high-dimensionality problem instances [160]. Aouay *et al.* [13] employed both topological features and domain-specific features to predict edges within a co-authorship network for which the  $k$ NN algorithm showcased superior predictive performance when compared with other supervised ML classifiers.

### Decision trees

Predictions carried out by means of DTs [228] are based on the derivation of simple decision rules based on the training data. A DT is constructed through binary recursive partitioning which involves iteratively *splitting* the data into partitions based on the most frequently occurring class within the training observations of that specific partition. In the context of classification problems, the best split may be determined by employing the *Gini index*, expressed as

$$G = \sum_{k=1}^K \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

where  $\hat{p}_{mk}$  is the proportion of training observations in the  $m^{\text{th}}$  partition which originate from the  $k^{\text{th}}$  class, and  $K$  denotes the number of classes, *i.e.*  $K = 2$  for binary classification [131]. If each value in  $\hat{p}_{mk}$  is close to either zero or one, the Gini index has a correspondingly small value which suggests that the observations contained in partition  $m$  may be classified into one of the  $K$  classes with a reasonably high level of confidence. Similarly, the *cross-entropy* measure, expressed as

$$H = - \sum_{k=1}^K \hat{p}_{mk} \log \hat{p}_{mk},$$

reflects small values for high-quality splits. Hasan *et al.* [113] reported that the DT was able to produce statistically similar results to the best performing classifier algorithms employed in their link prediction study.

### Random forests

RFs [36] are a type of *ensemble* learning method in which multiple DTs are constructed during training, each of which is developed from a different (random) sample of the original training data — a process known as *bootstrapped* aggregation, or *bagging*. The bootstrap data sets are randomly sampled according to a uniform distribution from the original data set (with replacement). In the case of a classification task, the final prediction of the RF model is performed by means of a majority vote approach in respect of the individual trees, resulting in a robust predictive model [36]. RFs have been applied to a number of link prediction studies in which favourable performance is demonstrated [61, 250, 299].

### Multi-layer perceptron

An MLP is an architectural manifestation of feed-forward artificial neural networks comprising at least three computational layers, *i.e.* an input layer, one/more hidden layers, and an output layer [195]. The hidden layers are responsible for learning different abstractions within the data. Their constituent computational modules (called neurons) perform a range of operations in respect of the input data. In the context of binary classification, the output layer comprises a single neuron which employs a sigmoid activation function yielding a (probabilistic) value between zero and one. The output may be interpreted as the probability of an instance belonging to the positive class which then is subjected to a threshold (*e.g.* 0.5) in order to obtain a binary prediction. During training the network's weights are adjusted algorithmically by propagating the output error backward through the network in order to determine the extent to which weights contribute towards the network's prediction error, a process referred to as *backpropagation* [195]. The weights are updated in an iterative manner so as to minimise a loss function, typically the *cross-entropy loss* in the case of binary classification [100]. Link prediction studies in which MLP achieved favourable performance include Hasan *et al.* [113] and Elkabani *et al.* [82].

## 3.5 Embedding-based link prediction

In this section, a discourse pertaining to embedding-based link prediction is presented. First, the field of graph representation learning is introduced which includes an overview of different problem domains relevant to the field. Thereafter, the mathematical prerequisites pertaining to graph embeddings are presented which is followed by a discussion pertaining to embeddings within the context of multi-relational graphs. Lastly, GNNs are explored in greater detail which

includes a discussion on the specific architectures relevant to the numerical work conducted in this thesis.

### 3.5.1 Graph representation learning

The inherent complexity of graph-based data may be ascribed to the relational information embedded within. The innate non-Euclidean nature of graphs is accompanied by various computational challenges, namely: Conventional distance metrics (employed, for example, to measure similarity in feature space) are not appropriate, the curse of dimensionality, and sparsity, to name but a few [44]. Defining rigid structural priors for graph data is challenging due to the sheer representational capacity possessed by graphs in respect of its topological structure (defined by the manner according to which vertices are joined by edges) [44]. Consequently, techniques commonly employed in respect of Euclidean data (*e.g.* image-based data with spatial dependencies or tabular data comprising features that are abstracted by means of a traditional Cartesian coordinate system) are notably ineffective and inefficient when applied to graph-based data. For example, the neighbourhood structure of each pixel under consideration within an image is identical, as each pixel is surrounded by a fixed number of neighbouring pixels arranged in a *regular grid*. Vertices within a graph, on the other hand, may each correspond to a distinct neighbourhood structure — *i.e.* varying in respect of size and the nature of its constituent neighbours — which renders the task of defining a generalised approach notably challenging.

The aforementioned challenges led to the development of geometric deep learning — *i.e.* the application of deep learning<sup>2</sup> techniques to non-Euclidean data<sup>3</sup>. An important component of geometric deep learning is *graph representation learning* (GRL) which aims to learn low-dimensional real-valued vector (*i.e.* Euclidean) representations, called *embeddings*, of graph-structured data [44]. A detailed discussion on embeddings is presented later. The two main learning tasks within GRL are (1) *unsupervised* GRL, in which the aim is to learn embeddings that preserve the structural properties of a graph in respect of its (input) features and (2) *supervised* (or *semi-supervised*) GRL, in which the aim is to learn embeddings in respect of a particular prediction task. Three popular GRL tasks include *node*<sup>4</sup> *classification*, *clustering and community detection*, *link*<sup>5</sup> *prediction*, and *graph classification* [108]. Although the focal point of this thesis is link prediction, brief discussions are presented in respect of node classification, clustering (*i.e.* community detection), and graph classification nonetheless. A detailed discussion on the task of link prediction is presented later in this chapter.

#### Node classification

A necessary precursor of a discussion on node classification relates to the notion of a label which represents some attribute that is assigned to the node (or vertex) [108]. Given a graph  $\mathcal{G}$ , its vertex set  $\mathcal{V}$ , and a set of predefined labels, denoted by  $\mathcal{L}$ , node classification involves predicting a label  $l_j \in \mathcal{L}$  for some vertex  $v_i \in \mathcal{V}$ . Notable practical examples of node classification include classifying the functions of proteins in a set of molecular interactions within a cell [110] and

<sup>2</sup>Deep learning represents a prolific solution methodology in the domain of ML [126]. Deep learning approaches employ large neural networks that learn insightful abstractions within data in an iterative manner. Such approaches can learn from vast amounts of data effectively. The term deep learning is typically used when artificial neural networks comprise multiple hidden layers [126].

<sup>3</sup>Data with an underlying structure that exists in a non-Euclidean space, *i.e.* the principles of Euclidean geometry do not apply [39].

<sup>4</sup>The terms vertex and node are used interchangeably in the literature.

<sup>5</sup>The terms edge and link are used interchangeably in the literature.

classifying documents within a citation graph according to their topic [147]. The key difference between node classification and standard supervised classification is that the vertices in a graph are not *independent and identically distributed* (i.i.d.). A fundamental assumption in the case of most standard supervised classification tasks is that features (defined in tabular format) are i.i.d [108]. Node classification therefore involves leveraging the relationships (*i.e.* edges) between vertices explicitly. A popular approach, as reported by McPherson *et al.* [190], involves utilising the sociological notion of *homophily* which asserts that individuals who share similar characteristics are often more likely to interact with one another [313]. In the context of graphs, homophily corresponds to the tendency of vertices to be connected with other vertices that share the same (or similar) attributes. This notion may be utilised in an algorithmic manner to perform node classification [324]. Another concept upon which node classification algorithms is based is *structural equivalence*, *i.e.* vertices sharing similar local neighbourhood structures share similar labels [76]. Another concept relates to *heterophily* which, contrary to homophily, asserts that vertices are more likely to be connected to other vertices that do not share similar attributes, *e.g.* in the case of a social network, gender-based relations may be described by means of heterophily [108]. Node classification is conventionally considered a semi-supervised ML task due to the presence of both labelled and unlabelled data points (vertices).

### Clustering and community detection

The task of clustering in respect of graphs (also referred to as community detection in the GRL literature) is an unsupervised learning task, similar to conventional clustering of tabular data. A community structure within a graph may be qualitatively defined as the grouping of vertices in clusters, according to which vertices from the same cluster are joined to one another by many edges while vertices belonging to different clusters are joined to one another by relatively few edges [90]. The aim in a graph-based clustering task is to uncover latent community structures within the graph which may then be leveraged for different analytical purposes [108]. The utility of clustering and community detection applies to a diverse range of real-world applications such as the identification of functional modules in genetic interaction networks [5] and the detection of fraudulent groups of users in financial transaction networks [218].

### Graph classification

Graph classification involves performing predictive tasks by considering the *entire graph*, as opposed to carrying out tasks in respect of the individual components within a single graph [108]. Graph classification is considered to be similar to conventional supervised learning as each graph may be considered as an i.i.d. data point with an associated label — the aim during the learning task is to learn a mapping from the graphs to the labels. Similarly, graph clustering may also be viewed as an extension of conventional unsupervised clustering. A practical example of graph classification is the task of predicting a molecule’s toxicity or solubility based on the molecule’s structure which is expressed by means of a graph-based representation [98].

#### 3.5.2 Graph embeddings

As alluded to earlier, graph embeddings represent a transformation of graph data into a lower-dimensional continuous vector representation whilst maintaining the essential structural and feature-based properties that are embedded within the graph — similar vertices (or edges) in the original graph space should therefore also be “close” (or similar) in the embedding space. The

general task of constructing a graph embedding is defined as follows. Given a graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  and a predefined embedding dimension  $d$  (where  $d \ll |\mathcal{V}|$ ), the task of constructing a graph embedding involves mapping the vertices and edges into a  $d$ -dimensional space  $\mathbb{R}^d$  — formally, the aim is to approximate a function  $f : v_i \mapsto \mathbb{R}^d$ , according to which each vertex  $v_i \in \mathcal{V}$  is mapped to a real-valued vector, denoted by  $f(v_i) = z_{v_i}$ . [52]. These graph embeddings may then be applied to GRL tasks in order to analyse and derive insights from complex graph-structured data.

### The encoder-decoder perspective

The *encoder-decoder* framework proposed by Hamilton *et al.* [109] forms the basis of the graph embedding discussion presented hereafter. This framework facilitates a standardised interpretation and assimilation of the diverse approaches towards constructing graph embeddings — it unifies the relevant terminology, concepts, and notations. The encoder-decoder framework disaggregates the GRL problem into two key functional components, namely: An *encoder function* which maps each vertex in the graph to an embedding (*i.e.* the functional mapping accomplished by  $f$ ) and a *decoder function* which transforms the constructed embedding into the original (structural and feature-based) properties [108]. A simple example of a vertex embedding is presented graphically in Figure 3.4.

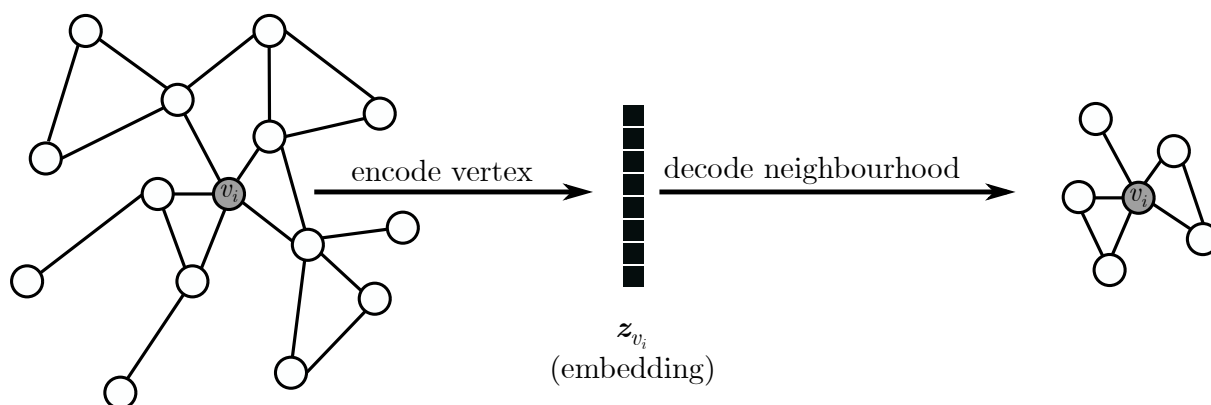


FIGURE 3.4: A simple graphical illustration of the encoder-decoder approach (adapted from Hamilton [108]). The vertex  $v_i$  is therefore encoded as the embedding  $z_{v_i}$  by means of the function  $f$ , which is followed by the decoding of this embedding so as to reconstruct the original local neighbourhood of vertex  $v_i$ .

Let ENC denote the mapping function  $f$  defined earlier, *i.e.*

$$\text{ENC} : v_i \mapsto \mathbb{R}^d.$$

Let DEC denote a *pairwise decoder* which may be expressed as

$$\text{DEC} : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R},$$

which approximates a (non-negative) similarity measure between pairs of vertices [108]. For example, a pairwise decoder may be employed to predict whether two vertices are neighbours in a graph. Consider the pair of vertex embeddings  $z_{v_i}$  and  $z_{v_j}$ , corresponding to distinct vertices  $v_i$  and  $v_j$  [109]. The main aim of the encoder-decoder approach is typically to minimise the reconstruction loss (or error) which may be defined as

$$\text{DEC}(\text{ENC}(v_i), \text{ENC}(v_j)) = \text{DEC}(z_{v_i}, z_{v_j}) \approx s(v_i, v_j), \quad (3.4)$$

where  $s$  denotes some graph-based similarity measure between two vertices of the graph [109]. For example, one may define  $s(v_i, v_j) \triangleq \mathbf{A}_{v_i, v_j}$ , according to which nodes are assigned a similarity score of one if they are adjacent, or zero otherwise. According to standard practice in the literature [109], the objective of minimising (3.4) may be formulated as the minimisation of an empirical reconstruction loss, denoted by  $L$ , in respect of a set of training vertex-pairs, denoted by  $\mathcal{D}$ , yielding

$$L = \sum_{(v_i, v_j) \in \mathcal{D}} \ell(\text{DEC}(\mathbf{z}_{v_i}, \mathbf{z}_{v_j}), s(v_i, v_j)), \quad (3.5)$$

where  $\ell : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$  is (typically) a differentiable loss function that measures the dissimilarity between the decoded similarity score of  $\text{DEC}(\mathbf{z}_{v_i}, \mathbf{z}_{v_j})$  and the “ground truth” similarity score of  $s(v_i, v_j)$ . The method of *stochastic gradient descent* is often employed to minimise the loss defined in (3.5) [109].

A majority of embedding algorithms employ a so-called *shallow* embedding approach which may be partitioned into two categories, namely *factorisation-based* approaches and *random walk* approaches. A discussion on each approach is presented hereafter.

### Factorisation-based approaches

Matrix factorisation is a technique employed to decompose some factorisable matrix (in this context, an adjacency matrix) into the product of two lower-rank approximations, the aim of which is to abstract the properties of the original graph from which potentially informative patterns and relationships may be inferred. Consider the adjacency matrix  $\mathbf{A}$  (which comprises  $n$  columns and  $n$  rows), matrix factorisation involves finding the matrices  $\mathbf{J} \in \mathbb{R}^{n \times d}$  and  $\mathbf{H} \in \mathbb{R}^{n \times d}$ , where  $d$  denotes the embedding dimension, from which the adjacency matrix ought to be reconstructed. In the case of *non-negative matrix factorisation*,  $\mathbf{J}$  and  $\mathbf{H}$  are randomly initialised and iteratively updated so that  $\mathbf{A} \approx \mathbf{J}\mathbf{H}^\top$  [316]. Depending on the factorisation approach adopted, each embedding  $\mathbf{z}_{v_i}$  may be derived from the entries of  $\mathbf{J}$  and/or  $\mathbf{H}$ , for all  $i \in \{1, \dots, n\}$ . For example, simply calculating the arithmetic mean of the corresponding rows of  $\mathbf{J}$  and  $\mathbf{H}$ . More advanced approaches include weighted averages or non-linear transformations. The task of decoding graph properties (*e.g.* local neighbourhood structure) from a vertex’s embedding may be performed by means of different approaches, two of which are discussed hereafter, namely: *Laplacian eigenmaps* [25] and *inner-product methods* [108].

### Laplacian eigenmaps

In the case of Laplacian<sup>6</sup> eigenmaps, the decoder function is expressed as

$$\text{DEC}(\mathbf{z}_{v_i}, \mathbf{z}_{v_j}) = \|\mathbf{z}_{v_i} - \mathbf{z}_{v_j}\|_2^2,$$

which corresponds to the Euclidean distance between the two vertex embeddings  $\mathbf{z}_{v_i}$  and  $\mathbf{z}_{v_j}$ . The corresponding loss function may be expressed as

$$L = \sum_{(v_i, v_j) \in \mathcal{D}} \text{DEC}(\mathbf{z}_{v_i}, \mathbf{z}_{v_j}) s(v_i, v_j), \quad (3.6)$$

according to which the decoder output and similarity score for vertex-pair  $v_i$  and  $v_j$  are simply multiplied. The intuition is that the minimisation of (3.6) corresponds to learning a decoder function that ensures similar vertices have embeddings that are close (in Euclidean space) [109].

<sup>6</sup>The graph Laplacian matrix is derived from the adjacency matrix and the degree matrix (*i.e.* diagonal matrix in which each degree entry corresponds to the summation of the relevant row entries in the adjacency matrix) [109].

### Inner-product methods

Inner-product methods are based on a pairwise inner-product decoder which may be expressed as

$$\text{DEC}(\mathbf{z}_{v_i}, \mathbf{z}_{v_j}) = \mathbf{z}_{v_i}^\top \mathbf{z}_{v_j},$$

according to which embedding similarity is computed by means of the dot product [108]. Popular examples of vertex embedding algorithms that adopt the inner-product method include the *graph factorisation* algorithm [7], *GraRep* [42], and *HOPE* [216]. Each of these approaches employs an inner-product decoder and a mean-squared error loss function which may be expressed as

$$L = \frac{1}{|\mathcal{D}|} \sum_{(v_i, v_j) \in \mathcal{D}} \|\text{DEC}(\mathbf{z}_{v_i}, \mathbf{z}_{v_j}) - s(v_i, v_j)\|_2^2.$$

The main differences between these approaches relate to the manner in which the similarity measure is employed. For example, the graph factorisation algorithm expresses the similarity between two vertices  $v_i$  and  $v_j$  in respect of the adjacency matrix, *i.e.*  $s(v_i, v_j) \triangleq \mathbf{A}_{v_i, v_j}$  [7], whereas GraRep considers various powers of the adjacency matrix, *e.g.*  $s(v_i, v_j) \triangleq \mathbf{A}_{v_i, v_j}^2$  [42]. Moreover, these methods are referred to as matrix-factorisation approaches ascribed to the fact that their loss functions can be minimised using factorisation algorithms such as non-negative matrix factorisation (discussed above) and *singular value decomposition* [108].

### Random walk approaches

Conceptually, these approaches approximate vertex embeddings by performing random walks (*i.e.* finite, alternating sequences) over the graph and relating embedding similarity based on similar random walk lengths [108]. Unlike factorisation-based approaches (which mostly employ deterministic vertex similarity measures), random walk methods utilise stochasticity towards measuring vertex similarity which has showcased empirical performance improvements [102]. The discussion pertaining to random walk approaches is structured according to the following popular approaches, namely DEEPWALK [222] and *node2vec* [105].

Perozzi *et al.* [222] proposed the so-called DEEPWALK approach which is based on a generalisation of language modelling<sup>7</sup>, the aim of which is to estimate the probability of visiting a vertex  $v_i$  given all previous vertices that have already been visited in the random walk. DEEPWALK assumes that the local neighbourhood of a vertex encapsulates sufficient information towards generating a meaningful embedding. Given a (partial) random walk  $v_1 v_2 v_3 \dots v_{i-1}$ , the probability of visiting vertex  $v_i$  next may be expressed as

$$P(v_i | (v_1 v_2 \dots v_{i-1})). \quad (3.7)$$

In the case of DEEPWALK, the transition probability of vertex  $v_i$  is expressed as a function of the vertex embeddings preceding it in the walk (*i.e.*  $v_1 v_2 \dots v_{i-1}$ ). Accordingly, expression (3.7) can be transformed into

$$P\left(v_i | (\text{ENC}(v_1) \text{ ENC}(v_2) \dots \text{ ENC}(v_{i-1}))\right).$$

<sup>7</sup>Language modelling is the computational task of predicting a sequence of tokens (*e.g.* words or characters) based on preceding ones, using statistical and/or ML approaches. The approximation of a language's probability distribution enables applications such as text generation, translation, and speech recognition [295].



The task of constructing an appropriate embedding for vertex  $v_i$  therefore involves finding the function ENC that maximises

$$\log P(v_{i-w}, \dots, v_{i-1}, v_{i+1}, \dots, v_{i+w} | \text{ENC}(v_i)), \quad (3.8)$$

where  $w$  denotes the window size, *i.e.* the distance traversed by the random walk in respect of vertex  $v_i$  [222]. This optimisation problem ought to be extended in terms of calculating (3.8) for multiple random walks and for all vertices within the graph.

Another embedding approach is node2vec which was proposed by Grover *et al.* [105]. Similar to DEEPWALK, node2vec also seeks to find an appropriate functional mapping by maximising a log-probability function that ensures that similar (*i.e.* neighbouring) vertices have embeddings that are “close” in the embedding space [102]. The main difference relates to the manner in which node2vec conducts neighbourhood sampling. As mentioned earlier, vertices within a network can be similar due to factors such as homophily or structural equivalence. In the case of GRL, homophily asserts that vertices that share a large number of edges with one another belong to the same community or cluster and should therefore have similar embeddings. According to structural equivalence, on the other hand, vertices that share similar structural properties in terms of their local neighbourhood should also have similar embeddings. Two different search (*i.e.* traversal) strategies for sampling neighbourhoods may be adopted, *i.e.* *depth first search*<sup>8</sup> (DFS) and *breadth first search*<sup>9</sup> (BFS). In particular, BFS results in embeddings that correspond closely to structural equivalence due to the search procedure being restricted to nearby vertices, therefore sampling is locally biased. Conversely, DFS explores local structures in a more exhaustive manner, obtaining a comprehensive view of a vertex’s neighbourhood which is essential for identifying homophily within a network [105]. In order to better assimilate the entire network structure, node2vec employs a flexible and controllable random walk approach capable of exploring neighbourhoods through DFS and BFS. Consider a source (initial) vertex  $s$  and a random walk of fixed length  $l$ . Let  $v_i$  denote the  $i$ -th vertex in the walk, starting with  $s = v_0$ . Vertex  $v_i$  is generated based on the following distribution

$$P(v_i | v_{i-1} = x) = \begin{cases} \frac{\pi_{xv_i}}{R} & \text{if } (x, v_i) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases},$$

where  $\pi_{xv_i}$  denotes the (non-normalised) transition probability between vertices  $v_{i-1} = x$  and  $v_i$ , while  $R$  denotes some normalising constant. A search bias, denoted by  $\alpha_{pq}(v_i, v_j)$ , along with the *return* parameter, denoted by  $p$ , and the *in-out* parameter, denoted by  $q$ , are employed to guide the random walk. In particular, parameter  $p$  determines the probability of immediately revisiting a vertex in the walk, while the parameter  $q$  determines the manner in which neighbouring vertices are explored — *i.e.* the search bias  $\alpha_{pq}(v_i, v_j)$ , together with parameters  $p$  and  $q$ , controls the rate at which the walk explores new vertices and moves away from the neighbourhood of the starting vertex  $s$ .

As presented in Figure 3.5, consider the vertex  $x$ , and its neighbours  $t, v_1, v_2$  and  $v_3$ , together with a random walk that has (thus far) traversed the edge  $\{t, x\}$ . The current vertex is therefore  $x$  and the previous vertex is  $t$  [105]. The next step in the walk is determined by evaluating the

<sup>8</sup>DFS involves traversing a graph by selecting a neighbour of some chosen starting vertex  $s$ , and then traversing as far as possible along that path before backtracking [121]. When a so-called *dead-end* vertex  $v$  is reached, the DFS algorithm backtracks to explore the unexplored edges leaving the vertex from which  $v$  was discovered. This process is repeated until all vertices reachable from  $s$  have been discovered.

<sup>9</sup>BFS involves traversing a graph structure by first exploring all (direct) neighbours of a chosen vertex  $v$ , before exploring all neighbours positioned two edges away from  $v$ , and so on [121]. Unlike DFS, the BFS algorithm progresses without the need of backtracking [117].

transition probabilities  $\pi_{xv_i}$  where  $v_i \in \{t, v_1, v_2, v_3\}$ . The non-normalised transition probability may be expressed as

$$\pi_{xv_i} = \alpha_{pq}(t, v_i)w_{xv_i},$$

where  $w_{xv_i}$  denotes the weighting of edge  $(x, v_i)$ <sup>10</sup> and

$$\alpha_{pq}(t, v_i) = \begin{cases} \frac{1}{p} & \text{if } d_{tv_i} = 0 \\ 1, & \text{if } d_{tv_i} = 1, \\ \frac{1}{q} & \text{if } d_{tv_i} = 2 \end{cases} \quad (3.9)$$

where  $d_{tv_i}$  denotes the shortest path between vertices  $t$  and  $v_i$ . A large value for  $p$ , *i.e.*  $p > \max(q, 1)$ , corresponds to a smaller probability of sampling a previous vertex during subsequent steps, subject to the assumption that the next vertex in the walk does not have another neighbour. If the value for  $p$  is small, *i.e.*  $p < \min(q, 1)$ , the walk is likely to backtrack which could constrain the walk in proximity to the starting vertex  $s$ . The in-out parameter  $q$  enables the search to differentiate between *inward* and *outward* vertices. Consider the example depicted in Figure 3.5. In the case of large  $q$  values, the random walk is biased towards vertices that are located close to vertex  $t$ , akin to BFS, whereas in the case of small  $q$  values, *i.e.*  $q < 1$ , the search is biased towards vertices located further away from vertex  $t$ , akin to DFS [105].

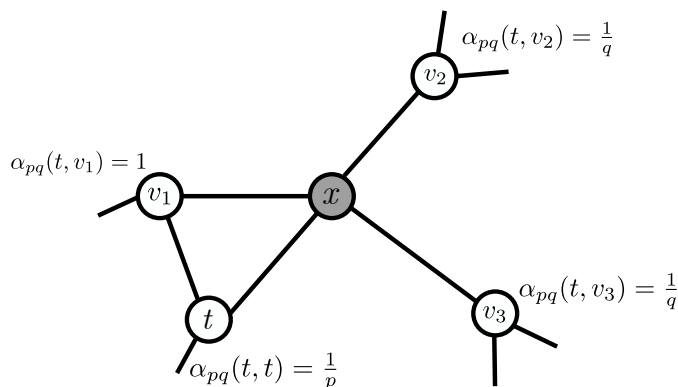


FIGURE 3.5: An example of the random walk procedure as part of the node2vec algorithm [105].

### Limitations of shallow embeddings

Despite successes in recent years [105, 108, 222], shallow embedding approaches exhibit a few key limitations. One of the main drawbacks is the lack of explicit parameter value sharing between vertices in respect of the encoder — a distinct embedding vector is created for each vertex, rendering the learning process computationally demanding. Furthermore, a large number of shallow embedding approaches do not incorporate vertex features therefore disregarding information that may potentially enrich the encoding process. Finally, shallow embedding approaches do not generalise to unseen data instances — its application is therefore limited to descriptive analyses and not predictive analyses [108].

### 3.5.3 Multi-relational data

A graph-based representation of multi-relational data, as defined in §2.1.5, comprises edges of the form  $\{v_i, \tau, v_j\} \in \mathcal{E}$ , where  $\tau \in \mathcal{R}$  denotes the type of edge joining vertices  $v_i$  and  $v_j$ . This

<sup>10</sup>For an unweighted graph  $w_{xv_i} = 1$ .

representation is especially applicable to KGs which convey semantics by means of different relationships [108, 276]. The presence of multiple edge types within KGs presents, however, a computational challenge. Towards mitigating this pitfall, the decoder is modified so as to facilitate operation on both pair vertex embeddings and the associated relation type which may be expressed as

$$\text{DEC} : \mathbb{R}^d \times \mathcal{R} \times \mathbb{R}^d \mapsto \mathbb{R},$$

according to which  $\text{DEC}(\mathbf{z}_{v_i}, \tau, \mathbf{z}_{v_j})$  may be employed towards determining the likelihood of the edge  $\{\mathbf{z}_{v_i}, \tau, \mathbf{z}_{v_j}\}$  being present within the graph [108]. Nickel *et al.* [215] proposed a generic modelling approach, called RESCAL, according to which the decoder is expressed as

$$\text{DEC}(\mathbf{z}_{v_i}, \tau, \mathbf{z}_{v_j}) = \mathbf{z}_{v_i} \mathbf{R}_\tau \mathbf{z}_{v_j}^\top,$$

where  $\mathbf{R}_\tau \in \mathbb{R}^{d \times d}$  is the relation embedding matrix in respect of relation  $\tau \in \mathcal{R}$  [215]. The remainder of the discussion on multi-relational decoders is partitioned into two categories, namely: *Translational decoders* and *multi-linear dot products*.

### Translational decoders

Translational-based models represent the edges in a graph as translations<sup>11</sup> in the embedding space [108]. A translational decoder approach translates the *head* vertex embedding  $\mathbf{z}_{v_i}$  according to the relation embedding which is followed by the computation of the distance between the translated head vertex embedding and tail vertex embedding  $\mathbf{z}_{v_j}$ . This type of decoder approach was first introduced in the TransE model proposed by Bordes *et al.* [33] and is shown in Figure 3.6. The decoder employed in the TransE model is defined as

$$\text{DEC}(\mathbf{z}_{v_i}, \tau, \mathbf{z}_{v_j}) = -\|\mathbf{z}_{v_i} + \mathbf{r}_\tau - \mathbf{z}_{v_j}\|,$$

where  $\mathbf{r}_\tau \in \mathbb{R}^d$  denotes the embedding for relation  $\tau$ .

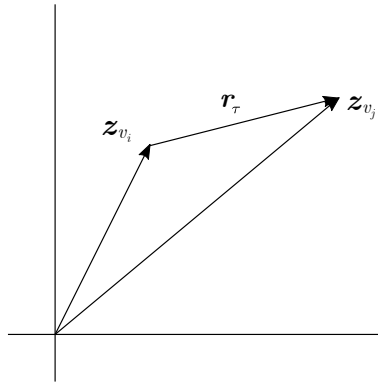


FIGURE 3.6: A simple graphical illustration of the TransE model proposed by Bordes *et al.* [33] which represents a relation by a translation vector  $\mathbf{r}_\tau$  so that the pair of embedded vertices  $\mathbf{z}_{v_i}$  and  $\mathbf{z}_{v_j}$  can be connected by  $\mathbf{r}_\tau$  (adapted from [294]).

The TransE model is arguably simple and, consequently, exhibits limitations. Appropriately, various extensions have been proposed, namely: TransR [169], TransH [294], and TransA [301]. These extensions perform a linear transformation of the entity (*i.e.* vertex) embeddings into a

<sup>11</sup>A translation is a geometric transformation that moves every point (of some object) a fixed distance in a specified direction [59].

relation-specific space prior to translation [276]. For example, the TransH model projects the vertex embeddings onto a learnable relation-specific hyperplane, defined by the normal vector  $\mathbf{w}_r$ , before translation [108]. The decoder employed in the TransH model is defined as

$$\text{DEC}(\mathbf{z}_{v_i}, \tau, \mathbf{z}_{v_j}) = -\|(\mathbf{z}_{v_i} - \mathbf{w}_r^\top \mathbf{z}_{v_i} \mathbf{w}_r) + \mathbf{r}_\tau - (\mathbf{z}_{v_j} - \mathbf{w}_r^\top \mathbf{z}_{v_j} \mathbf{w}_r)\|.$$

### Multi-linear dot products

Multi-linear dot products define decoders by means of generalising the dot-product decoder from simple graphs. The first (and arguably most simple) approach is called DistMult which was proposed by Yang *et al.* [305]. It may be expressed as as

$$\text{DEC}(\mathbf{z}_{v_i}, \tau, \mathbf{z}_{v_j}) = \mathbf{z}_{v_i} \odot \mathbf{r}_\tau \odot \mathbf{z}_{v_j}, \quad (3.10)$$

where  $\odot$  denotes the Hadamard product (*i.e.* element-wise multiplication). DistMult can, however, only represent symmetric (undirected) relations which presents a notable limitation with respect to its application to real-world graphs. Trouillon *et al.* [277] proposed the ComplEx model which enhances the DistMult model by employing complex-valued embeddings. The ComplEx decoder may be expressed as

$$\text{DEC}(\mathbf{z}_{v_i}, \tau, \mathbf{z}_{v_j}) = \Re(\mathbf{z}_{v_i} \odot \mathbf{r}_\tau \odot \bar{\mathbf{z}}_{v_j}),$$

where  $\{\mathbf{z}_{v_i}, \mathbf{r}_\tau, \mathbf{z}_{v_j}\} \in \mathbb{C}^d$  are complex-valued embeddings,  $\Re$  denotes the real part of these complex-valued embeddings, and  $\bar{\mathbf{z}}_{v_j}$  denotes the complex conjugate of the tail embedding. The complex conjugate of the tail embedding enables the abstraction of asymmetric relations [108].

### 3.5.4 Graph neural networks

GNNs are deep learning based-models that can be applied in a computationally effective manner to graph structured data [325]. Unlike the aforementioned shallow embedding approaches that approximate a low-dimensional embedding based on arguably rudimentary features, GNNs learn representations of graphs by combining graph and feature-based properties by means of techniques such as feature propagation and aggregation [153]. GNNs are a nascent approach which have been applied to a variety of real-world use cases, *e.g.* traffic prediction [312], financial fraud detection [288], disease prediction [291], and user behaviour analysis [144], to name but a few. This section is devoted to a detailed discussion on GNNs in respect of their conceptual working and functional components.

#### GNN modelling pipeline

In the paper titled “Graph neural networks: A review of methods and applications”, Zhou *et al.* [325] proposed a generalised modelling pipeline for the design of GNNs. The pipeline comprises four primary steps, namely: (1) Identify the graph structure, (2) specify the graph type and scale, (3) design the loss function, and (4) construct the model by means of computational modules. A graphical illustration of this generic pipeline may be observed in Figure 3.7.

The first step of the pipeline involves identifying the inherent structure of the graph [325]. The main consideration in this respect relates to the nature of the network<sup>12</sup> problem to be

<sup>12</sup>In this context, network pertains to the real-world problem itself, whereas graph represents the mathematical abstraction thereof.

modelled by the GNN. More specifically, there are two general network problems, namely: (1) Networks whose innate structure stems from some physical relationship (*e.g.* physical layout of a road network or molecular structure of some physical material) or (2) networks that are not physically bound (*e.g.* social networks or hierarchical linguistic structures) in which relations are intangible.

In the case of a graph’s type, different facets constitute the description of a type of graph under consideration, namely: Directed/undirected graphs (§2.1.2), homogeneous/heterogeneous graphs (§2.1.5), and static/dynamic graphs. Dynamic graphs comprise input features or topologies that change over time [43]. An example of a graph type can therefore be a dynamic directed heterogeneous graph, such as a citation graph that connects authors and articles by means of a “cites” relationship. This graph may therefore evolve over time based on new citation instances. The scale (*i.e.* in respect of order and size) of a graph is a relative notion as it is context-specific [325].

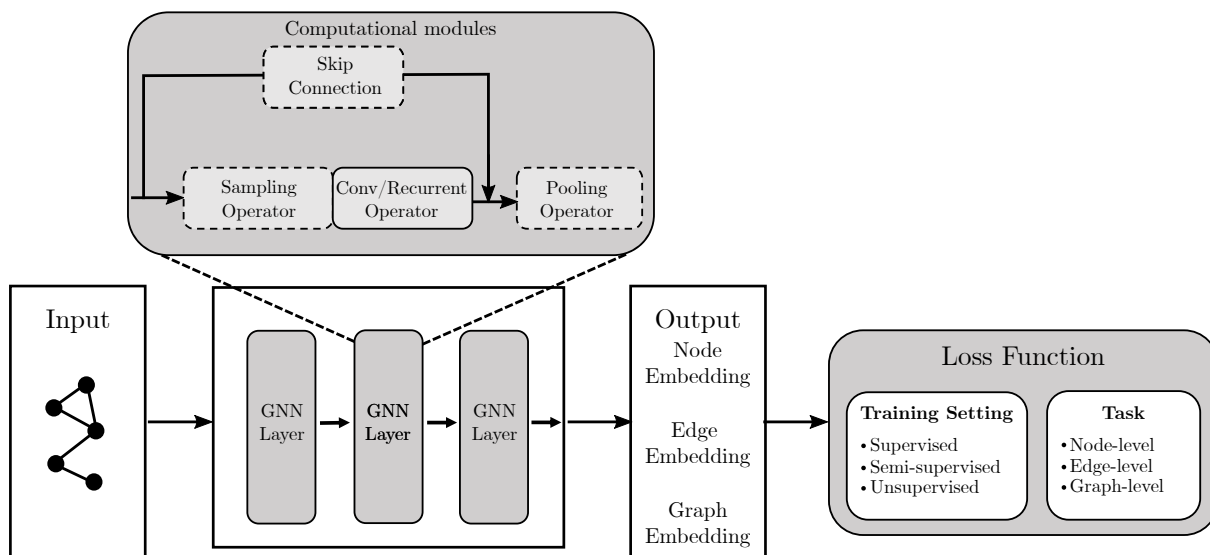


FIGURE 3.7: A generic modelling pipeline for the design of GNNs (adapted from [325]).

The loss function employed in GNN tasks depends on the type of task to be accomplished as well as the learning paradigm. Graph learning tasks can be carried out on three levels of abstraction, *i.e.* vertex-, edge-, and graph-level [108], as mentioned in §3.5.1. For example, node classification involves categorising vertices into different classes by assigning labels to them. Edge-level tasks include edge classification and link (relation) prediction. Lastly, graph-level tasks operate on the entire graph, examples of which include graph classification, graph regression, and clustering. Graph learning tasks may be further categorised based on conventional ML paradigms, *i.e.* supervised, semi-supervised, and unsupervised, as discussed in §3.4.1.

Finally, the GNN is to be constructed by means of different computational modules [325]. For example, so-called propagation modules are used to disseminate information between the vertices of the graph which induces the graph’s representational capabilities in respect of the connected vertices and resulting structure. The propagation module comprises different *convolutional* and *recurrent operators* which aggregate information from a vertex’s neighbours, while *skip connection* operators facilitate the processing of information from prior vertex representations for downstream processing. A *sampling* module operator is employed in the case of large graphs so as to mitigate the computational burden of propagation, whereas the pooling module is employed to extract information from vertices in order to generate high-level representations of the vertices.

### Neural message passing

A fundamental principle that underpins GNNs is the notion of *neural message passing*, according to which embeddings are propagated and updated between vertices algorithmically [98]. Consider a graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , with vertex-feature matrix  $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$  which contains the vectors  $\mathbf{x}_{v_i} \in \mathbb{R}^d$  that are employed when generating vertex embeddings  $\mathbf{z}_{v_i}$ , for all  $v_i \in \mathcal{V}$ . Vertex features may encompass attributes such as, for example, the gene expression levels in biological networks, the geographic coordinates in transportation networks, and the historical transaction behaviour in financial networks. During each iteration<sup>13</sup>, (let  $k$  denote the iteration counter), a so-called *hidden embedding* of message-passing  $\mathbf{h}_{v_i}^{(k)}$ , corresponding to each vertex  $v_i \in \mathcal{V}$ , is updated according to information collected from its neighbourhood  $\mathcal{N}(v_i)$ , expressed as

$$\begin{aligned} \mathbf{h}_{v_i}^{(k)} &= \text{UPDATE} \left( \mathbf{h}_{v_i}^{(k-1)}, \text{AGGREGATE} \left( \left\{ \mathbf{h}_{v_n}^{(k-1)}, \text{ for all } v_n \in \mathcal{N}(v_i) \right\} \right) \right) \\ &= \text{UPDATE} \left( \mathbf{h}_{v_i}^{(k-1)}, \mathbf{m}_{\mathcal{N}(v_i)}^{(k)} \right), \end{aligned} \quad (3.11)$$

where **UPDATE** and **AGGREGATE** are some differentiable functions, and  $\mathbf{m}_{\mathcal{N}(v_i)}^{(k)}$  denotes the *message* that is aggregated from the neighbourhood structure of vertex  $v_i$  in respect of the previous iteration (*i.e.*  $k - 1$ ).

During each iteration, the **AGGREGATE** function receives as input a set of embeddings in respect of the neighbourhood of  $v_i$  and subsequently generates a message  $\mathbf{m}_{\mathcal{N}(v_i)}^{(k)}$ , *i.e.* based on the information collected from the neighbourhood. The **UPDATE** function then combines the message  $\mathbf{m}_{\mathcal{N}(v_i)}^{(k)}$  with the previous hidden embedding  $\mathbf{h}_{v_i}^{(k-1)}$ , in order to generate the new hidden embedding  $\mathbf{h}_{v_i}^{(k)}$ . The (initial) embeddings at iteration  $k = 0$  are initialised as the input features of the vertices, yielding

$$\mathbf{h}_{v_i}^{(0)} = \mathbf{x}_{v_i}, \text{ for all } v_i \in \mathcal{V}.$$

After performing a total of  $K$  iterations, the vertex embeddings may be expressed as

$$\mathbf{z}_{v_i} = \mathbf{h}_{v_i}^{(K)}, \text{ for all } v_i \in \mathcal{V}.$$

A simple visual representation of the manner in which neural message passing conceptually operates is depicted in Figure 3.8. A simple graph is illustrated to which a two-layer message passing GNN model is applied, in which messages from target vertex A’s neighbourhood — *i.e.* vertices B, C, and D — are aggregated. The messages that emanate from these neighbours are, in turn, obtained by aggregating information obtained from their respective neighbourhoods [108].

As the algorithmic procedure progresses, each vertex embedding contains additional information, *i.e.* from vertices located “further” away in the graph. Accordingly, in the first iteration, each vertex embedding contains information from its immediate neighbourhood which can also be expressed as its 1-hop neighbourhood. Similarly, in the second iteration, each vertex embedding contains information from the immediate neighbourhood structure of its immediate neighbours, *i.e.* its 2-hop neighbourhood. Therefore, after  $k$  iterations each vertex embedding contains information pertaining to its  $k$ -hop neighbourhood. The total number of iterations, as denoted by  $K$ , therefore represents the *depth* of the search. The information extracted from the neighbourhood structures may also include, for example, traditional metrics such as the degree of the vertices within the neighbourhood [108].

<sup>13</sup>In the context of GNNs, the different iterations of message passing are referred to as the “layers” of the GNN [108].

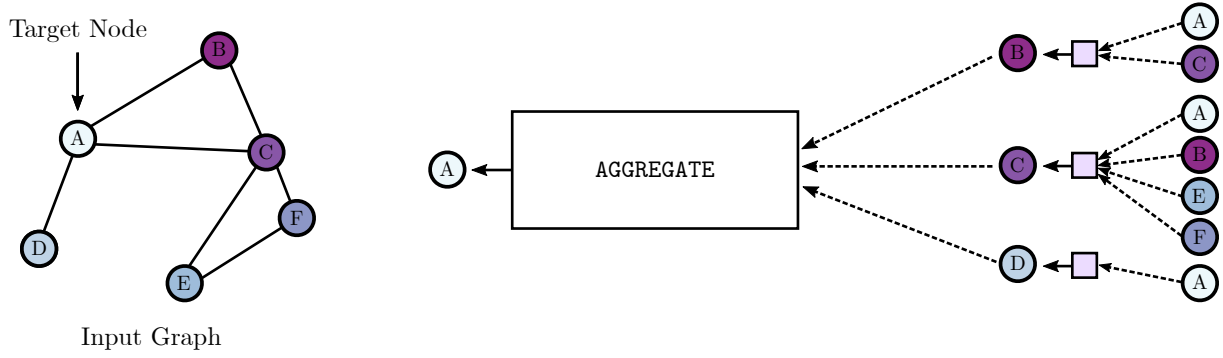


FIGURE 3.8: A simple example depicting the neural message passing in the case of  $k = 2$ , as performed in respect of target vertex  $A$  (adapted from [108]).

A standard (*i.e.* basic) implementation of a GNN is now elucidated which includes mathematical definitions for the UPDATE and AGGREGATE functions in (3.11). Standard GNN message passing may be defined as

$$\mathbf{h}_{v_i}^{(k)} = \sigma \left( \mathbf{W}_{\text{self}}^{(k)} \mathbf{h}_{v_i}^{(k-1)} + \mathbf{W}_{\text{neigh}}^{(k)} \sum_{v_n \in \mathcal{N}(v_i)} \mathbf{h}_{v_n}^{(k-1)} + \mathbf{b}^{(k)} \right),$$

where  $\mathbf{W}_{\text{self}}^{(k)}$  and  $\mathbf{W}_{\text{neigh}}^{(k)} \in \mathbb{R}^{d \times d}$  denote trainable parameter matrices corresponding to the vertex  $v_i$  and its neighbourhood, respectively. Furthermore,  $\sigma$  denotes an element-wise non-linearity such as tanh or ReLU, and  $\mathbf{b}^{(k)} \in \mathbb{R}$  denotes the bias term, which is often omitted for the purpose of notational simplicity [108]. The AGGREGATE and UPDATE functions of (3.11) may therefore be replaced by

$$\text{AGGREGATE} : \mathbf{m}_{\mathcal{N}(v_i)}^{(k)} = \sum_{v_n \in \mathcal{N}(v_i)} \mathbf{h}_{v_n}^{(k-1)} \quad (3.12)$$

and

$$\text{UPDATE} \left( \mathbf{h}_{v_i}^{(k-1)}, \mathbf{m}_{\mathcal{N}(v_i)}^{(k)} \right) = \sigma \left( \mathbf{W}_{\text{self}}^{(k)} \mathbf{h}_{v_i}^{(k-1)} + \mathbf{W}_{\text{neigh}}^{(k)} \mathbf{m}_{\mathcal{N}(v_i)}^{(k)} \right), \quad (3.13)$$

respectively. The matrices  $\mathbf{W}_{\text{self}}^{(k)}$  and  $\mathbf{W}_{\text{neigh}}^{(k)}$  are henceforth denoted as  $\mathbf{W}^{(k)}$  for notational simplicity. Towards simplifying the neural message passing approach, it is common to omit the UPDATE step by inserting so-called self-loops to the input graph [108]. This self-loop GNN approach may be expressed as

$$\mathbf{h}_{v_i}^{(k)} = \text{AGGREGATE} \left( \left\{ \mathbf{h}_{v_n}^{(k-1)}, \text{ for all } v_n \in \mathcal{N}_{\mathcal{G}}[v_i] \right\} \right).$$

Accordingly, the aggregation is performed over the closed neighbourhood of  $v_i$ . By employing this approach, there is no need to define an UPDATE function explicitly as the update is performed implicitly through the AGGREGATE function. This approach can help mitigate overfitting but also limits the expressibility of the GNN — no explicit distinction is made between the vertex itself and its neighbours [108].

### Neighbourhood aggregation approaches

One approach towards enhancing the standard GNN model is to consider different aggregation functions. The simplest aggregation operation corresponds to the summation of the neighbouring

vertex embeddings (*i.e.* vector addition), as defined in (3.12). This method is, however, prone to numerical instability and is sensitive to vertices that have large degrees. For example, consider vertex  $v_i$  and one of its neighbours  $v'_i$ . If  $|\mathcal{N}(v_i)| \gg |\mathcal{N}(v'_i)|$ , then it is reasonable to expect that

$$\left\| \sum_{v_n \in \mathcal{N}(v_i)} \mathbf{h}_{v_n} \right\| \gg \left\| \sum_{v'_n \in \mathcal{N}(v'_i)} \mathbf{h}_{v'_n} \right\|$$

for any appropriate vector norm  $\|\cdot\|$ , *e.g.* Euclidean norm. These large differences may result in numerical instabilities during the optimisation (training) procedure. A variety of mitigating strategies have been reported in the literature [108]. A simple alternative approach involves computing the average neighbourhood vertex embedding which may be expressed as

$$\mathbf{m}_{\mathcal{N}(v_i)}^{(k)} = \sum_{v_n \in \mathcal{N}(v_i)} \frac{\mathbf{h}_{v_n}^{(k-1)}}{|\mathcal{N}(v_i)|}.$$

Furthermore, Kipf *et al.* [147] employed symmetric normalisation, expressed as

$$\mathbf{m}_{\mathcal{N}(v_i)}^{(k)} = \sum_{v_n \in \mathcal{N}(v_i)} \frac{\mathbf{h}_{v_n}^{(k-1)}}{\sqrt{|\mathcal{N}(v_i)| |\mathcal{N}(v_n)|}},$$

which was empirically shown to improve convergence. *Graph convolution networks* (GCNs) [147] are regarded as a favourable baseline<sup>14</sup> GNN model which employs both symmetric-normalised aggregation and self-loops. Its message passing function may be expressed compactly as

$$\mathbf{h}_{v_i}^{(k)} = \sigma \left( \mathbf{W}^{(k)} \sum_{v_n \in \mathcal{N}[v_i]} \frac{\mathbf{h}_{v_n}^{(k-1)}}{\sqrt{|\mathcal{N}(v_i)| |\mathcal{N}(v_n)|}} \right).$$

Despite some improvements to algorithmic convergence, normalisation causes unfavourable information loss due to the difficulty associated with distinguishing between vertices of different degrees (and other graph structural features). Normalisation is therefore applied on an *ad hoc* basis and is typically deemed more favourable in respect of tasks for which feature information is prioritised over graph structural information [108].

## Update methods

A common issue associated with training GNNs is *over-smoothing* which refers to the undesired phenomenon of each vertex embedding converging to a similar expression after several iterations of message passing — consequently, vertex-specific information is lost. This issue is more prevalent in standard GNN models as well as models employing the self-loop update approach. A solution to this predicament involves so-called *generalised update methods*.

Upon considering the aforementioned definition of over-smoothing, it is reasonable to assume that over-smoothing can manifest when the information aggregated from a vertex's neighbours tends to dominate the updated vertex representation. Accordingly, the updated vertex representation  $\mathbf{h}_{v_i}^{(k)}$  contains excessive (or a disproportionate extent of) information from the aggregated messages of the neighbouring vertices. A potential solution involves employing vector concatenations or skip connections, the aim of which is to maintain information from previous iterations of

<sup>14</sup>An established method against which performance may be compared.



message passing explicitly. A simple skip connection update that preserves previous vertex-level information *via* concatenation may be defined as

$$\text{UPDATE}_{\text{concat}} \left( \mathbf{h}_{v_i}^{(k-1)}, \mathbf{m}_{\mathcal{N}(v_i)}^{(k)} \right) = \left[ \text{UPDATE}_{\text{base}} \left( \mathbf{h}_{v_i}^{(k-1)}, \mathbf{m}_{\mathcal{N}(v_i)}^{(k)} \right) \oplus \mathbf{h}_{v_i}^{(k-1)} \right],$$

where  $\text{UPDATE}_{\text{base}}$  is computed by means of (3.13) and  $\oplus$  denotes the concatenation operation, according to which the output of the base update function is concatenated with the vertex’s representation obtained in the previous iteration. The aim is to segregate the information, *i.e.* to separate the neighbouring vertices  $\mathbf{m}_{\mathcal{N}(v_i)}^{(k)}$  from the current vertex representation  $\mathbf{h}_{v_i}^{(k-1)}$ .

In addition to skip-connection’s, *gating methods* are also proposed — their origin stems from recurrent neural networks [73]. Popular gating methods include the gated recurrent units [54] and *long short-term memory* (LSTM) units [255].

### Multi-relational GNNs

Graphs that are employed towards modelling real-world phenomena comprise multiple edge and vertex types, therefore requiring alternative strategies in order to accommodate this type of data. Early contributions in this regard employed the aforementioned GCN approach [147]. Schlichtkrull *et al.* [251] proposed the first approach aimed at handling multi-relational data in their paper titled “Modelling relational data with graph convolutional networks”. Their approach is commonly referred to as *relational graph convolutional networks* (R-GCNs) which modifies the aggregation function so as to incorporate multiple relation types by assigning a separate transformation matrix for each type of relation. The corresponding computation may be expressed as

$$\mathbf{m}_{\mathcal{N}(v_i)}^{(k)} = \sum_{\tau \in \mathcal{R}} \left( \sum_{v_n \in \mathcal{N}_\tau(v_i)} \frac{\mathbf{W}_\tau^{(k)} \mathbf{h}_{v_n}^{(k-1)}}{f_n(\mathcal{N}(v_i), v_n)} \right),$$

where  $\tau \in \mathcal{R}$  denotes the relation type and  $v_n \in \mathcal{N}_\tau$  represents the neighbours of  $v_i$  that are incident to edges of type  $\tau$ . Furthermore,  $\mathbf{W}_\tau^{(k)}$  denotes a learnable weight corresponding to relation  $\tau$  and  $f_n$  denotes a normalisation function over both the neighbourhood of the vertex  $v_i$  as well as all the neighbour  $v_n$ .

### GNN architectures

A detailed overview of popular GNN architectures is now presented so as to provide the necessary technical background in respect of these architectures. Each architecture differs with respect to the manner in which embeddings are learnt from the neighbourhood structure of a vertex. Different GNN architectures therefore represent different computational means towards extracting inferential patterns from the considered graph-based data. Furthermore, the utility of each approach varies amongst different graph problem instances which is indicative of the important notion of *performance complementarity* [108]. Appropriately, the scope in this thesis covers various popular architectural manifestations — this wide coverage facilitates robust analyses. The different GNN architectures discussed in this section include *GraphSAGE* (SAGE) [110], *graph attention networks* (GATs) [282], *GATv2* [38] and *graph transformers* (GTs) [87].

## GraphSAGE

SAGE was proposed by Hamilton *et al.* [110] as a generic inductive framework that effectively employs vertex feature information to generate vertex embeddings. SAGE therefore proposes a framework that generalises the GCN approach by employing trainable aggregation functions, which extend beyond simple convolutions. The SAGE framework includes three aggregator functions for consideration, namely: Mean aggregator, LSTM aggregator, and a pooling aggregator [110]. The aggregator functions relevant to the numerical work conducted in this thesis are the mean and max (pooling) operator, therefore the LSTM aggregator is omitted from the further discussions. The mean aggregator function represents an inductive variant of the GCN approach [147] and may be expressed as

$$\mathbf{h}_{v_i}^{(k)} \leftarrow \sigma \left( \mathbf{W}^{(k)} \text{MEAN} \left( \left\{ \mathbf{h}_{v_i}^{(k-1)} \right\} \cup \left\{ \mathbf{h}_{v_n}^{(k-1)}, \text{ for all } v_n \in \mathcal{N}(v_i) \right\} \right) \right).$$

The max aggregator, on the other hand, involves the application of an element-wise max-pooling operation in order to aggregate information across the neighbourhood set, expressed as

$$\mathbf{m}_{\mathcal{N}(v_i)}^{(k)} \leftarrow \text{MAX} \left( \left\{ \sigma \left( \mathbf{W}^{(k)} \mathbf{h}_{v_n}^{(k-1)} \right), \text{ for all } v_n \in \mathcal{N}(v_i) \right\} \right).$$

## Graph attention network

Veličković *et al.* [282] proposed a novel GNN architecture called GATs which leverages masked self-attentional<sup>15</sup> layers along with multi-head<sup>16</sup> [281] attention in order to overcome limitations of previous graph convolution-based methods, such as SAGE. The GAT architecture comprises multiple graph attention layers, each containing multiple attention heads, denoted by  $t \in \{1, 2, \dots, T\}$ , operating in parallel. With respect to each attention head, the normalised attention coefficient  $\alpha_{v_i v_n}^{(t)}$ , where  $v_n \in \mathcal{N}(v_i)$ , is computed. The attention coefficient  $\alpha_{v_i v_n}^{(t)}$  indicates the importance of the features of the neighbour  $v_n$  in respect of vertex  $v_i$  and is computed by means of the self-attention mechanism which may be expressed as

$$\alpha_{v_i v_n}^{(t)} = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^\top [\mathbf{P}\mathbf{h}_{v_i} \oplus \mathbf{P}\mathbf{h}_{v_n}]))}{\sum_{v_m \in \mathcal{N}(v_i)} \exp(\text{LeakyReLU}(\mathbf{a}^\top [\mathbf{P}\mathbf{h}_{v_i} \oplus \mathbf{P}\mathbf{h}_{v_m}]))}, \quad (3.14)$$

where  $\mathbf{P}$  denotes a trainable weight matrix,  $\mathbf{a}$  denotes a trainable attention weight vector, and the embeddings  $\mathbf{h}_{v_i}$ ,  $\mathbf{h}_{v_n}$ , and  $\mathbf{h}_{v_m}$  correspond to previous attentional layer  $\mathbf{h}^{(k-1)}$  — the superscript  $(k-1)$  was omitted from (3.14) for the sake of simplicity [108]. Furthermore, LeakyReLU is a variant of the *rectified linear unit* (ReLU) activation function. LeakyReLU outputs a small, non-zero gradient for negative input values, as opposed to the ReLU activation function which outputs 0 for negative input values [183]. The hidden embedding  $\mathbf{h}_{v_i}^{(k)}$  may be computed by means of the expression

$$\mathbf{h}_{v_i}^{(k)} = \left\| \right\|_{t=1}^T \sigma \left( \sum_{v_n \in \mathcal{N}(v_i)} \alpha_{v_i v_n}^{(t)} \mathbf{W}^{(t)} \mathbf{h}_{v_n}^{(k-1)} \right),$$

<sup>15</sup>Self-attention is a mechanism that relates different positions within a single sequence in order to compute a comprehensive representation of the entire sequence. Masked self-attentional layers hide (*i.e.* mask) subsequent positions within the sequence to prevent information leakage by ensuring that the current position's representation is only affected by representations of preceding positions [281].

<sup>16</sup>In order to stabilise the learning process of self-attention, Veličković *et al.* [282] employ multi-head attention which enables the model to address information from different representation subspaces at different positions simultaneously.

where  $\mathbf{W}^{(t)}$  denotes the weight matrix of the corresponding input linear transformation for each attention head  $t$  and  $\parallel_{t=1}^T$  denotes the concatenation over the  $T$  attention heads [282]. In the case of the final iteration  $k = K$ , concatenation is unnecessary and instead a simple aggregation (or averaging) operation is employed [282]. The final embedding  $\mathbf{h}_{v_i}^{(K)}$  may be expressed as

$$\mathbf{h}_{v_i}^{(K)} = \sigma \left( \frac{1}{T} \sum_{t=1}^T \sum_{v_n \in \mathcal{N}(v_i)} \alpha_{v_i v_n}^{(t)} \mathbf{W}^{(t)} \mathbf{h}_{v_n}^{(K-1)} \right). \quad (3.15)$$

## GATv2

Brody *et al.* [38] proposed GATv2 in order to address certain limitations of the original GAT architecture. They reported that the original GAT employs a rudimentary incarnation of attention (which they called *static attention*), consequently, they proposed a more expressive form called *dynamic attention*. Static attention may be formally defined as follows. Given a (possibly infinite) set of scoring functions  $\mathcal{F} \subseteq \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$  which compute static scores for a given set of key vectors  $\mathcal{K} = \{\mathbf{k}_1, \dots, \mathbf{k}_n\}$  and query vectors  $\mathcal{Q} = \{\mathbf{q}_1, \dots, \mathbf{q}_m\}$ , where  $m, n \in \mathbb{N}$  and  $\mathcal{K}, \mathcal{Q} \in \mathbb{R}^d$ , if for every  $f \in \mathcal{F}$  there exists a “highest scoring” key denoted by  $j_f$  such that for every query  $i$  and key  $j$  it holds that  $f(\mathbf{q}_i, \mathbf{k}_{j_f}) \geq f(\mathbf{q}_i, \mathbf{k}_j)$ . Essentially, if the attention function always assigns at least as much weight to one key as it does to any other key, then the attention function is considered to be static. Within the context of graphs, the query vector is the representation of the current vertex and the key vectors are the representation of its neighbouring vertices. Brody *et al.* highlight that the main issue with the standard GAT scoring function, as expressed in (3.14), pertains to the learned layers  $\mathbf{P}$  and  $\mathbf{a}$  being applied consecutively, resulting in their so-called “collapse” into a single linear layer which is equivalent to a standard linear transformation. In order to address this limitation (dynamically), layer  $\mathbf{a}$  is applied after the nonlinearity, *i.e.* LeakyReLU, while layer  $\mathbf{P}$  is applied after the concatenation, thereby transforming (3.14) into

$$\alpha_{v_i v_n}^t = \frac{\exp(\mathbf{a}^\top \text{LeakyReLU}(\mathbf{P}[\mathbf{h}_{v_i} \oplus \mathbf{h}_{v_n}]))}{\sum_{v_m \in \mathcal{N}(v_i)} \exp(\mathbf{a}^\top \text{LeakyReLU}(\mathbf{P}[\mathbf{h}_{v_i} \oplus \mathbf{h}_{v_m}]))}.$$

## Graph transformer operator

The GT operator [227] is inspired by the *unified message passinging model* (UniMP) which was proposed by Shi *et al.* [260]. UniMP is a novel approach towards addressing the challenge of combining GNNs and the label propagation algorithm<sup>17</sup>, both of which are message-passing algorithms that have performed admirably in semi-supervised classification tasks [110, 329]. UniMP incorporates feature propagation of GNNs with label propagation during training and inference (*i.e.* prediction). This is accomplished through the implementation of a GT layer which employs both feature embeddings and label embeddings as input for performing propagation. Thereafter, a masked label prediction strategy — which masks (*i.e.* hides) a random portion of input labels from which predictive tasks are formulated — is employed in order to mitigate overfitting when utilising self-loop input label information during the training process. The

<sup>17</sup>The *label propagation algorithm* proposed by Raghavan *et al.* [229] is a time-efficient community detection algorithm that employs only the network structure to guide community detection. Furthermore, it does not rely on prior information such as the number, size, or central vertices of communities within the network.

resulting GT may be expressed as

$$\mathbf{h}_{v_i}^{(k)} = \mathbf{W}^{(k)} \mathbf{h}_{v_i}^{(k-1)} + \prod_{t=1}^T \left( \sum_{v_n \in \mathcal{N}(v_i)} \alpha_{v_i v_n}^{(t)} \mathbf{W}^{(k)} \mathbf{h}_{v_n}^{(k-1)} \right).$$

### 3.6 Link prediction on bipartite graphs

Kunegis *et al.* [157] formulated and contextualised link prediction through the lens of bipartite graphs. It was reported that standard formulations of CN-based approaches (as discussed in §3.2) cannot be applied effectively to bipartite graphs due to the requirement of *triadic closure*, *i.e.* two non-adjacent vertices that share a common neighbour should permit the formation of a link between them, as shown in Figure 3.9 [63]. In a bipartite graph, however, vertices are partitioned into two disjoint sets, and edges can only join vertices from different sets, *i.e.* no edges can form between vertices within the same partite set.

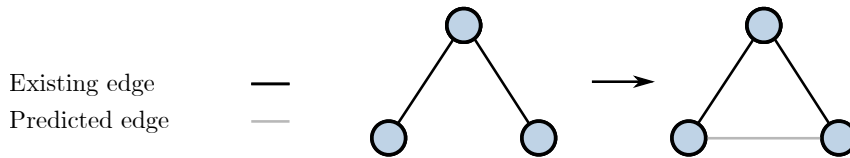


FIGURE 3.9: *The property of triadic closure.*

Various approaches have been proposed towards conducting link prediction in bipartite graphs. For example, Kunegis *et al.* [157] employed algebraic link prediction methods which are based on the eigenvalue decomposition of the adjacency matrix, together with the application of spectral transformation using odd pseudokernels. The odd components of the pseudokernels are employed due to connected vertices in a bipartite graph having odd path lengths. Other researchers modified the CN-based methods so as to enable their application to bipartite graphs [46, 63, 300]. Daminelli *et al.* [63] based their methodology on the principle of quadratic closure which is analogous to triadic closure but contextualised for bipartite graphs. Notably, Daminelli *et al.* define the CNs for two non-adjacent *seed*<sup>18</sup> vertices as the vertices that are involved in all possible quadratic closures between these seed vertices, as presented in Figure 3.10.

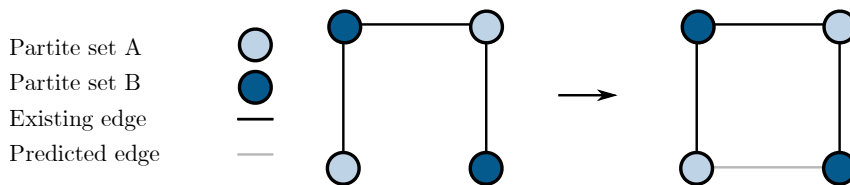


FIGURE 3.10: *The property of quadratic closure.*

Kumar *et al.* [156] proposed a framework for predicting missing links by converting the bipartite graph into a monopartite graph using weighted projections after which the potential energy and mutual information in respect of each vertex-pair in the projected graph are computed. Aziz *et al.* [17] defined a local similarity measure for link prediction within in a bipartite graph. The method extends the concept of *local community links* (LCL) proposed by Cannistraci *et al.* [41] which represents the number of links present in the cohort formed by the neighbours of two non-adjacent vertices belonging to disjoint sets. Consider the two disjoint sets  $\mathcal{V}_1$  and  $\mathcal{V}_2$  of

<sup>18</sup>The non-adjacent vertex-pair for which an edge is to be predicted.

a bipartite graph, where  $\mathcal{V}_1, \mathcal{V}_2 \subset \mathcal{V}$  and  $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$ . Furthermore, consider the vertices  $v_x \in \mathcal{V}_1$  and  $v_y \in \mathcal{V}_2$ , as well as  $v_i \in \mathcal{N}(v_x)$  and  $v_j \in \mathcal{N}(v_y)$ . Then the LCL between (non-adjacent) vertices  $v_x$  and  $v_y$  may be expressed as

$$s^{LCL}(v_i v_j) = \left| \{ \{v_i, v_j\} : \{v_i, v_j\} \in \mathcal{E}, v_i \in \mathcal{N}(v_x), v_j \in \mathcal{N}(v_y) \} \right|. \quad (3.16)$$

The proposed approach by Aziz *et al.* [17], called *path-based resource allocation* (PRA), may be expressed as

$$s^{PRA}(v_i, v_j) = \sum_{\substack{\{v_i, v_j\} : \{v_i, v_j\} \in \mathcal{E} \\ v_i \in \mathcal{N}(v_x) \\ v_j \in \mathcal{N}(v_y)}} \frac{1}{|\mathcal{N}(v_i)| |\mathcal{N}(v_j)|}. \quad (3.17)$$

Furthermore, Aziz *et al.* [17] proposed CN-based indices for bipartite graphs based on the work of Daminelli *et al.* [63] and Cannistraci *et al.* [41]. Before elucidating these CN-based indices, important notation is presupposed, more specifically, let  $\hat{\mathcal{N}}(v_i)$  denote the set of all neighbouring vertices of vertex  $v_i$ 's neighbours. Formally,  $\hat{\mathcal{N}}(v_i)$  may be expressed as

$$\hat{\mathcal{N}}(v_i) = \bigcup_{v_z \in \mathcal{N}(v_i)} \mathcal{N}(v_z).$$

The CAR index (*i.e.* the counterpart of CNs) may be expressed as

$$s^{CAR}(v_i, v_j) = \left| \{ \hat{\mathcal{N}}(v_x) \cap \mathcal{N}(v_y) \} \cup \{ \{ \mathcal{N}(v_x) \cap \hat{\mathcal{N}}(v_y) \} \} \right| s_{LCL}. \quad (3.18)$$

The RA counterpart for bipartite graphs may be expressed as

$$s^{RA}(v_i, v_j) = \sum_{v_z \in \{ \{ \hat{\mathcal{N}}(v_x) \cap \mathcal{N}(v_y) \} \cup \{ \{ \mathcal{N}(v_x) \cap \hat{\mathcal{N}}(v_y) \} \} } \frac{1}{|\mathcal{N}(v_z)|}. \quad (3.19)$$

The Cannistraci variant of the RA method, called the CRA method, may be expressed as

$$s^{CRA}(v_i, v_j) = \sum_{v_z \in \{ \{ \hat{\mathcal{N}}(v_x) \cap \mathcal{N}(v_y) \} \cup \{ \{ \mathcal{N}(v_x) \cap \hat{\mathcal{N}}(v_y) \} \} } \frac{|\gamma(v_z)|}{|\mathcal{N}(v_z)|}, \quad (3.20)$$

where  $|\gamma(v_z)|$  denotes the number of LCL that originate from  $v_z$ .

The Cannistraci variant of the AA method, *i.e.* called the CAA method, may be expressed as

$$s^{CAA}(v_i, v_j) = \sum_{v_z \in \{ \{ \hat{\mathcal{N}}(v_x) \cap \mathcal{N}(v_y) \} \cup \{ \{ \mathcal{N}(v_x) \cap \hat{\mathcal{N}}(v_y) \} \} } \frac{|\gamma(v_z)|}{\log_2 |\mathcal{N}(v_z)|}. \quad (3.21)$$

### 3.7 Link prediction on knowledge graphs

In §2.1.5, the notion of multi-relational data and heterogeneous graphs (of which KGs are a prominent manifestation) was introduced. These data representations have been afforded more attention in the literature due to their ability to capture multiple interactions amongst different entity types, rendering their utility in respect of modelling complex networks (from which complex graphs are derived) justified [293]. The complex and information-rich nature of these graph structures present notable computational challenges to traditional link prediction approaches. Wang *et al.* [293] argue that traditional CN-based approaches can be overly simplistic and

biased when applied to the homogenous (monopartite) graphs derived from the original heterogeneous graphs, as the different vertex and edge types are not abstracted sufficiently. Traditional methods also place greater emphasis on extracting structural features from vertices and edges, consequently the high-dimensional KGs cannot be processed effectively. Furthermore, Wang *et al.* state that the pairwise computations carried out by traditional methods can be intractable in respect of large-scale KGs.

Towards addressing this issue, methods such as non-negative matrix factorisation [50], probabilistic latent tensor factorisation methods [95], and learning-based methods [75] have been proposed in the literature. These methods do, however, have certain shortfalls. For example, the approaches proposed by Dong *et al.* [75] and Rosetti *et al.* [243] depend on domain-specific features as input to graphs, therefore rendering generalisability to other domains problematic. These methods cannot incorporate certain content<sup>19</sup> features which may result in information loss [293]. The development of GNNs [249] represents an approach towards addressing these issues, upon which further extensions, such as GCNs and other prominent architectures such as SAGE and GATs, introduced in §3.5.4, have been proposed [147].

As mentioned in §3.5.4, Schlichtkrull *et al.* [251] extended upon the original GCN model and applied it to heterogeneous graphs by introducing the R-GCN model. Furthermore, the approach proposed by Wang *et al.* [293] represents an extension on the original GCN model by identifying neighbourhood vertices that are structurally more informative with respect to the target vertex and filtering out redundant vertices. Their approach outperformed a variety of CN-based methods, classifier-based methods, and the GCN model in respect of four different heterogeneous data sets, highlighting the potential of GNNs designed specifically for heterogeneous networks.

Zhang *et al.* [317] introduced the notion of heterogeneous GRL, stating that although notable progress had been achieved towards heterogeneous graph embeddings and GNNs, only a small number of these methods consider heterogeneous structural information and heterogeneous vertex features. Appropriately, Zhang *et al.* [317] proposed the *heterogeneous* GNN, called HetGNN, comprising three main phases, namely: Sampling heterogeneous neighbours, encoding heterogeneous contents, and aggregating heterogeneous neighbours. First, the model employs a method known as *random walk with restart* [37] in order to sample a fixed number of heterogeneous neighbours for each vertex which is subsequently grouped according to vertex type. Thereafter, a neural network architecture comprising two modules is employed to perform feature aggregation on the sampled neighbours. The first module generates content embeddings for each vertex by encoding their respective heterogeneous features. The second module aggregates the content embeddings of different neighbouring groups (or types). The aggregated embeddings are combined by an attention mechanism in order to model the importance of the respective vertex types and obtain the final vertex embedding. Finally, a graph context loss function is employed together with a mini-batch gradient descent training procedure. The HetGNN model was subjected to extensive experimental analysis in respect of several data sets — it outperformed various state-of-the-art baselines with respect to link prediction and vertex classification.

The *topic-aware heterogeneous GNN* (THGNN) model by Xu *et al.* [302], on the other hand, involved the application of an alternating two-step aggregation mechanism called intra-metapath<sup>20</sup> decomposition as well as inter-metapath integration. This approach facilitates the distinctive aggregation of rich heterogeneous information according to inferential topic-aware factors and maintains the hierarchical semantics for learning vertex representations of different types for

<sup>19</sup>Entities may comprise different feature types, *i.e.* categorical and/or continuous features. Due to the complexities associated with representing these different features simultaneously, a large number thereof may be removed when generating heterogeneous graphs, leading to information loss [293].

<sup>20</sup>In the context of this study, a metapath corresponds to a relation type connecting two entities.

link prediction. Experimental results in respect of three data sets revealed that the THGNN model consistently outperformed state-of-the-art baselines.

It is evident that a number of powerful algorithmic approaches have been proposed in the literature, each with their own computational differences and algorithmic performance in respect of various data sets. A few noteworthy observations may be gleaned from these studies. Link prediction problems solved using traditional (*i.e.* CN) methods limit features to local neighbourhoods, as described in §3.2. Although these approaches are simple to implement, their sole application does not typically deliver superior performance [180]. In the case of classifier-based approaches, according to which various vertex features are taken into account together with structural information, only limited abstractions within the graph data can be learnt algorithmically [180]. GNNs (including extensions for heterogeneous data), on the other hand, demonstrate enhanced utility in respect of learning complex abstractions within graph data, taking into account both structural properties and vertex features, whilst utilising powerful mechanisms (such as attention) and favourable performance scaling in respect of large data sets comprising many instances.

### Graph ML-based disease prediction

Graph ML methods represent an effective approach towards analysing KGs and have been applied to the clinical domain in various studies, especially so in recent years (at the time of writing). This trend may be attributed to both the aforementioned increasing popularity of clinical KGs, and due to the inferential capabilities of the various algorithmic approaches that may be considered. Lu *et al.* [176] conducted a review on studies from 2015 to 2022 that implemented graph ML techniques to perform disease prediction in respect of EHR data. In particular, Lu *et al.* classified the disease prediction problem in graphs as a node classification task or a link prediction task.

With respect to node classification tasks, Liu *et al.* [171] proposed a novel temporal graph in which they considered patient event sequences (*e.g.* clinical notes, symptoms, medications, vital signs, laboratory reports) that were extracted from EHR data — it formed the basis for performing downstream predictive tasks. The approach was subsequently validated by performing two prediction tasks, *i.e.* predicting the onset risk of heart failure and predicting the risk of heart failure related hospitalisation in patients with a chronic obstructive pulmonary disease pre-condition. The results of the study showcased that diagnosis prediction experiences significant improvement as a result of the implementation of the proposed approach. Khan *et al.* [145] employed ML classifiers to predict the risk of type 2 diabetes on a disease network based on comorbid conditions of 4600 patients. Various graph related features were derived from the graph which then served as input to the predictive models. In a similar study, Lu *et al.* [177] constructed a graph comprising clinical conditions for a group of patients diagnosed with type-2 diabetes. Socio-demographic information, behavioural characteristics and network features were employed in ML models such as LR,  $k$ NNs, NBs, DTs, and RFs towards predicting chronic disease risk in patients.

Predicting disease interactions in complex networks through means of a link prediction approach is becoming increasingly significant and challenging [176]. A study by Davis *et al.* [64] employed a collaborative filtering approach to predict the most significant diseases that each patient is likely to contract based on personal medical history and that of similar patients. The model developed for the study employed medical ontology codes as the basis for the predictions and is regarded as the first study to employ collaborative filtering in this context. Wang *et al.* [292] conducted a similar study in which they investigated the task of performing multiple disease

risk prediction for patients in a post-discharge context. Their proposed framework combined a directed disease network containing temporal aspects together with recommendation system techniques to compute a disease-risk score for patients. Del Valle *et al.* [67] applied a path-based random walk approach, called metapath2vec [74], to a heterogeneous disease-symptom network to predict disease comorbidities. The results obtained from their study were supported by medical literature and showcased improved performance when compared to similar studies based on biological data.

### GNNs in the clinical domain

GNN models have achieved excellent performance in a variety of domains due to their ability to extract features based on the network structure of the data, thereby facilitating automated feature extraction as opposed to relying on manually computed graph connectivity information. Furthermore, unlike shallow embedding methods, GNNs can generate embeddings in respect of unseen data [108, 176]. Sun *et al.* [266] achieved state-of-the-art performance with respect to the node classification task using GNNs. They argued that existing disease prediction approaches, which are based on sequential EHR data, are unable to generalise to new patients without historical EHR data, reducing their practical feasibility. In order to address this, they proposed a GNN based model which augments and supplements EHR data with medical knowledge from external sources and subsequently learns vertex embeddings for patients, diseases, and symptoms. The proposed neural graph encoder showcased the ability to infer embeddings for new patients based on the symptoms reported in their EHRs — enabling accurate prediction of both general and rare diseases.

Wang *et al.* [287] proposed a clinical data model which integrated multiple genomic data and clinical data based on a GCN to predict cancer survival and showcased improved results when compared with previous works. Similarly, Gao *et al.* [94] presented a framework that employed GNNs towards predicting cancer survival using embeddings obtained from a bipartite graph containing patient and multimodal data. Wang *et al.* [291] conducted a link prediction task on a patient-disease bipartite graph comprising 750 000 patients and 42 unique conditions for clinical risk prediction. A GCN was employed to learn a vertex's representation based on its neighbourhood structure and showcased improved performance when compared with baseline techniques.

Lu *et al.* [176] concluded that GNN-based methods achieve superior performance when compared with the shallow embedding approaches for disease prediction. The number of published studies employing GNNs for disease prediction have also increased over recent years (at the time of writing). GNNs, according to Lu *et al.* [176], have proven to be effective in modelling disease prediction, exhibiting improved performance in respect of a variety of experimental outcomes when compared with alternative approaches. Parshotam and Nel [219] conducted a comprehensive evaluation of four different GNN architectures, namely: SAGE, GAT, GATv2, and the GT operator. The link prediction task conducted in their study involved the prediction of new edges between patient and condition vertices. The study compared the performance of the GNN architectures with respect to both scale (*i.e.* different data set sizes) and complexity (*i.e.* increased heterogeneity of vertex and edge types). The GATv2 and GAT architectures consistently outperformed their counterparts with respect to their link prediction performance on the four graphs employed in their study.



### 3.8 Performance evaluation

The performance evaluation of an ML algorithm typically comprises two main components, *i.e.* evaluating the ML algorithm’s performance with respect to the training and validation data sets, after which the ML algorithm’s performance is evaluated with respect to unseen (hold-out) data. Evaluating an algorithm’s performance with respect to the training set provides some insight into the extent to which the informative patterns and abstractions within the training data have been inferred. Training performance alone may not necessarily provide an accurate representation of the algorithm’s *generalisation* capabilities. Appropriately, the algorithm’s performance is evaluated with respect to unseen (*i.e.* testing) data which provides a more robust and unbiased measure of the model’s performance.

A validation set may be employed (prior to testing) as a proxy for evaluating generalisation capabilities. Three popular approaches are employed in respect of the validation data set, namely: *k-fold cross-validation* [263], *bootstrapping* [79], and *regularisation* [275]. In the case of *k-fold cross-validation*, the training set is partitioned into *k* subsets, also called *folds*, and *k* training iterations are carried out. During each training iteration, the algorithm is trained using *k* – 1 folds, while the remaining fold is employed for validation purposes. An example of *k-fold cross-validation*, for which *k* = 5, is illustrated in Figure 3.11. The performance of a model is then determined by computing the average performance over all *k* folds, providing a more robust performance evaluation. Bootstrapping, on the other hand, involves randomly sampling the data with replacement for *k* iterations and then calculating the average performance of the model over the *k* iterations. Finally, a validation set can also be employed for regularisation purposes, according to which the training process is terminated once validation performance decreases even though training performance might continue increasing [138, 330].

Iteration 1	Validate	Train	Train	Train	Train
Iteration 2	Train	Validate	Train	Train	Train
Iteration 3	Train	Train	Validate	Train	Train
Iteration 4	Train	Train	Train	Validate	Train
Iteration 5	Train	Train	Train	Train	Validate

FIGURE 3.11: A graphical illustration of *k-fold cross-validation* where *k* = 5 [330].

Algorithmic performance may be contextualised by means of a so-called *goodness-of-fit* (GOF) test which provides an indication of a model’s ability to replicate the observations within the training set. A GOF test may be described by means of two concepts, namely: *Overfitting* and *underfitting*. Overfitting corresponds to the algorithm memorising the training data set as opposed to learning an adequate (latent) representation of the data — the algorithm therefore performs well with respect to the training set, but poorly with respect to the validation and test set [138, 286]. This phenomenon can be ascribed to the algorithm possibly abstracting noise as actual patterns within the data, resulting in errors of variance<sup>21</sup>. Underfitting, on the other hand, relates to the algorithm performing poorly with respect to both the training and validation sets, suggesting errors of bias<sup>22</sup> in the algorithm. Overfitting, as illustrated in Figure 3.12(a), often corresponds to overly complex algorithms memorising instances within the

<sup>21</sup>Variance relates to errors caused by fluctuations within the data [330].

<sup>22</sup>Bias relates to errors caused by incorrect model assumptions [330].

training set whereas underfitting, as illustrated in Figure 3.12(c), often corresponds to overly simplistic algorithms that cannot abstract the complexity embedded within the training data due to representational inabilities. The so-called *bias-variance* trade-off ought to be considered when selecting an appropriate algorithm, in order to deliver performance akin to that of Figure 3.12(b) [330].

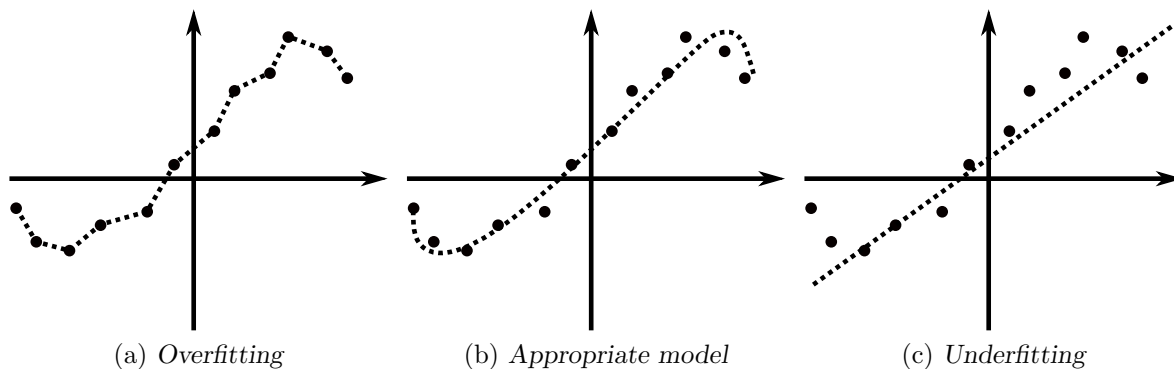


FIGURE 3.12: A graphical illustration of the bias-variance trade-off, *i.e.* a model that is (a) too complex or (c) too simplistic for the given data results in inferior performance. An appropriate model (b) results in suitable performance.

### 3.8.1 Evaluation metrics

Evaluation metrics quantify algorithmic performance from which inference can be drawn into the efficacy of the modelling approach adopted. The nature of the problem to be solved (as part of the data mining process) can provide some insight into the type of evaluation metrics that should be considered, the most common of which are *accuracy*, *precision*, *recall*, *F-score*, and the *area under the curve* (AUC) of a *receiver operating characteristic* (ROC) curve (denoted by AUROC) as well as the *area under the precision-recall curve* (denoted by AUPRC). In the case of binary classification problems, algorithmic performance can be gleaned from a confusion matrix, as shown in Figure 3.13.

		Actual class	
		+	-
Predicted class	+	True Positive	False Positive
	-	False Negative	True Negative

FIGURE 3.13: A confusion matrix for a binary classification problem in which an entry in the first row and first column corresponds to correctly classified positive (+) class instances, while an entry in the second row and second column corresponds to the correctly classified negative (-) class instances.

The confusion matrix provides two classification errors, namely: *False positives* (FP) which correspond to negative instances that have been erroneously classified as positive instances, and *false negatives* (FN) which represent positive instances that have been erroneously classified as negative. FPs and FNs are also referred to as Type 1 and Type 2 errors, respectively. *True positives* (TP) represent positive observations that have been correctly classified as positive, while *true negatives* (TN) refer to negative instances that have been correctly classified as negative. A variety of metrics, such as accuracy, precision, recall, and *F-score* may be computed in respect of this confusion matrix. Accuracy is defined as the proportion of correctly classified

instances in respect of both the positive and negative classes, expressed as

$$\frac{TP + TN}{TP + TN + FP + FN}.$$

Accuracy is often deemed inadequate due to its inefficacy in respect of imbalanced (*i.e.* skew) data sets [330]. Precision, on the other hand, is the proportion of correctly classified positive observations with respect to the total number of observations that are classified as positive, expressed as

$$\frac{TP}{TP + FP}.$$

Recall is the proportion of correctly predicted positive observations with respect to the total number of positive observations which may be expressed as

$$\frac{TP}{TP + FN}.$$

Lastly, the *F*-score denotes the harmonic mean between precision and recall. The value of the *F*-score ranges between zero (worst) and one (best) and can be computed by means of

$$\frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}.$$

The AUROC is a popular approach when dealing with class imbalance as it provides a more robust measure of the extent to which an ML algorithm can correctly distinguish between the positive and negative observations [138]. It is important to note that AUROC is threshold<sup>23</sup> independent, as it considers all thresholds between zero and one. An ROC curve represents the *true positive rate* (TPR) plotted against the *false positive rate* (FPR), where

$$TPR = \frac{TP}{TP + FN}$$

and

$$FPR = \frac{FP}{TN + FP}.$$

The TPR is therefore identical to recall. Three distinct ROC curves are presented in Figure 3.14. ROC1 represents the most desirable case, according to which the algorithm can perfectly distinguish between the positive and negative observations — the corresponding AUROC value equates to 1. ROC3 is equivalent to random guessing and has an AUROC of 0.5. The model is therefore unable to distinguish between the positive and negative observations reliably. ROC2 represents a classifier for which performance is deemed to be intermediate, *i.e.* between the perfect classifier, ROC1, and random guessing, ROC3 [140].

The AUPRC evaluates the performance of a classification algorithm by calculating the area under the precision-recall curve. The precision-recall curve is obtained by plotting precision (the proportion of true positive predictions among all positive predictions) against recall (the proportion of true positive predictions among all actual positives). Similarly, AUPRC is also threshold independent [65]. The AUPRC metric is particularly beneficial in respect of problems for which the positive class is under-represented (as is the case with link prediction) — this may be ascribed to AUPRC's greater bias towards the positive class when compared with AUROC. Yang *et al.* [306] conducted a comprehensive investigation of link prediction methods in which experimental analysis revealed that the precision-recall curve and AUPRC are considered more robust estimators of performance in link prediction tasks.

<sup>23</sup>A threshold may be defined as a set point representing the boundary for classifying predicted probabilities (or scores) into either positive or negative observation labels, *i.e.* probabilities (or scores) greater than or equal to the threshold are classified as a positive observation, whereas probabilities (or scores) less than the threshold are classified as a negative observation.

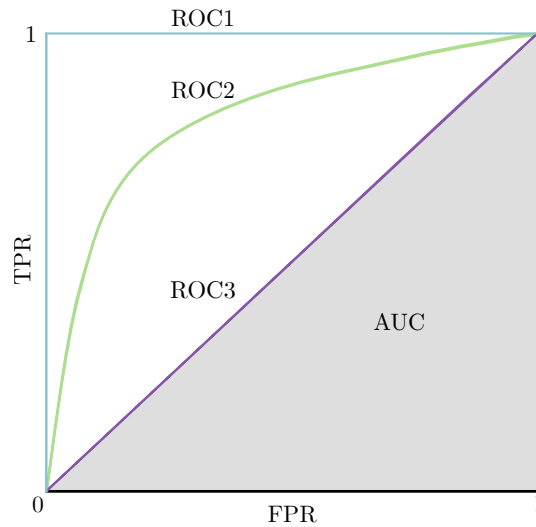


FIGURE 3.14: Plots of three distinct ROC curves illustrating the ideal performance ROC1 (blue), average performance ROC2 (green), and inadequate performance ROC3 (purple) [330].

### 3.8.2 Graph partitioning

Different methodologies for partitioning graph data into training, validation, and test sets are now elucidated. The methodologies employed for partitioning graph data depend on the applied link prediction algorithmic approach. Three main approaches are discussed, namely: Traditional methods (relating to CN and path-based methods), classification based-methods, and GNNs. In respect of each of these methods, partitioning is performed either randomly or temporally (depending on the problem context).

#### Traditional methods

The approach adopted towards conducting performance evaluation of traditional link prediction methods with respect to the aforementioned metrics (which require positive and negative observations) is now elucidated. Towards this end, positive observations and negative observations are to be generated — recall that positive observations correspond to existing edges and negative observations correspond to non-existent edges.

Consider the graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ . First, a set of negative observations, *i.e.* non-existent edges corresponding to the set  $\mathcal{E}' = \mathcal{U} \setminus \mathcal{E}$ , are generated (typically referred to as *annotation*). The set of existing edges  $\mathcal{E}$  (*i.e.* positive observations) is subsequently partitioned into two disjoint sets, namely: A training set  $\mathcal{E}^T$  and a *probe* set  $\mathcal{E}^P$  — the probe set serves as the set of positive observations which have been removed for computational purposes. An adjacency matrix  $\mathbf{A}$  is constructed in respect of the training set, expressed as

$$\mathbf{A}_{v_i, v_j} = \begin{cases} 0, & \{v_i, v_j\} \notin \mathcal{E}^T \\ 1, & \{v_i, v_j\} \in \mathcal{E}^T. \end{cases}$$

The adjacency matrix serves as the basis from which the similarity scores are calculated. Accordingly, a score is calculated for all vertex-pairs corresponding to negative observations and the positive observations for which the associated ground truth is known. The predicted scores are assessed with respect to some threshold in order to ascertain the presence or absence of an edge. Subsequently, these determinations are compared with the ground truth values in order to compute the evaluation metrics.

In the case of static networks, *i.e.* networks comprising no temporal aspects, the respective sets are generated by randomly sampling vertex-pairs from a uniform distribution without replacement, therefore these two sets are mutually exclusive, *i.e.*  $\mathcal{E}^T \cap \mathcal{E}^P = \emptyset$  [179]. In the case of graphs that have some temporal properties, a common approach involves selecting a specific point in time and partitioning all edges prior to this point into the training set, whereas all edges following this point are partitioned into the test set. The underlying intuition of this approach is that future links are based on historical interactions within the network. Aziz *et al.* [17] employed this method of partitioning a temporal graph data set in respect of predicting multi-morbidity for patients, while Kunegis *et al.* [157] applied this approach to link prediction on bipartite graphs.

### Classification-based methods

In order to approximate the functional mapping that underpins learning-based algorithms (as discussed in §3.4), the graph data must first be partitioned into distinct training and test sets. Consider the data set

$$\mathcal{D} = \{(\mathbf{x}_{ij}, y_{ij})\}$$

where  $i, j$  denote all vertex-pairs  $v_i, v_j \in \mathcal{U}$ , *i.e.*  $\mathcal{D}$  contains the feature vectors and corresponding labels pertaining to all adjacent and non-adjacent vertex-pairs. A link prediction problem is based on the underlying assumption that there are some edges *missing* from the graph  $\mathcal{G}$  — *i.e.* some negative instances in  $\mathcal{D}$  should instead be classified as positive instances. The functional mapping  $f$  is approximated through the process of training a classification-based link prediction method in respect of correct data instances, *i.e.* the ground truth. This may be achieved by partitioning the network problem's derived data set  $\mathcal{D}$  into a separate training data set, denoted by  $\mathcal{D}_{\text{train}}$ , comprising true observed edges and true missing edges (*i.e.* non-adjacent vertex-pairs), from which inferential relationships may be inferred algorithmically. The remaining data instances constitute the test set, denoted by  $\mathcal{D}_{\text{test}}$ , and are employed to conduct performance evaluation in respect of the trained link prediction algorithm.

A fraction  $p$  of instances belonging to  $\mathcal{D}$  are randomly sampled (without replacement) in order to construct  $\mathcal{D}_{\text{train}}$ , while the remaining fraction of instances (*i.e.*  $1 - p$ ) is allocated to the test set, denoted by  $\mathcal{D}_{\text{test}}$ . The link prediction algorithm is tasked with predicting the labels  $y_{ij}$  of positive and negative training instances based on their respective feature vectors  $\mathbf{x}_{ij}$ . The function mapping

$$f : \mathbf{x}_{ij} \rightarrow \{0, 1\}$$

is therefore learned in respect of  $(\mathbf{x}_{ij}, y_{ij}) \in \mathcal{D}_{\text{train}}$ . The learnt function mapping is subsequently employed to perform predictions with respect to the labels of testing instances  $\hat{y}_{ij}$  based on their respective feature vectors  $\hat{\mathbf{x}}_{ij}$ , expressed as

$$f(\hat{\mathbf{x}}_{ij}) \approx \hat{y}_{ij},$$

where  $(\hat{\mathbf{x}}_{ij}, \hat{y}_{ij}) \in \mathcal{D}_{\text{test}}$ . The approach adopted towards partitioning  $\mathcal{D}$  into  $\mathcal{D}_{\text{train}}$  and  $\mathcal{D}_{\text{test}}$  should take into account the inherent class imbalance of graphs. *Stratified sampling* may be performed in order to ensure that the distribution of positive and negative instances are similar between  $\mathcal{D}_{\text{train}}$  and  $\mathcal{D}_{\text{test}}$  [179].

## Data leakage

A notable concern during data partitioning is *data leakage*<sup>24</sup>. The inherent interconnectivity of a graph-based data representation, however, often results in data leakage. This is attributable to the fact that the calculation of a training instance’s feature vector  $\mathbf{x}_{ij}$  depends on the two end-vertices of this instance’s edge which, in turn, is dependent on the (other) observed and non-existing edges that are incident with these end-vertices. Consequently, it is typically<sup>25</sup> not possible to calculate a training instance’s feature vector without considering some observed or non-existing edge in the testing set. Furthermore, due to the testing edges being removed (*i.e.* annotated), the calculation of training feature vectors may therefore be based on incorrect information. A common approach towards mitigating data leakage in a rudimentary manner involves limiting the size of the testing set (equivalent to selecting a large fraction  $p$ ) such that  $|\mathcal{D}_{\text{train}}| \gg |\mathcal{D}_{\text{test}}|$ . This approach may be deemed appropriate when considering the typical link prediction context in which it is reasonable to assume that only a relatively small number of missing edges exist. Training may therefore be performed in respect of the majority of instances in  $\mathcal{D}$ , resulting in large training set in relation to the testing set.

There is, however, an evidential lack of consensus in the literature with respect to the efficacy of different partitioning approaches. Important (and often nuanced) factors are either disregarded or not considered collectively. Such factors include: The partition size  $p$ , deciding whether partitioning should be performed randomly or be based on the inherent structural characteristics of the graph, and class imbalance. Despite its computational efficiency, random selection tends to neglect certain innate properties of the graph. Alternative approaches towards partitioning involve distance-based partitioning [306] (which might introduce bias) and criteria-based selection, *e.g.* selecting edges that are incident with vertices that meet some degree threshold [168].

## GNNs

In the context of GNNs, the aim of data partitioning for link prediction is to “hide” certain edges (akin to the annotation procedure described earlier), called supervision edges, from the GNN [162]. The GNN then predicts whether these edges should be present or not based on information derived from the remaining vertices and edges, *i.e.* message-passing edges. The data partitioning process for link prediction therefore comprises two steps. The first step involves annotating the two edge types, *i.e.* message-passing edges and supervision edges, to edges within the original graph. Only the message-passing edges remain in the graph. The supervision edges are used as supervisory signals for the GNN’s predictive task and are therefore not presented to the GNN as input. This process is shown in Figure 3.15.

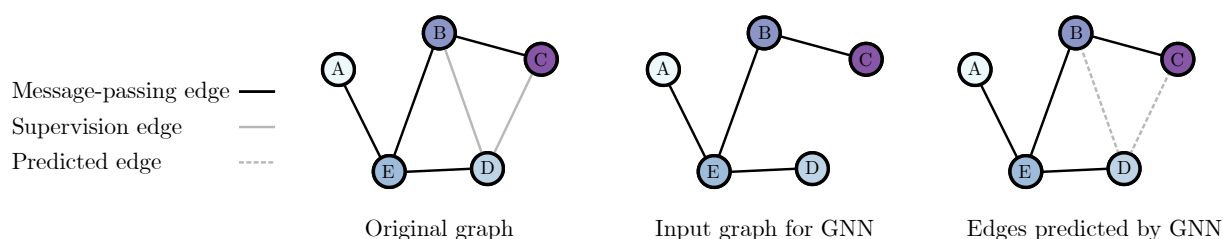


FIGURE 3.15: A basic illustration of the manner according to which edges are split into message-passing edges and supervision edges (adapted from [162]).

<sup>24</sup>Exposing information from the test set to the training process.

<sup>25</sup>Conventional random data partitioning approaches are rudimentary and do not explicitly circumvent data leakage.

As part of the second step, the edges are split into training, validation, and test sets by means of either an *inductive* or *transductive* link prediction split. In the case of an inductive split (presented in Figure 3.16), the training, validation, and test sets each contain an independent graph. In the case of a transductive link prediction split, the entire graph can be observed in all data set splits. Therefore, it is necessary to hold out the validation and test edges to ensure that the model is not trained on them. Furthermore, to train the GNN on the training set, it is necessary to hold out the supervision edges.

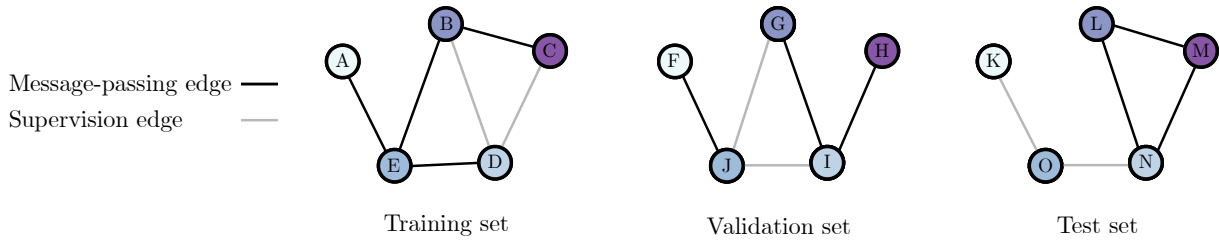


FIGURE 3.16: An example of an *inductive* link prediction data split where each split contains an independent graph (adapted from [162]).

A transductive split is carried out as follows: During training, the GNN employs the training message-passing edges to predict the training supervision edges. In the validation phase, the GNN employs both the training message-passing edges as well as the training supervision edges to predict the validation edges. Lastly, during the testing phase, the GNN employs the training message-passing edges, the training supervision edges, and the validation edges to predict the test edges. The graph progressively expands with the incorporation of new edges during the aforementioned phases, as shown in Figure 3.17.

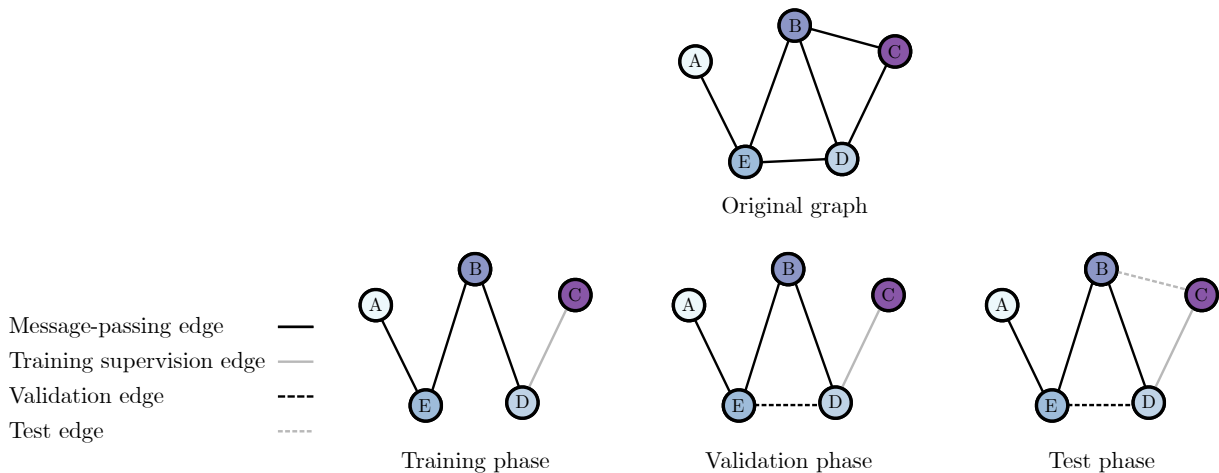


FIGURE 3.17: An example of a *transductive* link prediction data split (adapted from [162]).

### 3.9 Chapter summary

This chapter opened with an introduction to the field of link prediction in §3.1 which forms the basis of the analytical work conducted in this thesis. Thereafter, an overview of various link prediction algorithms, namely: Common neighbour-, path-, classifier-, and embedding-based algorithms, was presented in §3.2–§3.5, respectively. Graphs abstracting real-world phenomena

---

often comprise a variety of vertex and edge types in order to reflect the associated heterogeneity. Appropriately, a discussion of link prediction in respect of bipartite graphs and KGs was presented in §3.6 and §3.7, respectively. Important considerations in respect of performance evaluation of link prediction algorithms were then delineated in §3.8.





# Part II

# Framework



---

---

## CHAPTER 4

---

# Framework discussion

### Contents

4.1	Framework development process . . . . .	76
4.2	Data flow diagrams . . . . .	77
4.3	A generic data science paradigm . . . . .	77
4.4	Quality assurance . . . . .	79
4.4.1	<i>Verification</i> . . . . .	79
4.4.2	<i>Validation</i> . . . . .	80
4.5	Related frameworks . . . . .	81
4.6	The MEDIKAL framework . . . . .	84
4.6.1	<i>Database component</i> . . . . .	85
4.6.2	<i>The Processing component</i> . . . . .	86
4.6.3	<i>The KG construction component</i> . . . . .	90
4.6.4	<i>The Analysis component</i> . . . . .	94
4.7	Design verification . . . . .	99
4.8	Chapter summary . . . . .	100

The solution posited in this thesis is presented in the form of a framework. Appropriately, the aim in this chapter is to delineate the approach adopted towards developing the proposed MEDIKAL framework. First, the reader is introduced to the notion of a framework and its development process. Thereafter, a discussion on DFDs is presented which represent a visually intuitive approach towards assimilating the framework, its constituent components and modules, as well as the flow of data and information. The generic data science paradigm through which the framework is contextualised is subsequently introduced. Thereafter, a discussion pertaining to prominent related frameworks within the clinical domain is presented. A detailed overview of the MEDIKAL framework, its functional components, and their constituent modules is presented during which the aim is to facilitate an improved understanding of the framework's functional working, together with data and information flow within the framework. An overview is then presented of the manner in which quality assurance is incorporated during the design of the proposed framework for verification purposes. Lastly, the chapter concludes with a summary of its contents.

## 4.1 Framework development process

A framework may be defined as an integrated structure of predefined modules which comprise conceptual and programmatic information that can be employed (in an iterative manner) towards encapsulating the fundamental operation of some system or solution methodology [262]. This structure can then serve as the foundation for developing a variety of domain-specific applications, as shown in Figure 4.1. The framework development process commences with domain analysis, *i.e.* the process of identifying, capturing, and organising re-usable information pertaining to the development of a computerised (*i.e.* software) instantiation [226]. The information extracted during such an analysis may include source code, documentation, design plans, knowledge from the relevant literature (including domain experts) as well as current and future system requirements. This information may be subsequently employed in order to generate generic architectures and domain-specific taxonomies which facilitates the ensuing framework design process. The aim during the framework's design is to develop a flexible framework structure that is amenable to a variety of applications within the specified domain [262].

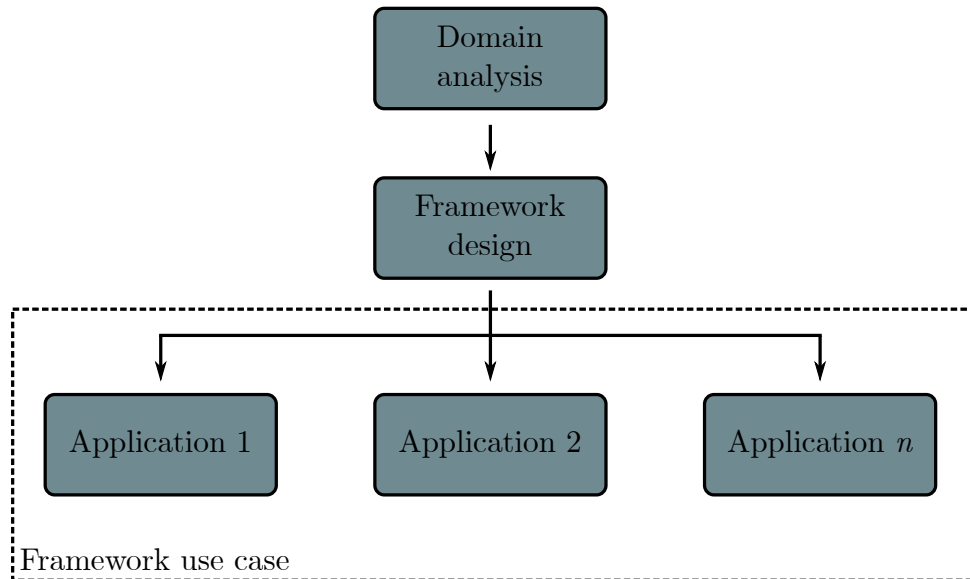


FIGURE 4.1: A graphical representation of the framework development process (adapted from [262]).

During the framework design process, an initial framework structure is constructed based on the requirements and architectures identified during domain analysis. Thereafter, the framework is enhanced by adapting the conceptual and programmatic information (*e.g.* reusable code) identified during the domain analysis and incorporating it into the framework. The framework is then verified with respect to its domain requirements and then subjected to design analyses in order to verify the overarching design process of the framework. A notable task during the framework design process entails a thorough documentation of the framework in order to provide users with a comprehensive understanding of the framework's conceptual and technical working, as well as to elucidate the protocols for its deployment in respect of various applications [89]. Lastly, the framework is transformed into an application-specific instance by constructing application-specific components, and subsequently integrating these components within the framework's functional design in order to generate an instantiation for purposes of the use case under consideration.

## 4.2 Data flow diagrams

The flow of data through some system is best represented by means of a DFD [142]. In DFDs, various informative facets (of a data-driven nature) are depicted in a visually intuitive manner, namely: (1) Data propagation through the different processes that constitute some structured system, (2) data transformations, and (3) outputs from each process. Towards this end, DFDs employ four simple symbols, each of which represent an important aspect of data or information flow, namely: *Entity*, *data flow*, *process*, and *data store*, as presented in Figure 4.2.

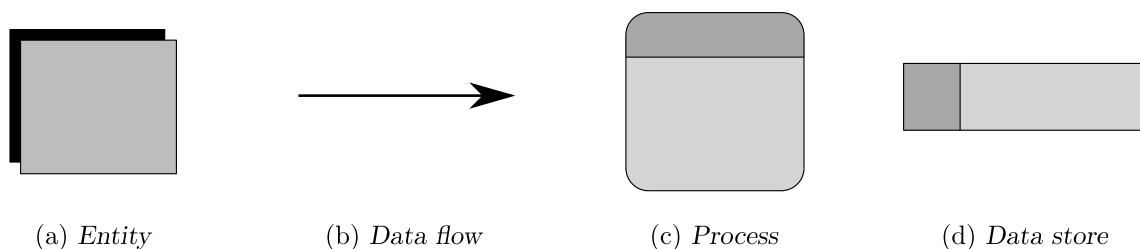


FIGURE 4.2: The four symbols employed to describe the flow of data or information in a DFD [142].

The *entity* symbol is used to depict an external entity that interacts with the system by either sending data to the system or receiving data from the system. The *data flow* symbol represents the movement of data in the system from one point to another. Data flows that occur simultaneously are denoted by parallel arrows. Furthermore, the *process* symbol denotes the transformation of data from its input (native) format to some output format while the *data store* symbol denotes the storage of data [142]. DFDs are developed using a *top-down* approach, therefore different levels of abstraction can be represented. The *context* (or *level-zero*) diagram represents the most generalised abstraction of the system and comprises a single process representing the entire system. Lower levels of abstraction are illustrated by means of, for example, *level-one* and *level-two* diagrams which are obtained by expanding (*i.e.* disaggregating or decomposing) the processes of the preceding diagram [142]. Appropriately, DFDs are employed as an intuitive and informative means towards elucidating the design and working of the framework proposed in this thesis.

## 4.3 A generic data science paradigm

The aim in this thesis, as mentioned in §1.2, is to develop a framework for transforming data obtained from EHRs into KGs in order to extract actionable insights for clinical practitioners by means of various modelling techniques. Ascribed to the data-driven nature of the research aim, the framework proposed in this thesis corresponds to the steps of a generic data science methodology, *i.e.* the CRISP-DM methodology, as described in §3.1.2. By adopting this approach, the benefits associated with CRISP-DM (*e.g.* generic and iterative nature) can be realised throughout the design and implementation of the framework.

A high-level schematic of a generic data science paradigm is presented in Figure 4.3 and comprises three functional components, namely: A *graphical user interface* (GUI), a *central functional component*, and a *data store*. The GUI facilitates communication between the user and the underlying system, while the functional component is employed to execute the primary purpose of the framework. It comprises three subcomponents, namely: *Processing*, *Modelling*, and *Deployment*. Finally, the data store component facilitates the storage of data pertaining to any components (in respect of both required inputs and produced outputs) of the framework [138].

Input data presented by means of the GUI are processed by the modules, labelled  $P_1, \dots, P_i$ , which constitute the processing component. These modules are responsible for the execution of the data preparation phase of the CRISP-DM methodology and therefore perform tasks such as data cleaning, data normalisation, and data reformatting, in order to ensure that the data are in a format that is deemed appropriate for the subsequent modelling component.

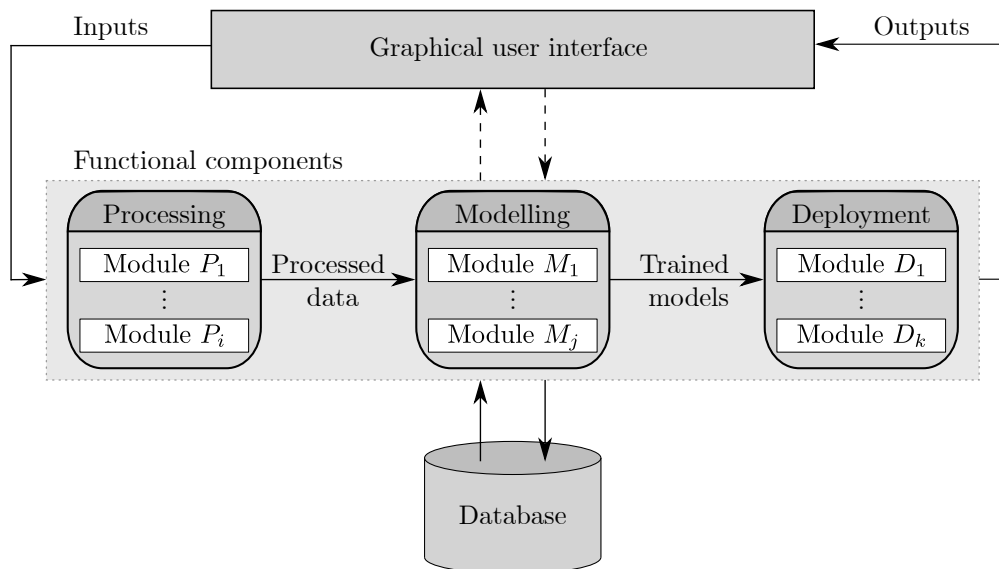


FIGURE 4.3: A high-level schematic of a generic data science paradigm [138].

The modelling component represents the modelling phase of the CRISP-DM methodology and receives as input processed data from the processing component. The constituent modules, labelled  $M_1, \dots, M_j$ , perform different procedures pertaining to model development and evaluation. The modelling techniques employed in this phase are based on the research aim, as well as the type and extent of data available.

The trained models obtained during the modelling component represent the input for the subsequent deployment component. The evaluation and deployment phases of the CRISP-DM methodology are executed within the modules of the deployment component, labelled  $D_1, \dots, D_k$ . These modules are employed towards analysing the results of the trained models in order to synthesise and contextualise insightful information. The analysis performed may range from simple descriptive analytics and visualisations to more complex analytical methods (depending on the research aim).

A centralised data store is central to data management, the aim of which is to reduce interdependence between modules that are embedded within different components. Data can flow from a particular component to the centralised data store, from which it can be retrieved and processed by modules of other components. The modules within a component do not have to be performed sequentially and may be performed in an iterative manner. The user can interact with any of the processes at any given time, according to which any data inputs and/or outputs can be analysed based on one or more modules being executed. User interaction is entirely optional and is indicated by the dashed lines in Figure 4.3.

## 4.4 Quality assurance

Kendall and Kendall [142] proposed three approaches towards quality assurance during system development, namely: (1) Employing a top-down, modular approach to system design, (2) adequately documenting software, and (3) performing system validation and testing. A top-down approach facilitates the process of identifying the objectives of a system as well as the manner according to which these objectives are to be realised. It involves partitioning a system into various subsystems (modules), each of which comprise individual requirements. This approach therefore mitigates the challenges that arise when attempting to design an entire system simultaneously and reduces time requirements by facilitating the development of subsystems that can be executed separately (or in parallel). Most importantly, a top-down approach ensures that the overarching objectives of the final system are not disregarded during the design of more detailed lower level components, a commonality in a bottom-up approach.

Software documentation represents an important part in the maintenance and improvement of a system by enabling and empowering stakeholders to understand the system and its constituent modules without direct interaction [142]. A notable manifestation (or artefact) of system documentation is procedure manuals which comprise a variety of descriptive comments, step-by-step process instructions, and troubleshooting information. Finally, the system can be subjected to testing for verification and validation purposes. Due to their applicability to this research project, a more detailed discussion on each follows.

### 4.4.1 Verification

Verification is the process of determining whether a system has been accurately constructed in accordance with its specified requirements and parameters — it may be conducted by eliminating errors pertaining to syntax, logic, and compilation. Kendall and Kendall [142] proposed a systematic methodology for conducting system verification which incorporates the various stakeholders (and their respective roles) that are involved in system development and testing. A schematic representation of their approach is presented in Figure 4.4.

In the first step, called *Program testing with test data*, programmers manually inspect (*i.e.* by hand) the intuition that underpins each component of the program in order to determine whether the logic is functionally correct. This process is referred to as *desk checking*. Thereafter, valid and invalid test data are generated and propagated through the different modules in order to verify the respective outputs. The second step, called *Link testing with test data*, involves discerning whether independent modules function correctly. Test data are once again generated which includes both valid and invalid data for the purpose of detecting errors.

The third and fourth steps involve input from both *operators* and *end-users* for the testing process. In the third step, called *Full systems testing with test data*, generated test data are employed to determine whether the predefined system objectives have been met, thereby reaffirming quality standards. System documentation is also scrutinised so as to determine its understandability and the extent to which it can convey the appropriate preparation of system input data. Furthermore, this step ensures that information flows correctly through the system and that the correct output can be interpreted by users effectively. In the final step, called *Full systems testing with live data*, the system is subjected to live data, *i.e.* data that have been successfully processed by the developed system. This process facilitates an effective comparison of the new system's output with some produced output that has been proven correct *a priori*.



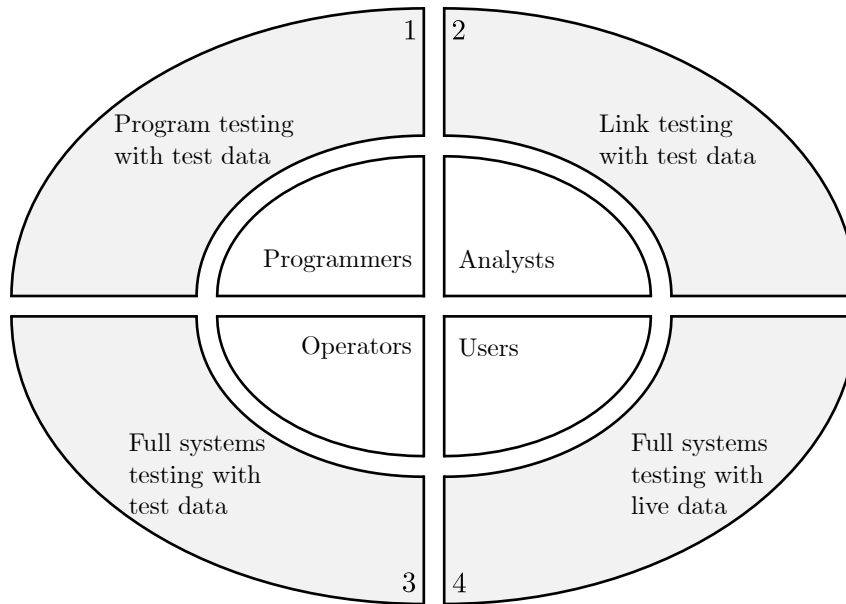


FIGURE 4.4: A systematic methodology for conducting system verification, as proposed by Kendall and Kendall [142].

#### 4.4.2 Validation

As reported by Banks *et al.* [20], the aim during validation is to determine whether the correct system has been constructed. It also involves evaluating and comparing a model's working and outputs with its real-world counterpart. This comparison may be achieved by performing a range of qualitative and quantitative tests. Qualitative tests typically involve stakeholders who are knowledgeable in respect of one or more aspects of the real system, *i.e.* domain experts who scrutinise the model and its output. Quantitative tests, on the other hand, require data pertaining to the real-world system's operation as well as the corresponding model's output data. A notable methodology for performing validation is the three-step approach proposed by Naylor and Finger [204]. The three steps constituting this approach are:

1. Construct a model with high face validity,
2. validate model assumptions, and
3. compare the model input-output transformations to the corresponding input-output transformations of the real-world system.

The first aim involves constructing a model that may be deemed reasonable by individuals who have innate knowledge of the real-world system. Ideally, these individuals should be involved throughout the entirety of the system development process, *i.e.* from model conception to model implementation, so as to ensure relevance and alignment in respect of the desired aim. In the second step, model assumptions made during development are validated with respect to an appropriate collection of reliable data and the appropriate statistical analysis thereof. Lastly, the model's capabilities are tested in respect of predicting future system behaviour based on realistic input data. This may involve evaluating the level of similarity between the output generated by the system and the true output of the real-world system.

## 4.5 Related frameworks

The aim in this section is to conduct domain analysis by presenting an overview of notable frameworks in order to determine current and future requirements of frameworks applied within the clinical domain.

In their article titled “Real-world data medical knowledge graph: Construction and applications”, Li *et al.* [164] propose an end-to-end systematic approach towards constructing a KG from EHR data by means of NLP approaches. The proposed approach, shown in Figure 4.5, comprises eight steps, namely: Data preparation, entity recognition, entity normalisation, relation extraction, property calculation, graph cleaning, related-entity ranking, and graph embedding. An important step in this process, especially in the context of a clinical decision support system, relates to the related-entity ranking step. Diseases, symptoms, and medications are often associated with a number of medical concepts and therefore effective recommendations cannot be made without ranking the importance or significance of the relationship between entities. In order to address this issue, Li *et al.* introduced a so-called *probability-specificity-reliability* function to rank and retrieve the most relevant entities in respect of a given patient from other related entities, the extraction of which is carried out in step four, *i.e.* the relation extraction step.

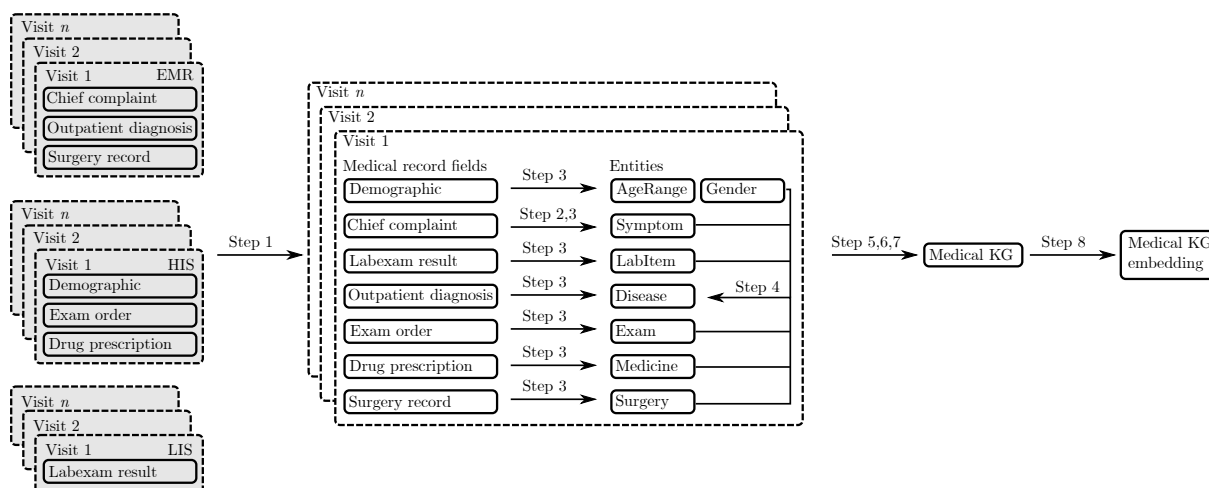


FIGURE 4.5: The modelling pipeline proposed by Li *et al.* [164] for extracting medical concepts from unstructured text data (adapted from [164]).

Cheng *et al.* [53] employed international standard medical terminology to label and analyse stroke cases which formed the basis upon which their relationship classification system was designed. The constructed stroke disease dictionary was then employed to identify symptoms, treatments, medications, and other medical entities related to strokes. The approach adopted towards constructing their KG was abstracted by means of a framework, as presented in Figure 4.6. Cheng *et al.* highlighted the importance of understanding the purpose of the KG in the construction process. The purpose of the graph constructed in their study was to provide patients with a self-examination of strokes as well as creating a medical knowledge reference base for doctors. Accordingly, the graph comprised data pertaining to stroke disease statistics, related treatment methods, symptoms, inspection methods, and medication. The stroke ontology data store was then formed by combining the stroke disease dictionary with the information obtained from the vertical medical websites, crowdsourced websites, and medical literature. In order to ensure an effective construction process, a semi-automated approach was adopted which incorporates both a data-driven methodology as well as medical domain expert knowledge.

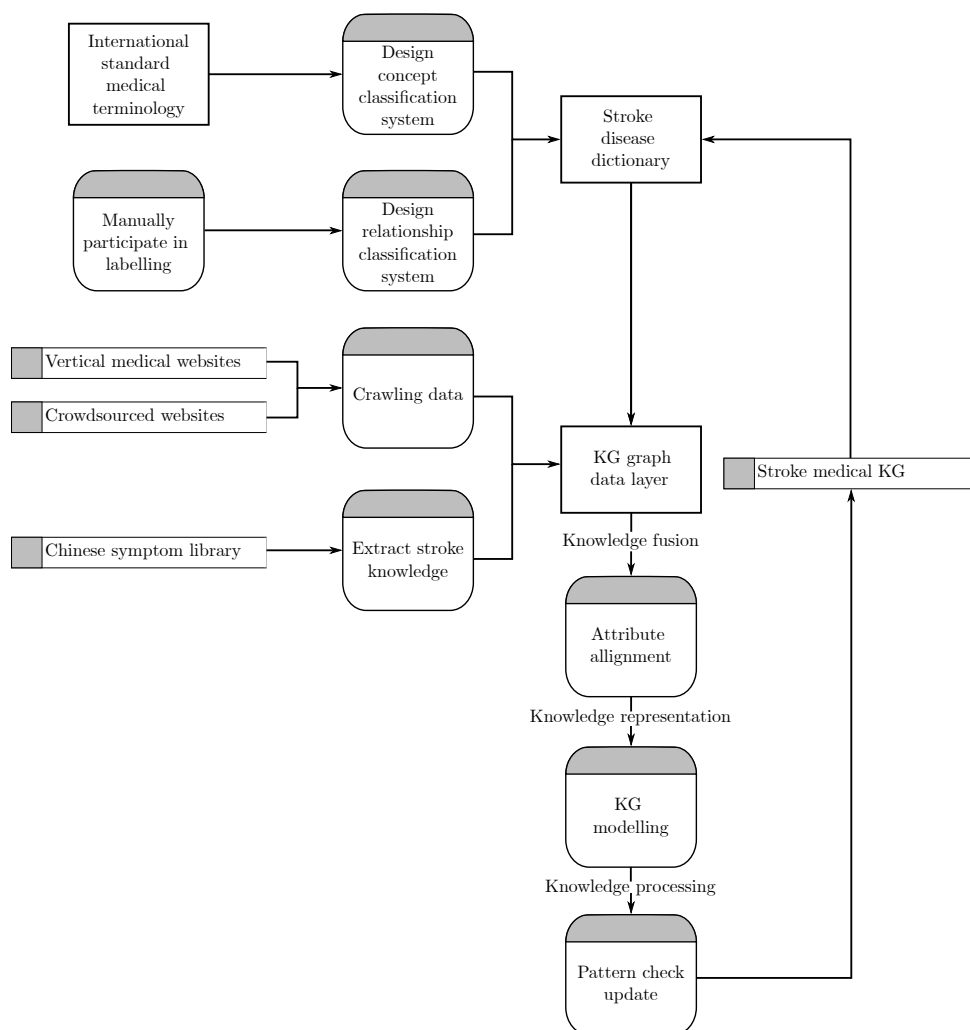


FIGURE 4.6: An adapted diagrammatic representation of the framework construction process employed by Cheng *et al.* [53].

In a notable paper by Santos *et al.* [248], the methodology undertaken to construct a KG for the purpose of interpreting clinical proteomics<sup>1</sup> data was detailed. The proposed KG, called *clinical knowledge graph* (CKG), comprised approximately 20 million vertices and 220 million edges based on information extracted from experimental data, public data stores as well as relevant literature. The data model corresponding to the CKG is depicted in Figure 4.7. The CKG framework comprised four main steps, namely: *Preprocessing*, *KG construction*, *KG connection*, and *result analysis*. The preprocessing step was responsible for formatting and analysing the available proteomics data and therefore comprises the primary steps of a generic data science pipeline. These steps included data preparation by means of normalisation, filtering, formatting, and imputation as well as data exploration through the generation of statistical metrics and distributions. Data analysis and visualisation also constituted the preprocessing step of the CKG framework. The construction of the graph data store was carried out in the KG construction step which extracted entities, relationships, and properties from the presented data *via* configuration files that provided rules describing the manner in which ontologies and data stores were to be processed. An important step in this process involved defining the data model for the graph

<sup>1</sup>The analysis of the entire protein complement of a cell, tissue, or organism under a specified set of conditions [311].



A heterogeneous clinical KG was constructed from medical domain knowledge obtained from an online medical bibliographic data store and subsequently analysed to determine noteworthy correlative relationships between different diseases and their respective symptoms. Each vertex in the graph corresponded to a concept within a biomedical domain. The construction of the KG was based on the previous work of Pham *et al.* [224], in which biomedical domain knowledge was inferred by mining a corpus comprising citations of various publications. The proposed model was then subjected to empirical experiments which showcased superior performance in respect of previous state-of-the-art baselines.

Lu *et al.* [176] constructed a disease network from a patient-disease KG as part of their proposed framework which employed recommender systems for predictive risk modelling of chronic diseases. A bipartite KG was employed to represent the relationships between patients and diseases, together with a projection thereof onto a monopartite graph in order to obtain a so-called disease network. In their study, six recommender system models were employed, together with pertinent descriptive network features. The task at hand involved determining the severity levels of diseases in order to produce a ranked list of predicted diseases. An algorithmic performance comparison was also carried out in respect of an ablation study in which network features were omitted, from which it was inferred that the inclusion of network features resulted in improved predictive performance with respect to chronic diseases and their latent comorbidities.

In a large number of frameworks pertaining to clinical KGs, one of two distinct aims is typically pursued. Certain frameworks within the literature may be characterised by a systematic and integrated approach towards *constructing* clinical KGs from an abundance of biomedical (often unstructured) data sources. Alternatively, some frameworks pertain solely to a methodological approach for *analysing* established KGs through various graph-based techniques. Furthermore, in the notable work of Pham *et al.* [223] a framework was proposed for constructing a KG for analysis, they formulated the task of disease prediction as a multi-label node classification problem, according to which patient nodes were classified based on different disease labels. A link prediction approach towards deriving clinical insights facilitates the inference of potential relationships between entities pertaining to different vertex types within a clinical KG, as opposed to node classification which is primarily employed to categorise individual nodes based on their inherent attributes.

Consequently, a need is identified to develop a generic end-to-end framework for constructing, analysing, and deriving various insights from a clinical KG through a link prediction approach. In particular, this framework should address the construction of a medical KG from structured and/or unstructured data sources within the medical domain. The construction process should also facilitate the integration of medical ontological data so as to supplement the KG with generic medical information. Two prominent clinical use cases may be induced from such a framework, the first of which relates to condition diagnosis (*i.e.* to predict misdiagnosed conditions or diseases), while the second use case pertains to medication prescription (*i.e.* to predict medication to prescribe). Towards this end, the MEDIKAL framework is proposed.

## 4.6 The MediKAL framework

The proposed framework extends upon the generic data science paradigm, as delineated in Figure 4.3, through the inclusion of modules specific to the relevant domain addressed in this project. These changes may be observed in the second and third components, according to which the *Modelling* component is renamed to *KG Construction*, while the *Deployment* component is renamed to *Analysis*. A high-level schematic of the MEDIKAL framework is presented

in Figure 4.8. The proposed framework represents a generic computational approach towards inferring clinical insights from KGs which are derived from clinical data pertaining to EHRs. The analytical foundation upon which this framework is constructed relates to the application of graph-based methods that explicitly leverage the innate interconnected nature of the clinical data under consideration. Two main KGs are considered in respect of the framework’s implementation, namely Patient KG and Medical ontology KG. To the best of the author’s knowledge, this framework represents a novel contribution as it encapsulates each of the functional components that pertain to clinical data processing, graph-based modelling, as well as the analysis and synthesis thereof so as to derive contextual insight. The framework also includes a database component for data management purposes. Within each component distinct data-driven modules are embedded which are employed towards executing the function of that component. A GUI is omitted from design considerations due to scope delimitations. A detailed discussion on the framework’s design follows hereafter.

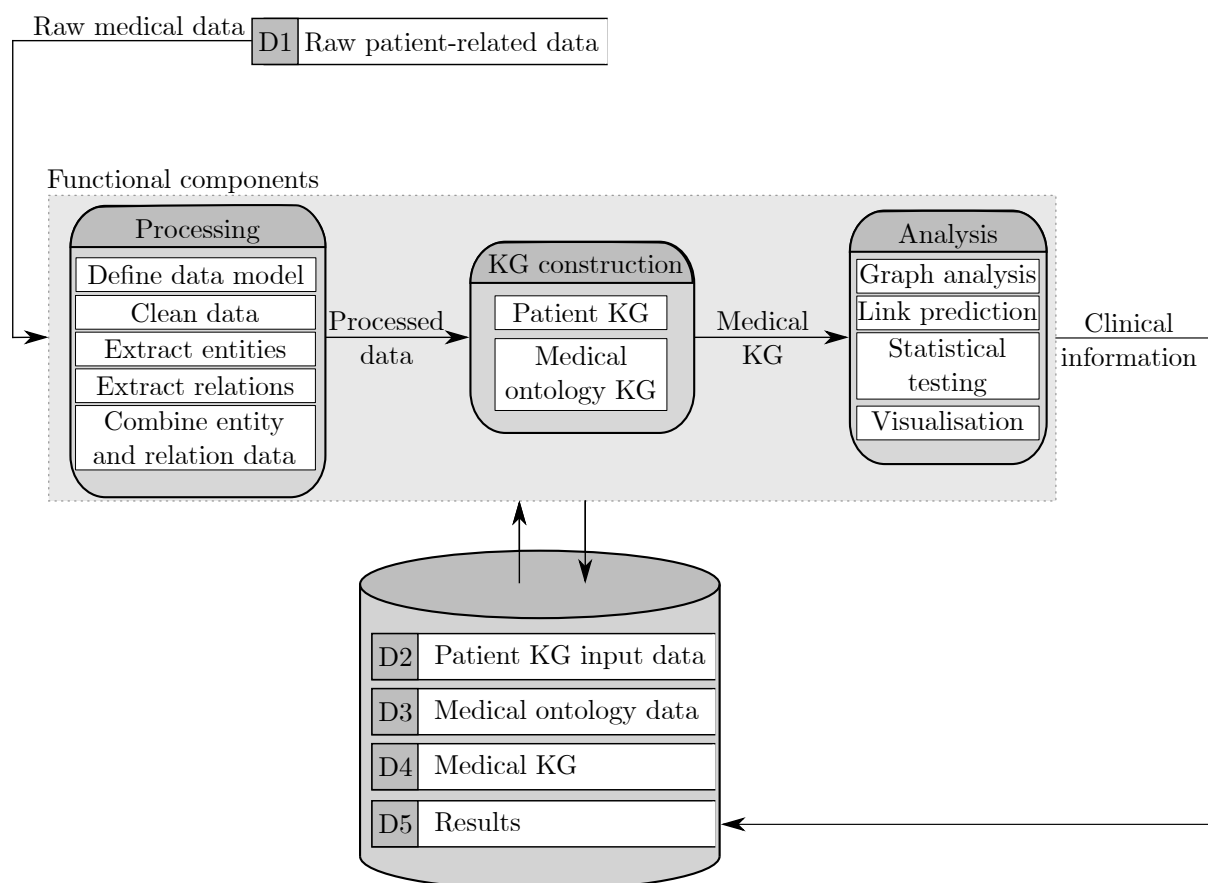


FIGURE 4.8: A high-level schematic of the MEDIKAL framework.

#### 4.6.1 Database component

The MEDIKAL framework comprises five key data stores, labelled D1–D5. The first data store employed by the framework, *Raw patient-related data* (D1), comprises structured and/or unstructured data pertaining to the patients considered for inclusion in the so-called Patient KG. Structured data may be tabular in nature, storing information pertaining to patients together with their associated medical conditions, medications, treatments, and surgical procedures. Other forms of structured medical data sources include patient demographics such as age, gender, height, weight, and blood type [273]. Unstructured data, typically presenting a

more notable data processing challenge, are typically stored in EHRs, expressed as clinical “narrative” data [164]. Examples of unstructured patient-related data include clinical notes compiled by healthcare practitioners, surgical records, patient discharge summaries, and radiology reports, to name but a few [264]. The MEDIKAL framework is amenable to both real-world and synthetically generated data (discussed in §2.3.2), the latter case being the focal point in this thesis (due to data availability matters).

The second data store, *Patient KG input data* (D2), contains the data required for constructing the Patient KG. This data store comprises processed formats of the specific entities and relations that constitute the subsequent Patient KG. The third data store employed in the MEDIKAL framework, *Medical ontology data* (D3), contains medical ontological data which represent the abundance of medical concepts in a standardised and structured manner. As discussed in §2.3.3, popular medical ontologies include *systematised nomenclature of medicine clinical terms* (SNOMED CT) [31] and RXNORM [208]. The data store titled *Medical KG* (D4) stores the final KG which is analysed in the subsequent Modelling component. This KG represents a combination of both patient-specific information as well as normalised medical concepts which facilitates the process of analysing the KG and deriving insights therefrom. The fifth (and final) data store, *Results* (D5), stores pertinent output from the modelling approach, including information pertaining to graph visualisations, model hyperparameters, performance, and visualisations of the results.

## 4.6.2 The Processing component

The first functional component of the MEDIKAL framework, *i.e.* the Processing component, comprises five distinct modules, numbered 1.0 to 5.0, which may be observed in the level-one DFD presented in Figure 4.9. The names of the modules (in ascending order of their respective module numbers) are: *Define graph data model*, *Clean data*, *Extract entities*, *Extract relations*, and *Combine entity and relation data*.

The first step in the processing component, *i.e.* Module 1.0, involves defining the graph data model which forms the foundation upon which the graph data store is structured. The raw data of data store D1 represents the input to this module. The graph data model serves as a generic schema for the graph data store, illustrating the type of nodes that constitute the graph data store, the different relationship types (and their respective directions) that connect these vertices to one another, as well as the properties associated with each vertex and relationship type. The graph data model is therefore essential towards understanding and identifying which entities, relations, and properties are to be extracted from the raw data. In a clinical context, a data model defines the manner according to which different contextual objects (such as diseases, patients, medications, and procedures) relate to one another *via* specific edges. For example, in Figure 4.10, four vertex types are considered, namely: Patient, Disease, Medication, and Surgical procedure, which are connected by means of three relationships, namely: HAS, PRESCRIBED, and UNDERWENT. The graph is further enriched through the addition of node properties which relay information such as the respective names and ontological codes of diseases, medications, and surgical procedures as well as the name, age, and gender of patients within the graph. The module in which the data model is defined therefore represents a conceptual step that facilitates determining the underlying schema of the graph data store. The data model is constructed according to the end requirements of the graph data store and therefore any decision with respect to node, edge, and property classification should align accordingly.

The raw data are cleaned in Module 2.0 which corresponds to the high-level methodology of CRISP-DM’s data preparation phase, as described in §3.1.2. The data may be subjected to

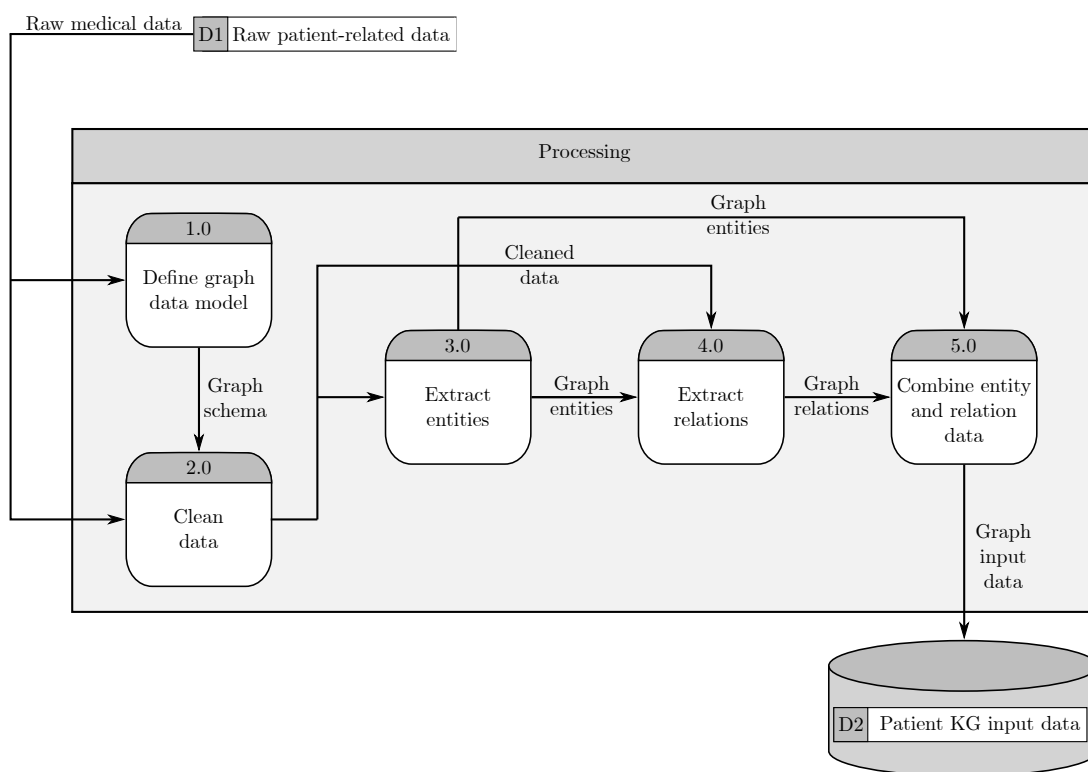


FIGURE 4.9: The level-one DFD of the Processing component.

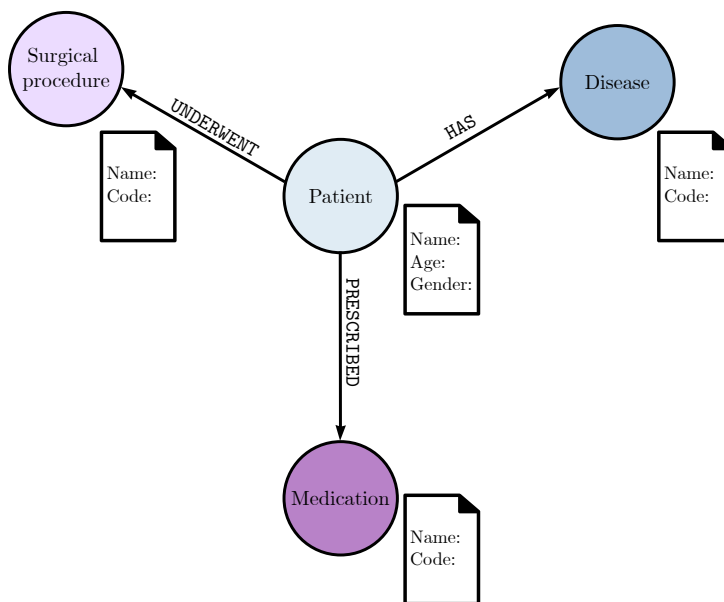


FIGURE 4.10: An example of a KG data model, as contextualised within the clinical domain.

various data cleaning procedures depending on the quality of the raw input data. The raw data employed as input to the MEDIKAL framework contain patient medical records which may be obtained from sources such as EHRs and other similar medical data sources. These patient records typically contain information relating to symptoms, conditions, diseases, medications as well as procedures or treatments. Furthermore, the established graph schema is considered in Module 2.0 in order to determine the specific entities and properties that require data cleaning.



Module 3.0 facilitates the task of extracting entities from the cleaned data. These entities correspond to the nodes of the graph data store which are defined by the graph data model in Module 1.0. Examples of entities may include patients, symptoms, diseases, and medications to name but a few. Entity extraction may be performed using a variety of methods depending on the format of the cleaned data. For example, if the data are unstructured text, such as certain fields in an EHR, NLP techniques such as NER [104] may be employed. NER is a prominent entity detection technique in the domain of NLP that can be employed to analyse unstructured text documents so as to extract entities and assign them to predefined categories [104]. One may consider adopting approaches similar to the pipeline proposed by Li *et al.* [164] (as discussed in §4.5) according to which NER is employed to identify medical entities such as diseases, symptoms, and laboratory results from unstructured medical text which is originally stored in electronic medical records, laboratory information systems, radiology information systems, and other medical sources. If, however, the data are presented in a more structured (typically tabular) format, such as in a `.csv` or `.json`<sup>2</sup> file, (examples of which are presented in Tables 4.1 and 4.2), the entity extraction process is simplified and may be performed by employing simpler procedures, *e.g.* rule-based approaches. In Figure 4.11, an example is depicted of patient data presented in a `.json` format. A patient’s conditions may be extracted from the file by searching through each object, finding the key “condition”, and then compiling a list of all corresponding instances together with the associated properties. If the data are presented in the format expressed in Tables 4.1 and 4.2, the process is simplified even further.

The aim of the subsequent relation extraction process, *i.e.* Module 4.0, is to identify and formalise the relation(s) between entities within the presented data. Recall that the relationship types, along with their respective end-node types and directions, are defined beforehand in Module 1.0. Possible relation types include a `HAS` relationship which connects a patient node to a disease (or symptom) node and a `PRESCRIBED` relationship which connects a patient node to a medication node, as shown in Figure 4.10. The approach adopted for the relation extraction process depends on the format of the cleaned patient data. In the case of EHR data, *i.e.* unstructured text data, a number of approaches may be employed, namely: Rule-based methods based on sentence patterns and trigger verbs [193], shallow ML models [36, 228, 141], or deep learning methods [193, 247]. On the other hand, if the data are presented in a structured format, a different approach may be necessitated. For example, relations may be extracted between columns representing different entities in a `.csv` file, such as in Tables 4.3 and 4.4, which contains a “Count” column indicating the frequency with which a patient experiences a condition or is prescribed a medication.

TABLE 4.1: An example of patient data represented in a tabular `.csv` format obtained from Synthea’s COVID-19 10K synthetic data set [285].

Patient ID	First name	Last name
1ff7f10f-a204-4bb1-aa72-dd763fa99482	Jacinto644	Kris249
9bcf6ed5-d808-44af-98a0-7d78a29ede72	Alva958	Krajcik437
5163c501-353c-4a82-b863-a3f1df2d6cf1	Jimmie93	Harris789
cc3c806f-4a09-4a89-a990-4286450956be	Gregorio366	Auer97

<sup>2</sup>JavaScript object notation.

TABLE 4.2: An example of condition data represented in a tabular .csv format obtained from Synthea’s COVID-19 10K synthetic data set [285].

SNOMED code	Condition name
10509002	Acute bronchitis (disorder)
195967001	Asthma
36971009	Sinusitis (disorder)
840539006	COVID-19

TABLE 4.3: An example of HAS relationship data represented in a tabular .csv format obtained from Synthea’s COVID-19 10K synthetic data set [285].

Patient ID	SNOMED code	Count
0000b247-1def-417a-a783-41c8682be022	248595008	1
0000b247-1def-417a-a783-41c8682be022	25064002	1
0000b247-1def-417a-a783-41c8682be022	386661006	1
0000b247-1def-417a-a783-41c8682be022	49727002	1

TABLE 4.4: An example of PRESCRIBED relationship data represented in a tabular .csv format obtained from Synthea’s COVID-19 10K synthetic data set [285].

Patient ID	RXNORM code	Count
00049ee8-5953-4edd-a277-b9c1b1a7f16b	2001499	1
00049ee8-5953-4edd-a277-b9c1b1a7f16b	477045	1
00049ee8-5953-4edd-a277-b9c1b1a7f16b	310436	1
00049ee8-5953-4edd-a277-b9c1b1a7f16b	849574	1

In Module 5.0, the entity and relation data are then consolidated and represented in a format that is appropriate for constructing the KG. The graph input data are stored in a centralised data store, *i.e.* D2, from which it can be accessed by downstream modules.

The level-two DFD, shown in Figure 4.12, pertains to an expansion of the modules shown in Figure 4.9 providing a more detailed view of the Processing component. Module 1.0 is not, however, disaggregated into child processes as its working is arguably rudimentary.

Module 2.0 is disaggregated into three child processes, numbered 2.1–2.3. Module 2.1 is employed to remove duplicate data instances, while Module 2.2 is responsible for addressing missing values corresponding to omitted or unrealistic values (including outliers), such as a negative value for patient metrics (*e.g.* age, height, and weight) or erroneous medical ontology codes for symptoms and diseases. Finally, Module 2.3 reformats the data into appropriate representations in respect of the subsequent modules. Modules 3.0 and 4.0 are both disaggregated into two child processes each, numbered 3.1 and 3.2 as well as 4.1 and 4.2, respectively. Module 3.1 extracts entities from the cleaned data, after which Module 3.2 aims to extract certain properties associated with these entities, *e.g.* a patient’s age or height, a symptom’s ontology code or severity score. Modules 4.1 and 4.2 are similar to Modules 3.1 and 3.2, but pertain to relations. While Module 4.1 extracts the relations between the identified entities, Module 4.2 extracts properties of the relationship, such as, for example, onset date and time in respect of a HAS relation or the dosage associated with a PRESCRIBED relation. Lastly, Module 5.0 is disaggregated into two child processes, numbered 5.1 and 5.2. Module 5.1 consolidates the entity and relation data into a single file, while Module 5.2 reformats the combined data into a format that enables an effective KG construction process.

```

{
  "patient_id": "P001",
  "name": "John Doe",
  "date_of_birth": "1980-05-15",
  "gender": "Male",
  "height_cm": 180,
  "weight_kg": 75,
  "blood_type": "A+",
  "allergies": ["Penicillin", "Peanuts"],
  "medical_history": [
    {
      "condition": "Hypertension",
      "diagnosis_date": "2010-08-20",
      "treatments": [
        {
          "name": "Lisinopril",
          "start_date": "2010-08-21",
          "end_date": null,
          "dosage": "10mg once daily"
        }
      ]
    },
    {
      "condition": "Chronic Migraine",
      "diagnosis_date": "2015-03-10",
      "treatments": [
        {
          "name": "Sumatriptan",
          "start_date": "2015-03-11",
          "end_date": null,
          "dosage": "50mg as needed"
        }
      ]
    }
  ]
}

```

FIGURE 4.11: An example of a patient's medical history presented in a .json format.

### 4.6.3 The KG construction component

The Medical KG from which clinical insights are to be derived is constructed in the KG construction component which is presented in Figure 4.13. This component comprises two key modules, namely: Construct Patient KG (Module 6.0) and Construct Medical ontology KG (Module 7.0). The main aim is to construct a KG that contains both patient-specific information as well as normalised medical information in order to avoid computational challenges in respect of the constructed graph representations. For example, numerous patients may exhibit flu symptoms that vary in respect of severity, onset times, and other properties. By representing each patient's flu node by means of different properties, unnecessary duplication of information would transpire due to the same overarching concept being expressed repetitively, rendering it more challenging to derive insight from. By linking each patient-specific flu node to a flu node normalised by some medical ontology data store (*e.g.* SNOMED), both patient-specific and general medical information are represented — the extent to which this process is carried out depends, however, on the input data's level of abstraction.

A more detailed representation of the KG construction component is presented in Figure 4.14. Module 6.0 is disaggregated into five child processes, numbered 6.1–6.5, while Module 7.0 is disaggregated into seven child processes, numbered 7.1–7.7. The first step involved in constructing the Patient KG, *i.e.* Module 6.1, involves generating the different nodes constituting the Patient KG. The module takes as input a patient node and edge list from data store D2 and generates nodes for each distinct instance of a medical concept and patient contained within the edge list. The particular format of Module 6.1's output depends on the inherent data object structure of

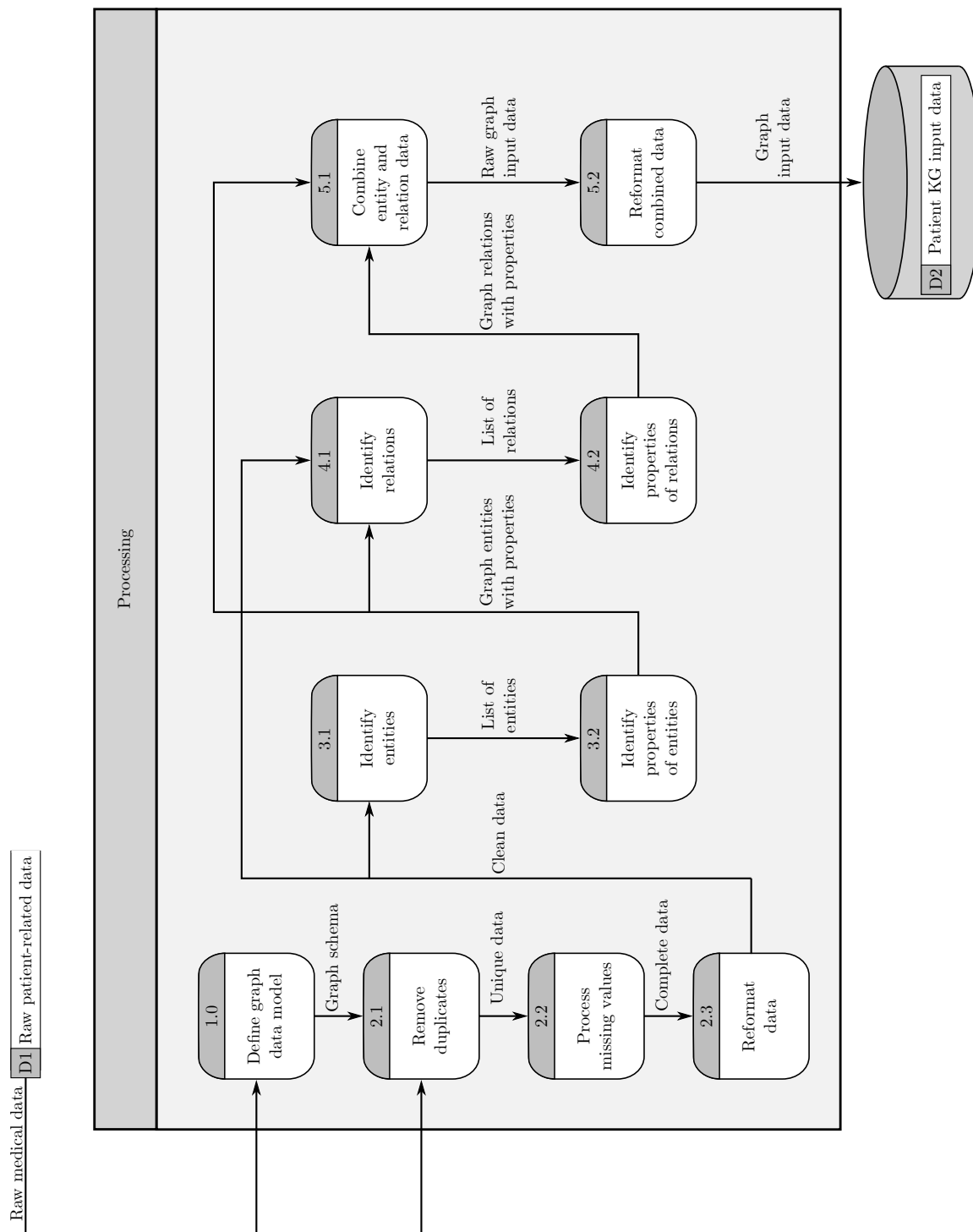


FIGURE 4.12: The level-two DFD of the Processing component.

the chosen graph modelling software. Node features (which may include onset times, level of severity, and descriptions) are subsequently generated in Module 6.2. A larger number of node features can help produce an informative graph data store but may result in increased computational complexity when deriving insights during the execution of downstream modules. Nodes are connected to one another in Module 6.3 which takes as input the edge list stored in data store D2 and constructs relations within the Patient KG. Similarly, in Module 6.4, relation-specific

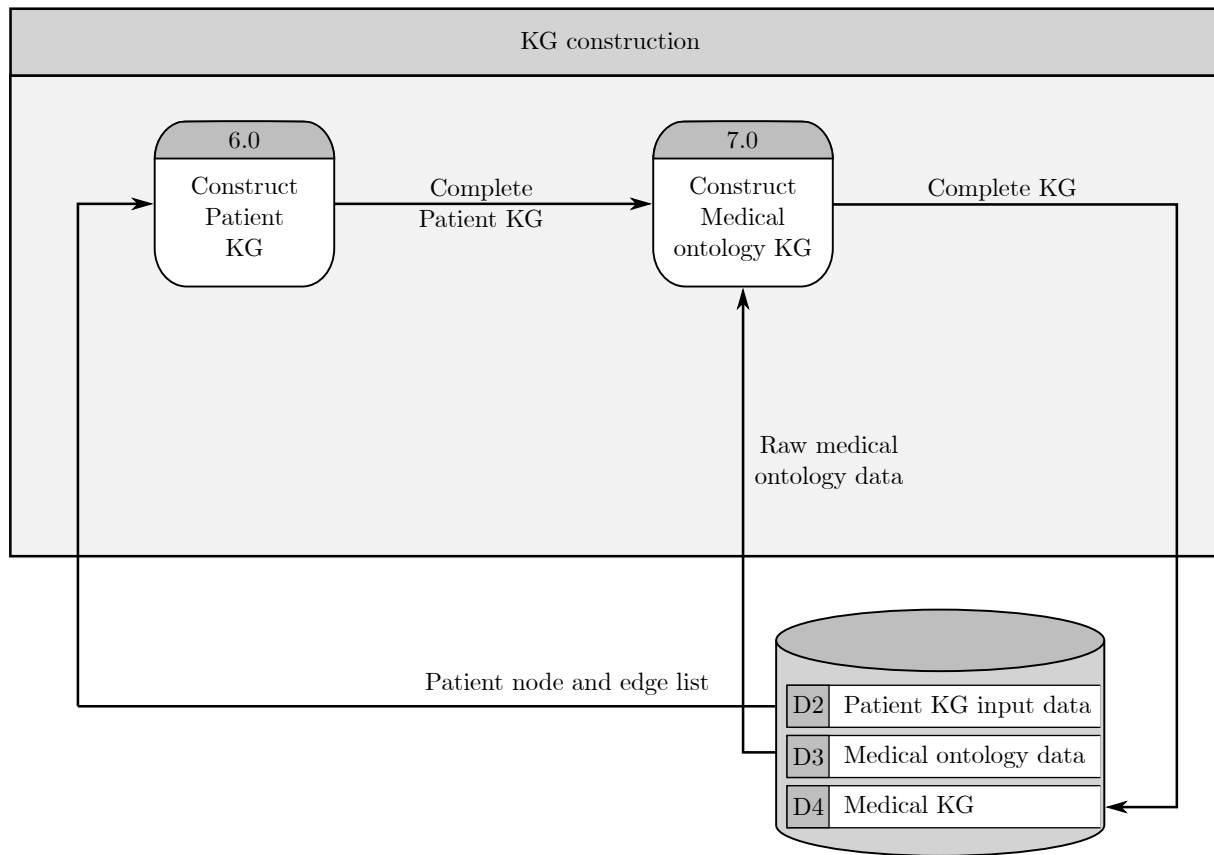


FIGURE 4.13: The level-one DFD of the KG construction component.

features are incorporated so as to enrich the KG further. Finally, in Module 6.5, verification of the graph structure is performed so as to ensure that the KG is both built correctly and conveys the information contained in the input data. This may be achieved by conducting a variety of tests such as counting the number of each node and edges for each type and comparing them with the corresponding input files as well as analysing the edge representations to verify whether the correct nodes are connected to one another.

The extent to which Module 7.0, together with its constituent child processes, is executed depends on the input data stored in data store D2 and the nature of the insights to be derived from the final KG (*i.e.* what the user of the framework seeks to achieve in respect of actionable insight). In some cases, the data used to construct the Patient KG may already contain specific ontological information, such as codes or normalised descriptions, as opposed to more patient-specific information, such as onset times or prescribed dosages.

The aim of analysing the constructed (*i.e.* complete) KG can be extended beyond abstracting patient-specific information to modelling the manner according to which a large data store of patients and medical concepts interlink with one another in a more generalised sense. Modelling that data in this manner facilitates the identification of overarching patterns across a broader range of patients and/or medical entities. If these aforementioned features (*i.e.* ontological entity descriptions or codes) are already present within the input data and the aim of the KG is simply to model a generalised clinical KG, then it may be assumed that the ontological information already forms part of the inherent data structure and there is no need to execute Modules 7.1–7.7 explicitly. If, however, the ontological information is not explicitly described by the input data

and the goal of the complete KG is to express patient-specific information, then Modules 7.1–7.7 are to be carried out in a comprehensive manner, as described hereafter.

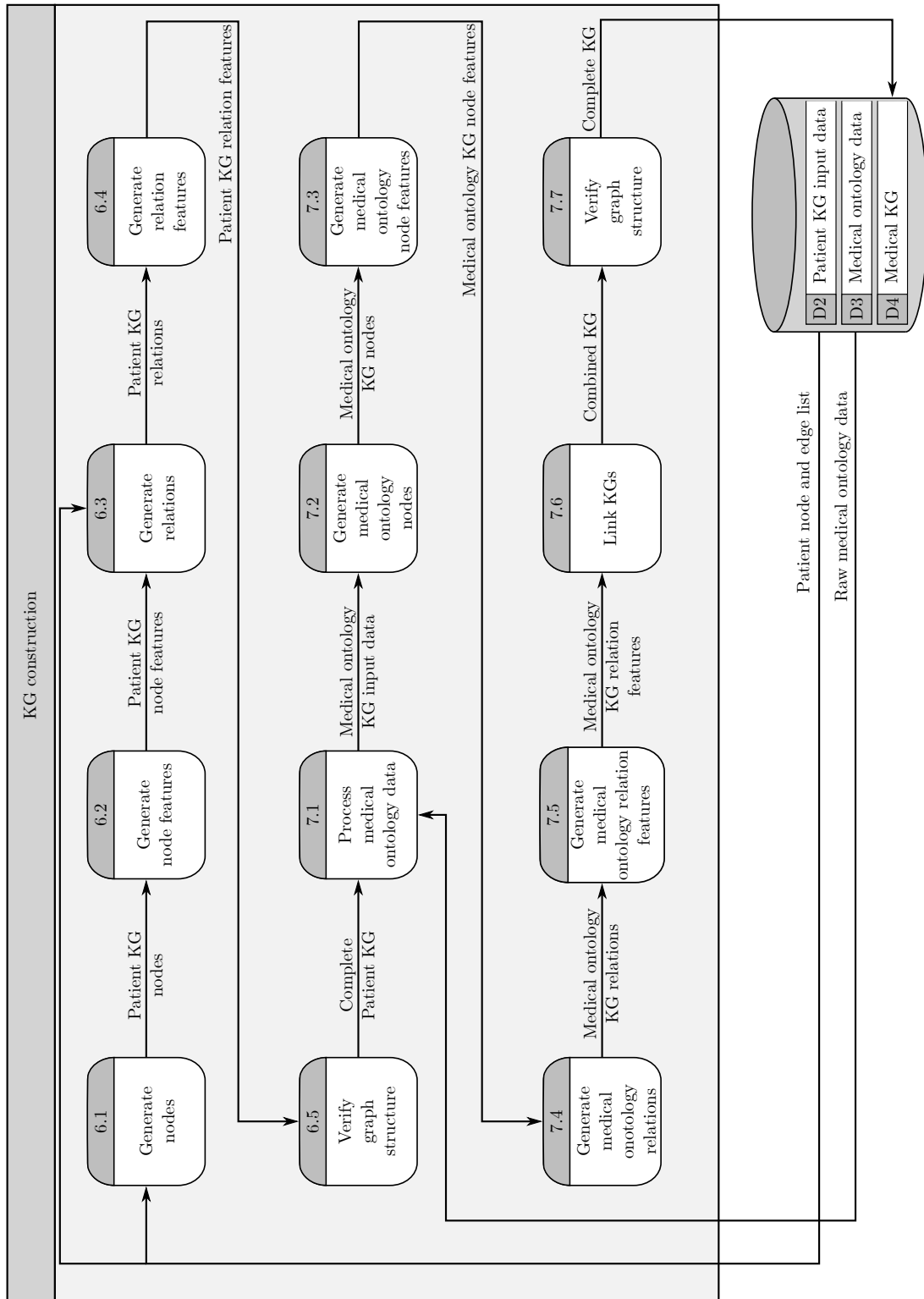


FIGURE 4.14: The level-two DFD of the KG construction component.

The construction of the Medical ontology KG commences in Module 7.1 which takes as input the raw medical ontology data (stored in data store D3) which are processed and transformed into a format that is deemed appropriate for the execution of the downstream modules. Therefore, the raw data should be formatted in a manner that facilitates constructing and combining the Medical ontology KG with the existing Patient KG and subsequently performing graph analysis thereon. After the medical ontology data are processed, a procedure similar to Module 6.1 is performed as part of Module 7.2 according to which the relevant medical ontology nodes are generated. Module 7.3 is responsible for generating node features for these ontology nodes. In Module 7.4, relationships between the various medical ontology nodes are established, while Module 7.5 serves to generate features for these newly established relationships.

An important step relates to the linking of the two KGs (*i.e.* the Patient KG and the Medical ontology KG) which is performed in Module 7.6. Accordingly, patient-specific instances from the Patient KG are linked to normalised medical concepts from the Medical ontology KG which facilitates the simplification of information for computational purposes. By integrating both KGs, information loss is mitigated whilst ensuring accessibility of patient-related information by means of graph queries. Clinical insights can be derived from this more generalised representation of the medical KG. Finally, the complete KG structure is verified in Module 7.7 in a manner similar to Module 6.5, with the inclusion of ontological-based queries. The complete KG, called the Medical KG upon which subsequent analysis is performed, is stored in data store D4.

#### 4.6.4 The Analysis component

The final component of the MEDIKAL framework is responsible for deriving clinical insights from the constructed Medical KG and generating information pertaining to the utility of these insights. A level-one DFD of the Analysis component is presented in Figure 4.15 which comprises four modules, numbered 8.0–11.0. Module 8.0, *i.e.* conduct graph analysis, involves understanding the underlying structure of the Medical KG which can aid the process of selecting a set of appropriate link prediction algorithms. In Module 9.0, link prediction is performed in respect of the constructed Medical KG, from which insight can be derived. Module 10.0 facilitates inferential statistical testing in respect of the algorithmic performance data, while in Module 11.0 visualisations are generated so as to provide further insight into the modelling results.

A more detailed representation of the Analysis component is presented in Figure 4.16 in which the respective child processes of Modules 8.0, 9.0, and 11.0 are depicted. In Module 8.1, graph visualisation is employed in order to facilitate a qualitative understanding of the nature according to which nodes and edges are distributed in the Medical KG. Graph visualisation may be carried out by employing different software libraries which utilise algorithmic approaches for visualising graph-structured data sets. Notable examples of such libraries include GraphViz [83], Cytoscape [256], Gephi [22] and igraph [60]. The specific format according to which the input data to Module 8.1 should be represented varies depending on the selected graph visualisation approach and may therefore require additional processing of the graph data object stored in data store D4. Furthermore, graph data objects constructed by certain libraries can be converted to a different graph data file format and subsequently exported so as to leverage the capabilities of various graph visualisation libraries. Consider, for example, NetworkX [107] which is a software package that facilitates the creation, manipulation, and analysis of the structure, dynamics, and functions of complex graphs. It is, however, considered relatively limited in respect of its visualisation capabilities when compared with the aforementioned libraries. One of the main

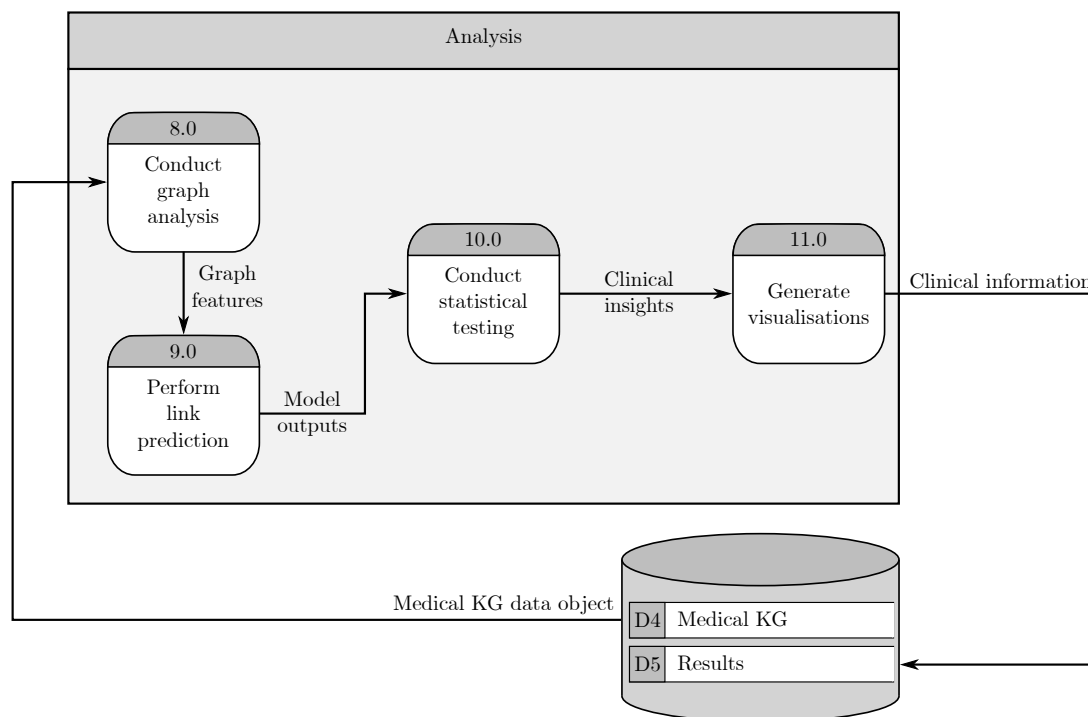


FIGURE 4.15: The level-one DFD of the Analysis component.

advantages of NetworkX is its exporting capabilities in respect of the useful GraphML<sup>3</sup> format which facilitates visualisation by means of libraries such as Cytoscape and Gephi.

Module 8.2 facilitates the task of conducting basic graph feature analysis with the aim of supplementing the insights gained in Module 8.1 with quantitative information pertaining to the graph's structure. The graph feature analyses that may be employed during Module 8.2 include estimating the connectedness of the graph, analysing degree distribution of the nodes, identifying prominent nodes within the graph, and simplifying the graph structure by removing potentially redundant community structures (clusters), to name but a few. Some important (and contextual information) can therefore be extracted, *e.g.* identifying diseases or symptoms that are most frequently experienced by patients, identifying which patients are contracting a large number of illnesses, identifying medications that are predominantly (or excessively) prescribed to patients. The task of removing potentially redundant community structures from the graph necessitates meticulous consideration with respect to phenomena deemed inconsequential within the specific clinical context. In the context of clinical graphs, this may include relatively small clusters of nodes that are entirely disconnected from other nodes.

Clinical insights are derived through the application of link prediction approaches in Module 9.0 which comprises three child processes, numbered 9.1–9.3. As mentioned earlier, two main use cases may be instantiated by means of the framework, *i.e.* condition diagnosis and medication prescription. Depending on the user's aim, the type of investigation carried out governs certain considerations during the execution of this module. In Module 9.1, the user selects a set of link prediction algorithms to apply to the KG. These algorithms may include CN-based approaches, recommender system approaches, classifier-based ML approaches, or deep learning approaches, as discussed in Chapter 3. Selecting an appropriate set of link prediction algorithms requires

<sup>3</sup>GraphML is a mark-up based file format for graph structures which consists of a language core describing the graph's structural properties [35].



an understanding of the data constituting the graph structure as well as the aim of the link prediction task.

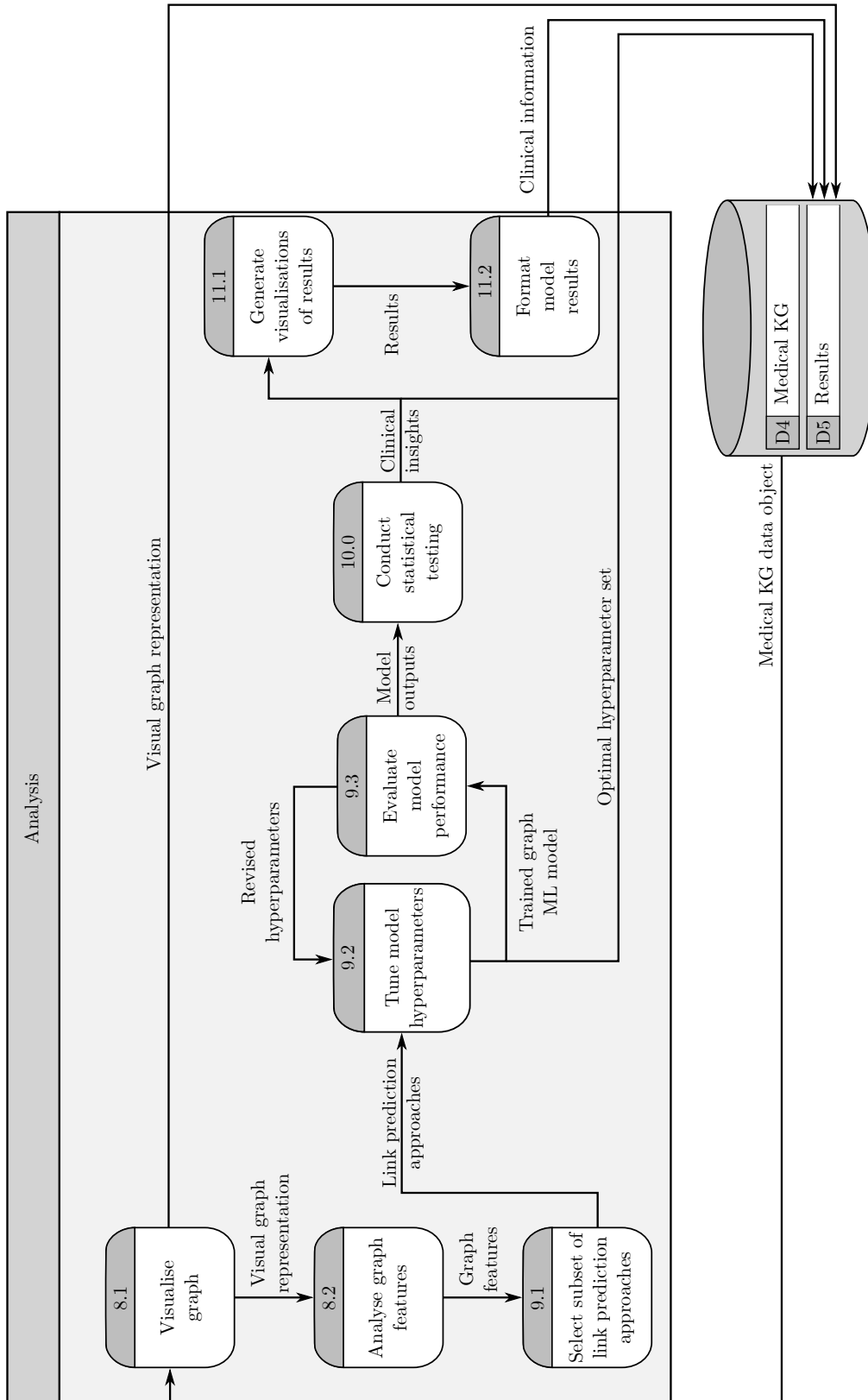


FIGURE 4.16: The level-two DFD of the Analysis component.

For example, graphs comprising a single node type without additional node properties may be better suited for link prediction by means of traditional, heuristic-based methods [4, 137]. Recommender system approaches, on the other hand, can be employed in the case of a weighted bipartite graph [165], however, this is excluded from the scope of this project. Supervised ML approaches may also be employed for performing link prediction — these approaches are especially applicable if the presented data comprise a variety of categorical and continuous features relating to patients. Due to their inherent nature, KGs typically contain detailed and interconnected information and are markedly large in scale rendering deep learning approaches (such as GNNs) potentially more suited when attempting to derive clinical insights — these techniques have been specifically designed to perform computations on large heterogeneous graph-structured data sets [47, 148, 308, 318].

After an appropriate modelling approach is selected, the training process is initiated. The manner according to which data are partitioned varies according to the chosen modelling approach, as discussed in §3.8.2. The intuition is as follows: The set of edges  $\mathcal{E}$  is partitioned into two disjoint sets, namely: A training set  $\mathcal{E}^T$  and a probe (test) set  $\mathcal{E}^P$  by randomly sampling edges (by means of a uniform distribution) without replacement, such that no edges in  $\mathcal{E}^T$  are present in  $\mathcal{E}^P$ , and *vice versa*. In the case of temporal graphs, the data split is obtained by partitioning all edges prior to a particular point in time as training edges and all edges from the selected point onwards as test edges. Supervised ML approaches require that the input data be labelled, therefore negative edges must first be generated before partitioning can occur. This may be achieved by randomly selecting non-connected node pairs or node pairs that span a certain distance (*i.e.* geodesic distance) from one another. Data partitioning in the context of GNNs is performed by selecting supervision edges which serve as supervisory signals for training the GNN in respect of the link prediction task, as detailed in §3.8.2. Moreover, a validation set may be employed should the model require hyperparameter tuning. It is important to note that the selection of edges depends on the use case. More specifically, if condition diagnosis is to be performed, then condition edges within the graph are subjected to the modelling process. If, on the other hand, medication prescription is to be performed, then medication edges within the graph are subjected to the modelling process.

Modules 9.2 and 9.3 are executed in an iterative manner, as indicated by the cyclic arrows in Figure 4.16. Should a desired level of performance not be achieved after evaluation of specific performance metrics and plots, the hyperparameters are revised and the process repeated. Evaluation metrics may include accuracy, precision, AUROC, and AUPRC, as discussed in §3.8. Performance plots such as precision-recall curves and training loss may also be considered. In Module 9.2, hyperparameter tuning may be conducted by means of manual search, grid search, randomised search, or automated hyperparameter tuning. In the context of manual search, the user manually selects the hyperparameter values (based on established judgement and findings in the literature), after which performance is evaluated. Grid search, on the other hand, involves defining a multi-dimensional grid in which each coordinate represents a specific combination of feasible hyperparameter values. The model is then trained and its performance evaluated in respect of each combination. Due to the computational burden imposed by grid search, randomised search may be preferred, according to which combinations of random hyperparameter values are sampled from specified distributions before training and evaluation are performed. Finally, automated hyperparameter tuning involves the application of other advanced techniques such as Bayesian optimisation [196], metaheuristics [122], and gradient-based optimisation methods [238]. The hyperparameters that may be tuned are specific to the chosen modelling approaches, a summary of which is provided in Table 4.5. During each iteration of Modules 9.2 and 9.3's execution, a performance summary detailing the respective scores for the

chosen evaluation metrics (together with the corresponding hyperparameter values) is generated for perusal and further analysis.

TABLE 4.5: A summary of different link prediction approaches and their hyperparameters.

Category	Algorithm	Hyperparameters
Supervised ML	LR	Learning rate, regularisation type, regularisation strength, maximum iterations, tolerance
	NB	Prior probabilities, kernel type, smoothing parameter, binning strategy
	$k$ NN	Number of neighbours, distance metric, weighting function
	DT	Maximum depth, minimum samples split, minimum samples leaf
	RF	Number of trees, maximum depth, minimum samples split, minimum samples leaf, maximum features
	MLP	Number of hidden neurons, learning rate, batch size, activation function, regularisation parameter
Recommender systems	Collaborative filtering	Number of neighbours, similarity measure, user-based or item-based
	Content-based filtering	Number of features, feature weighting, similarity measure
	Matrix factorisation	Learning rate, regularisation strength, number of latent factors, number of iterations
GNNs	GraphSAGE	Number of layers, aggregation type, size of input and hidden channels, activation function
	GAT	Number of multi-head-attentions, size of input and hidden channels, activation function
	GATv2	Number of multi-head-attentions, learning rate, size of input and hidden channels
	Graph transformer	Number of multi-head-attentions, dropout probability, activation function

Statistical testing is conducted in Module 10.0 whereby an appropriate statistical test is selected for the purpose of scrutinising algorithmic performance in a statistically sound manner. Non-parametric tests such as the Friedman test [91] as well as the *post hoc* Nemenyi procedure [209] are more suitable if specific assumptions in respect of the performance data's distribution cannot be made reliably. A sufficiently large sample size (at least twenty to thirty replications) should be generated in respect of each modelling approach so as to ensure statistical robustness — attributable to the stochasticity involved.

After achieving a satisfactory level of performance, the results obtained by the link prediction algorithms (including box plots<sup>4</sup> detailing the performance of the different algorithms with respect to the chosen evaluation metrics) are visualised in Module 11.1. The newly predicted links may also be visualised for contextualisation purposes. For example, it may be beneficial to visualise the most frequently occurring conditions or medications both before and after conducting the link prediction task so as to compare the impact of the predictions on the condition or medica-

<sup>4</sup>Box plots convey more information than the simply using the sample mean and standard deviation [152]. The median, inter-quartile range, minimum and maximum, and outliers (indicated by “o”) are illustrated visually in box plots, providing a more comprehensive representation of the central tendency and spread of the data samples.

tion distribution within the data set. This could be accomplished by means of bar plots or pie charts corresponding to condition or medication information. These visualisations can provide useful insight into the prevalence of conditions or medications within the data set as well as the potential changes associated with the newly predicted links. Certain conditions or medications may increase in prominence after execution of the link prediction task which may suggest that these conditions or medications are under-diagnosed or under-prescribed, respectively, amongst the patient population. Conversely, the prominence of some conditions or medications may diminish which may be indicative of over-representation in the original data set. Moreover, it may be beneficial to implement a colour-coding scheme or to vary the size of vertices based on their degrees as a means of visually (and intuitively) showcasing their relevance within the graph. This approach may be conducted, for example, in respect of patient vertices in order to indicate health risk; in respect of disease vertices in order to indicate frequently contracted diseases; or in respect of medication vertices in order to indicate frequency of prescription.

Finally, in Module 11.2, the derived clinical insights are structured into an appropriate format so as to facilitate both interoperability and perusal by stakeholders. The raw output data stemming from the link prediction algorithms are synthesised and subsequently converted into clinical insights, *i.e.* contextualised in respect of the original aim of the link prediction task. For example, it may be beneficial for healthcare practitioners to be presented with a list of historical and predicted medical entities for each patient which could facilitate the task of verifying whether an underlying medical relationship is present between these historical and predicted entities. The results are subsequently stored in data store D5 which represent the basis of the clinical decision support.

## 4.7 Design verification

The design of the MEDIKAL framework conforms to the guidelines proposed by Kendall and Kendall, as discussed in §4.4, in order to ensure the development of a high-quality, end-to-end pipeline. During the developmental stages, a top-down modular approach has been employed to conceptualise the overarching aim of transforming raw patient-related data into clinical insights by means of a link prediction approach. This approach is exemplified by means of the diagrammatic illustrations in Figure 4.8 — a high-level schematic of the MEDIKAL framework is presented showcasing the flow of information between the main functional components and their constituent modules. This schematic is then further expanded into individual level-one DFDs for each functional component which, in turn, are further accompanied by level-two DFDs. In §4.6, a detailed elucidation of the data stores, functional components, and modules constituting the MEDIKAL framework is provided. Consequently, adequate documentation (and accompanying illustrations) pertaining to the methodologies underpinning the MEDIKAL framework have been proffered for the sake of verifying the correctness of its design. Furthermore, the design of various components and the selection of their constituent modules (such as the Processing component and its constituent modules as well as the Medical ontology KG module in the KG construction component) are based on related approaches in the literature, as discussed in §4.5. The incorporation of functional notions and design considerations from well-established studies in the literature contributes towards the MediKAL framework's verification (from a design perspective) as it is grounded in reputable methodologies.

## 4.8 Chapter summary

This chapter opened in §4.1 with a formal introduction to the notion of a framework as well as an overview of its developmental process. In §4.2, DFDs were discussed in respect of its informative and intuitive nature. Thereafter, the generic data science paradigm, which represents the methodological foundation of the MEDIKAL framework, was discussed in §4.3, while in §4.4, approaches aimed towards quality assurance during the development of the framework were discussed. In §4.5, a discourse on related frameworks was presented in order to provide the reader with background on notable work pertaining to clinical KGs. Finally, a detailed discussion of the MEDIKAL framework was presented in §4.6, in which its different functional components were elucidated. The discussion was supplemented with DFDs to facilitate an improved understanding of the manner according to which the framework may be executed. Thereafter, a discussion pertaining to design verification was presented in §4.7.

---

---

## CHAPTER 5

---

# Framework implementation

### Contents

5.1	Algorithmic verification . . . . .	102
5.2	Clinical data set background . . . . .	103
5.3	First framework instantiation . . . . .	104
5.3.1	<i>Processing component</i> . . . . .	105
5.3.2	<i>KG construction component</i> . . . . .	106
5.3.3	<i>Analysis component</i> . . . . .	108
5.4	Second framework instantiation . . . . .	121
5.4.1	<i>Processing component</i> . . . . .	122
5.4.2	<i>KG construction component</i> . . . . .	123
5.4.3	<i>Analysis component</i> . . . . .	123
5.5	Third framework instantiation . . . . .	128
5.5.1	<i>Processing component</i> . . . . .	129
5.5.2	<i>KG construction component</i> . . . . .	130
5.5.3	<i>Analysis component</i> . . . . .	130
5.6	Reflection . . . . .	135
5.7	Methodological utility of MEDIKAL framework . . . . .	136
5.8	Chapter summary . . . . .	137

The purpose in this chapter is to demonstrate the functional working of the MEDIKAL framework proposed in this thesis by implementing different instantiations thereof in respect of synthetic data sets within the clinical domain. First, a discourse pertaining to computational verification is presented in order to establish the functional correctness of the algorithmic link prediction methods employed — a well-regarded link prediction data set (albeit unrelated to a clinical context) forms the basis of this endeavour. Background information is then provided with respect to the main clinical data sets under consideration and the nature of the insights to be derived from the Medical KG constructed by means of the MEDIKAL framework’s instantiations. Thereafter, the framework’s implementation is discussed in detail with respect to each component and its constituent modules. Three different computerised instantiations are carried out, each of which differs in respect of the data considered and/or the clinical use case. The main findings derived from the three instantiations are subsequently synthesised and presented. This is followed by a discourse pertaining to the framework’s validation by a subject matter expert. The chapter concludes with a summary of its contents.

## 5.1 Algorithmic verification

In this section, algorithmic verification is conducted in respect of certain link prediction algorithms that are considered as part of the computerised instantiations presented in the remainder of the chapter. The aim during this algorithmic verification process is therefore to ensure that the computerised instantiation of Module 9.0 is functionally correct. Focus is specifically placed on algorithms that stem from less established (and documented) approaches in the literature. More specifically, the CN-based link prediction algorithms in respect of bipartite graphs, proposed by Aziz *et al.* [17] (discussed in §3.6), are subjected to algorithmic verification. This may be ascribed to the limited<sup>1</sup> reported usage of their MATLAB code, as published in their original work. The GNNs considered in this thesis are also subjected to algorithmic verification — attributable to the limited reported usage of the PyG [87] library employed in this thesis.

Aziz *et al.* considered drug-target interaction networks (modelled as bipartite graphs) to evaluate the performance of their proposed approach which was compared with conventional CN-based link prediction algorithms. A drug-target interaction network is an abstraction of the structured manner according to which drug molecules interact with target proteins [17]. Furthermore, four drug-target interaction networks were employed, namely: *Nuclear receptors* [136], *G-protein-coupled receptors* (GPCRs) [252], *ion channels* [106], and *enzymes* [297], each of which is publicly available and considered benchmark data sets for link prediction algorithms for bipartite graphs [72]. Each data set is presented in the format of an edge list<sup>2</sup> (described in §2.1.3). An overview of the structural properties pertaining to the four networks is presented in Table 5.1.

TABLE 5.1: *The structural properties pertaining to the four networks employed during algorithmic verification of the CN-based algorithms.*

Data set	$ \mathcal{V} $	$ \mathcal{V}_1 $	$ \mathcal{V}_2 $	$ \mathcal{E} $	$\bar{d}(\mathcal{V}_1)$	$\bar{d}(\mathcal{V}_2)$
Nuclear receptors	80	26	54	90	3.46	1.67
GPCR	318	95	223	635	6.68	2.85
Ion channels	414	204	210	1476	7.24	7.03
Enzymes	1109	664	445	1109	1.67	2.49

Aziz *et al.* partitioned the data sets into a training set and a probe set randomly<sup>3</sup> based on a uniform distribution — a 90%:10% split was adopted in respect of the training and probe sets, respectively. The training set and probe set were subsequently employed to calculate the AUROC score of the link prediction algorithms. Aziz *et al.* employed a modified approach towards estimating AUROC due to computational limitations. The mean AUROC scores achieved by the algorithms, expressions of which correspond to (3.16)–(3.21), in respect of 100 replications are presented in Table 5.2.

In the case of the verification study carried out in this thesis, the MATLAB code published by the authors [16, 62] was implemented in respect of the computational environment discussed in the project scope, *i.e.* §1.3 The data sets were partitioned according to the same split adopted by the original authors. Only thirty replications were carried out due to the limited empirical variation in algorithmic output observed. The mean AUROC scores achieved are presented in Table 5.3. It may be argued that these results are comparable to the results achieved by Aziz *et al.* (as summarised in Table 5.2) thereby demonstrating a reasonably sound algorithmic

<sup>1</sup>When compared with the ubiquitous Scikit-learn library [220] employed for the classifier-based link prediction algorithms.

<sup>2</sup>The edge lists pertaining to each data set is presented in the form of a `.txt` file.

<sup>3</sup>The authors did not provide a random seed and therefore an exact replication of the results is not possible.


TABLE 5.2: The reported AUROC scores in respect of the CN-based link predictions algorithms of Aziz *et al.* [17] in respect of four drug-target interaction networks.

	Nuclear receptor	Ion channels	GPCR	Enzymes
LCL	0.6970	0.9131	0.8396	0.8850
RA	0.6961	0.9150	0.8499	0.8857
CRA	0.6977	0.9233	0.8487	0.8867
CAA	0.6977	0.9233	0.8487	0.8867
CAR	0.6969	0.9117	0.8402	0.8850
PRA	0.6984	0.9280	0.8521	0.8872

implementation in respect of the framework’s computerised instantiation. It is important to note that Aziz *et al.* only evaluate the performance of the link prediction algorithms with respect to the AUROC metric which, according to Yang *et al.* [306] (§3.8), is not particularly suited for the evaluation of link prediction algorithms due to the imbalanced nature of link prediction data sets.

TABLE 5.3: The AUROC scores achieved in respect of Module 9.0’s implementation of the CN-based link predictions algorithms of Aziz *et al.* [17] in respect of four drug-target interaction networks.

	Nuclear receptor	Ion channels	GPCR	Enzymes
LCL	0.6903	0.9096	0.8301	0.8903
RA	0.6900	0.9115	0.8394	0.8910
CRA	0.6912	0.9197	0.8385	0.8921
CAA	0.6911	0.9158	0.8356	0.8914
CAR	0.6901	0.9081	0.8306	0.8903
PRA	0.6941	0.9262	0.8348	0.8893

The GNNs considered in this study, as delineated in §3.5.4, were implemented by means of the  *Pytorch Geometric*<sup>4</sup> (PyG) library which is an open-source library dedicated towards performing deep learning on graph structures, and is accompanied by extensive documentation and active support communities. PyG is, however, relatively nascent, especially when compared with established libraries such as Scikit-learn [220]. Consequently, additional verification is necessitated so as to ensure the functional correctness of its computerised instantiation (as part of Module 9.0). The PyG library was verified in respect of the established MovieLens [112] data set, the task of which relates to predicting links between movie and user vertices. A total of 100 000 vertices constituted this data set. An AUROC score of 0.9231 was achieved by implementing one of the GNN algorithms, more specifically the SAGE algorithm, which is comparable to the score of 0.9299 achieved by another author Lenssen [161]. The programmatic similarity of the different GNN algorithms within the PyG library renders this single implementation sufficiently representative. The computerised implementation of the GNN algorithms considered in this chapter may therefore be regarded as correct.

## 5.2 Clinical data set background

Due to the challenges associated with real-world clinical data (such as availability and privacy concerns), the MEDIKAL framework is implemented in respect of synthetic EHRs generated by

<sup>4</sup>Pytorch Geometric is a Python library for building and training GNNs for various use cases [87].



Synthea [284] (discussed in §2.3.2). The main aim during the implementations proffered in this chapter is two-fold — first, to verify correct functionality of the various computational modules constituting the framework and second, to showcase a proof-of-concept during which the utility of the proposed framework is demonstrated. The implementations are not explicitly concerned with inferring real-world clinical insight that can be directly employed towards carrying out certain decisions (*e.g.* diagnoses or prescriptions). Instead, a more methodological-based focus is adopted during the framework’s implementations. Consequently, the consideration of synthetic data may therefore be deemed appropriate. The utility of the framework is to be demonstrated in respect of a generic approach towards carrying out different use cases, *e.g.* diagnosis prediction and medication prescription.

Two synthetic data sets are considered for the MEDIKAL framework’s different instantiations, the first of which is the so-called *1K sample synthetic patient records* [284] data set and the *COVID-19 10K* [285] data set. The former data set, henceforth referred to as 1K for simplicity, comprises historical clinical information of approximately 1 000 patients across a range of generic healthcare contexts. In this data set, emphasis is not placed on any particular medical condition and instead a more generic clinical context is represented. The COVID-19 10K data set, on the other hand, comprises approximately 10 000 patients with upper respiratory-related symptoms, the majority of which have been diagnosed with COVID-19, therefore a particular area of focus within the clinical domain is considered. The COVID-19 10K data set is henceforth referred to as 10K for simplicity. The two data sets differ in terms of scope which facilitates more insightful analyses.

The aim during the first instantiation is to conduct link prediction in respect of condition diagnosis by deriving inferential relationships and patterns between patients and conditions that are embedded within the Medical KG. Furthermore, the complexity of this constructed KG is purposefully inhibited so as to facilitate an investigation of the impact that additional data (and complexity) can effectuate, as part of the second and third instantiation. The aim during the second instantiation is to conduct link prediction in respect of prescribing new medications to patients by leveraging the inherent structural properties of a graph constituting patient, condition, and medication vertices. Lastly, in the third instantiation, the Medical KG is supplemented with additional ontological information derived from SNOMED’s hierarchical structure. The link prediction task is once again performed in respect of prescribing new medications to patients, however, both patient-specific and generalised ontological information are considered.

### 5.3 First framework instantiation

In this section, a computerised instantiation of the MEDIKAL framework is demonstrated with respect to its components and the constituent modules, the aim of which is to verify functional correctness and, consequently, demonstrate the utility of the framework. Furthermore, the nature of the framework’s Analysis component facilitates an extensive algorithmic performance analysis in respect of the various link prediction algorithms considered. The framework is implemented within the Python 3.9 [278] software environment — a decision attributable to the language’s extensive support for a range of computational libraries in respect of data processing, data analysis, graph ML, and data visualisation, to name but a few. As mentioned earlier, the aim during this instantiation is to demonstrate the framework’s utility in respect of condition diagnosis, *i.e.* to predict misdiagnosed symptoms or diseases.

### 5.3.1 Processing component

The first component of the MEDIKAL framework, *i.e.* the Processing component, receives as input the raw-patient related data generated by Synthea. Particulars pertaining to the format of the raw data are discussed later. The first module within the Processing component is dedicated to defining the graph data model, *i.e.* the schema according to which the Medical KG is constructed. As mentioned in §4.6.2, the task of defining the graph data model is contingent on the aim of the analyses performed in respect of the constructed Medical KG. Accordingly, analyses are to be performed in respect of a data set comprising patients that have one (or more) conditions, the aim of which is to predict potential missing or latent links as a means of decision support. Towards this end, the graph data model formulated is a bipartite graph comprising a patient vertex, a condition vertex, and a HAS relationship, as shown in Figure 5.1.

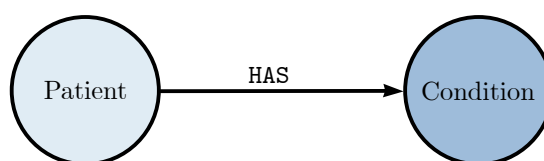


FIGURE 5.1: The graph data model employed in the first framework implementation which comprises a patient vertex, a condition vertex, and a HAS relationship.

In Module 2.0, the raw data are cleaned, as part of the CRISP-DM methodology. Synthea generates an individual `.csv` file for a variety of medical entities such as patients, conditions, allergies, medications, and procedures, however, in this instantiation, only the patient and condition `.csv` files are considered. A `patients1K.csv` file is generated containing a distinct ID as well as information pertaining to the gender, age, marital status, address, and health insurance for each patient. Another file called `conditions1K.csv` file is generated which contains a list of all conditions experienced by patients. Each condition is labelled according to a distinct SNOMED code and description. Recall from §2.3.3 that SNOMED comprises a hierarchical structure of clinical terms that may be employed towards standardising clinical documentation. Accordingly, the various conditions experienced by a patient may be identified by performing a search for that particular patient’s ID. An extract of the `conditions1K.csv` file is provided in Table 5.4. The raw data are then cleaned according to Modules 2.1 and 2.2. Module 2.3 is deemed unnecessary due to the conformity of the `.csv` format in respect of downstream tasks.

TABLE 5.4: An extract of the `conditions1K.csv` file in which the conditions experienced by various patients may be observed. The entries corresponding to the “Code” and “Description” column are derived from the SNOMED medical ontology.

Start	Stop	Patient	Code	Description
2019-02-15	2019-08-01	f0f3bc8d-ef38-49ce-a2bd-dfdda982b271	65363002	Otitis media
2019-10-30	2020-01-30	f0f3bc8d-ef38-49ce-a2bd-dfdda982b271	65363002	Otitis media
2020-03-01	2020-03-30	f0f3bc8d-ef38-49ce-a2bd-dfdda982b271	386661006	Fever (finding)
2020-03-01	2020-03-01	f0f3bc8d-ef38-49ce-a2bd-dfdda982b271	840544004	Suspected COVID-19
2020-03-01	2020-03-30	f0f3bc8d-ef38-49ce-a2bd-dfdda982b271	840539006	COVID-19
2020-02-12	2020-02-26	067318a4-db8f-447f-8b6e-f2f61e9baaa5	44465007	Sprain of ankle
2020-03-13	2020-04-14	067318a4-db8f-447f-8b6e-f2f61e9baaa5	49727002	Cough (finding)

Entity extraction is subsequently performed by means of Modules 3.1 and 3.2. The entities constituting the Medical KG, as defined in the graph data model provided in Figure 5.1, are patients and conditions. Each patient (represented by their ID) is extracted from the `patients1K.csv` file, and stored in a separate data object. The conditions are extracted by identifying each dis-

tinct condition (represented by their distinct SNOMED code and description) within the *conditions1K.csv* file, and are similarly stored in a separate data object. This process is performed in respect of both data sets and each entity, resulting in four new *.csv* files, namely: *patient\_nodes1K.csv*, *patient\_nodes10K.csv*, *condition\_nodes1K.csv*, and *condition\_nodes10K.csv*. The process of relation extraction is conducted in a similar manner by means of Module 4.0, whereby the distinct conditions corresponding to each patient ID are grouped, counted, and exported to respective files, named *has1K.csv* and *has10K.csv* — a sample of which may be observed in Table 5.5. Lastly, Modules 5.1 and 5.2 are carried in order to combine the entity and relation data. This represents the conclusion of the Processing component’s implementation.

TABLE 5.5: An extract of the *has1K.csv* file in which each row denotes a patient-condition vertex pair corresponding to the respective patientID and SNOMED code.

patientID	conditionCode
0042862c-9889-4a2e-b782-fac1e540ecb4	10509002
0042862c-9889-4a2e-b782-fac1e540ecb4	195662009
0042862c-9889-4a2e-b782-fac1e540ecb4	65363002
0047123f-12e7-486c-82df-53b3a450e365	10509002
0047123f-12e7-486c-82df-53b3a450e365	444814009
0047123f-12e7-486c-82df-53b3a450e365	74400008

### 5.3.2 KG construction component

The KG construction component comprises two main modules, namely: Construct patient KG (Module 6.0) and Construct Medical ontology KG (Module 7.0), both of which comprise similar child processes. The KG in this instantiation is constructed as a PyG `HeteroData` object in order to facilitate the subsequent application of GNNs. The `HeteroData` object represents a description of a heterogeneous graph native to the PyG library which is capable of representing multiple vertex and edge types (including attributes) in the format of disjunct storage objects (similar to a regular nested Python dictionary data structure). The construction of the Patient KG is conducted in a similar manner for both data sets. In order to generate the vertices and their associated properties, as performed by means of Modules 6.1 and 6.2, a function generates a mapping<sup>5</sup> for all instances within the entity type *.csv* files, whilst encoder functions are applied to the columns containing the different properties for each (row) instance. The corresponding output is a vertex-level feature representation for each instance within the *.csv* file. For example, consider the input files *condition\_nodes1K.csv* and *condition\_nodes10K.csv*, comprising two columns, titled: *conditionCode* and *conditionName*, where *conditionName* contains a text description for a condition corresponding to a particular SNOMED code in *conditionCode*. An extract of the *condition\_nodes10K.csv* file’s contents are presented in Table 5.6. The distinct SNOMED codes in the *conditionCode* are employed to generate a condition vertex mapping for the condition vertices, while the *conditionName* column is subject to a `Sentence-transformer` [233] encoder<sup>6</sup> in order to generate text embeddings for each condition’s description. The embeddings therefore represent the features of the condition vertices, whereby similar condition descriptions are represented by similar embeddings.

<sup>5</sup>In this context, “mapping” refers to the task of assigning a distinct numerical identifier to each instance within the entity type, thereby facilitating a systematic manner for referencing individual data entries.

<sup>6</sup>Text encoders such as `Sentence-transformer` employ a tokenizer to decompose text data into tokens, *i.e.* fundamental units such as words, phrases, or punctuation marks. These tokens are subsequently employed by the encoder to generate vector representations capable of capturing the underlying semantics of text data [191].

TABLE 5.6: An extract of the data contained within *condition\_nodes10K.csv* which comprises only the SNOMED code and the description corresponding to each condition within the data set.

conditionCode	conditionName
10509002	Acute bronchitis (disorder)
109838007	Overlapping malignant neoplasm of colon
110030002	Concussion injury of brain
124171000119105	Chronic intractable migraine without aura
126906006	Neoplasm of prostate
127013003	Diabetic renal disease (disorder)
127295002	Traumatic brain injury (disorder)
128613002	Seizure disorder

A separate function is employed in order to generate the **HAS** relationship which connects the patient and condition vertices with one another. The function takes as input the *has1K.csv* file and subsequently generates the edges required to construct the KG by connecting the patient mappings (source) to the condition mappings (destination) according to the rows of *has1K.csv*. This step concludes Module 6.3. Module 6.4 is not applicable due to the absence of relation features during this specific instantiation. In Module 6.5, the constructed KG was verified by inspecting the **HeteroData** object. Towards demonstrating this process, consider the 10K patient data set — its graph data model comprises a patient vertex and condition vertex as well as a **HAS** relation. A simple example of output generated from the mapping process is

$$\{10509002: 0, 109838007: 1, 110030002: 2, 124171000119105: 3, \dots\},$$

where each key (vertex label) represents the SNOMED code. In order to generate edges, the relevant function receives as input a *.csv* file (in the format of *has10k.csv*) and generates a two-dimensional **tensor** in which the first row contains the mapping of the source vertex of each edge and the second row contains the mapping of the destination vertex of each edge, *i.e.* each column denotes an edge. This function is then employed separately to generate each edge type. An extract of the output is

$$\begin{bmatrix} 0 & 0 & 0 & \dots & 12351 & 12351 & 12351 \\ 47 & 49 & 81 & \dots & 77 & 162 & 163 \end{bmatrix}, \quad (5.1)$$

where the top row denotes the patient mappings and the bottom row denotes the condition mappings. It may be observed from (5.1) that there is an edge between patient 0 and condition 47, patient 0 and condition 49, patient 0 and condition 81, and so forth. This tensor can subsequently be employed to confirm the validity of the edges by comparing whether the mappings of the vertices in each edge correspond to the entities within the original *has10k.csv* file. For example, the first three and last three edges in the *has10k.csv* file are presented in Table 5.7.

The corresponding patient and condition mappings are presented below, from which it may be observed that the edges in the **HeteroData** convey the same information as the input file.

$$\begin{aligned} &\{0000b247-1def-417a-a783-41c8682be022: 0\} \\ &\{ffea1d41-7562-499d-af1b-284ac72a7c3c: 12351\} \\ &\{248595008: 47\}; \{25064002: 49\}; \{386661006: 81\} \\ &\{36955009: 77\}; \{840539006: 162\}; \{840544004: 163\}. \end{aligned}$$

TABLE 5.7: An extract of the *has10k.csv* file in which each row denotes a patient-condition vertex pair corresponding to the respective patientID and SNOMED code.

patientID	conditionCode
0000b247-1def-417a-a783-41c8682be022	248595008
0000b247-1def-417a-a783-41c8682be022	25064002
0000b247-1def-417a-a783-41c8682be022	386661006
⋮	⋮
ffea1d41-7562-499d-af1b-284ac72a7c3c	36955009
ffea1d41-7562-499d-af1b-284ac72a7c3c	840539006
ffea1d41-7562-499d-af1b-284ac72a7c3c	840544004

As discussed in §4.6.3, the extent to which Module 7.0 is executed depends on the nature of the input data and the aim of the analyses in respect of the Medical KG. In this particular instantiation, the input data contain ontological information. Furthermore, the purpose of the Medical KG is to provide a generic clinical network structure. Consequently, the execution of Modules 7.1–7.7 is not warranted and therefore it is omitted from this discussion.

### 5.3.3 Analysis component

The Analysis component is initiated by means of Module 8.0, the aim of which is to enable the user (of the framework) to gain a qualitative and quantitative understanding of the constructed Medical KG which precedes the selection and application of link prediction algorithms. In Module 8.1, graph visualisations were generated in order to illustrate the topology of the Medical KG. Graph visualisations were conducted by means of a combination of NetworkX and Gephi. Both KGs were first constructed in NetworkX and subsequently exported as a *.gexf*<sup>7</sup> file for input to Gephi. Graph visualisations can subsequently be created — in this case, a force-based layout was adopted. The size of the vertices are based on their degree, according to which their size increases as their degree increases. Consequently, insight into the degree distribution of vertices can be gleaned. For example, it may be observed that for both the 1K and 10K data sets there are certain conditions that manifest more frequently amongst the respective patient populations. The constructed graphs, *i.e.* the 1K data set and the 10K data set, may be observed in Figures 5.2 and 5.3, respectively. In Module 8.2, simple graph feature analyses are conducted in order to gain a quantitative understanding of the overall graph structure — a summary of which is presented in Table 5.8.

TABLE 5.8: A summary of the graph features computed in Module 8.2 of the first instantiation.

Graph feature	1K data set	10K data set
Number of patient vertices $ \mathcal{V}_p $	1 152	12 165
Number of condition vertices $ \mathcal{V}_c $	123	178
Total number of vertices $ \mathcal{V} $	1 275	12 343
Total number of edges $ \mathcal{E} $	6 990	113 239
Average degree of patient vertices $\bar{d}(\mathcal{V}_p)$	6.07	9.31
Average degree of condition vertices $\bar{d}(\mathcal{V}_c)$	56.83	636.17

<sup>7</sup>Graph exchange XML format (*.gexf*) is a markup language-based file format that describes (often complex) network structures.

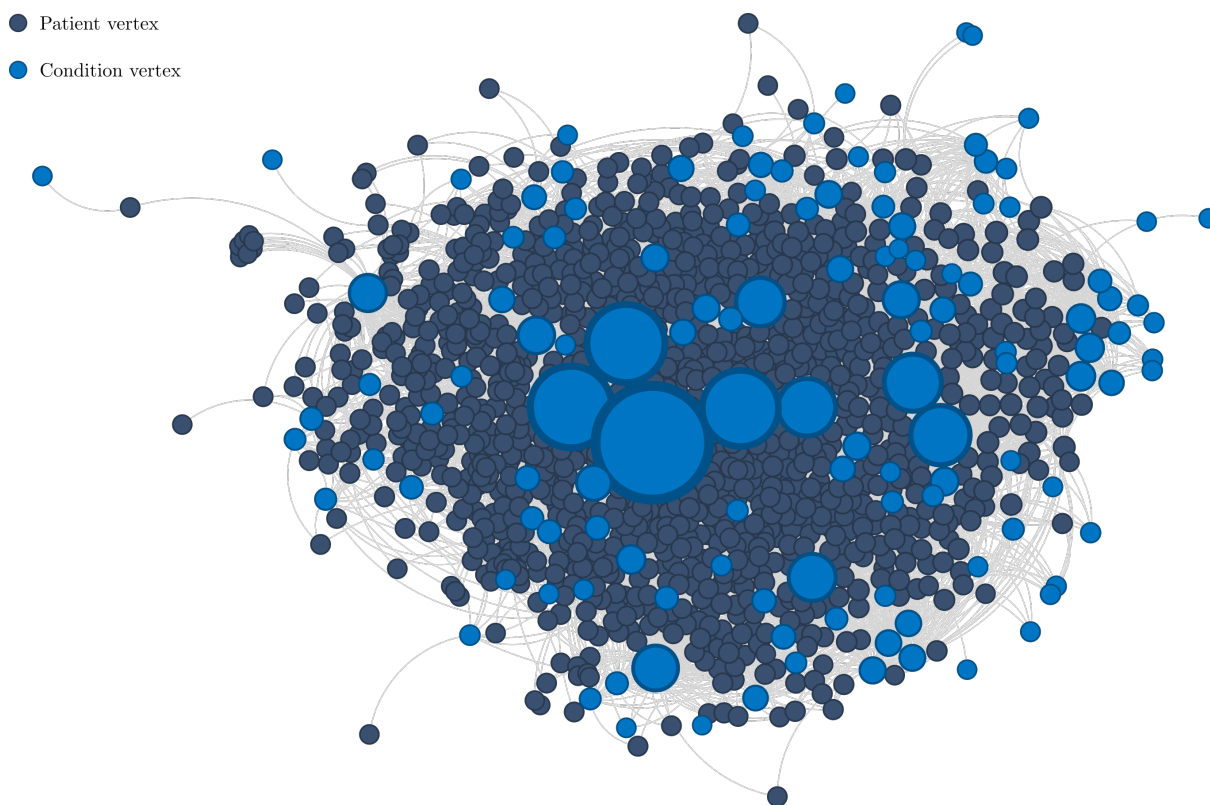


FIGURE 5.2: A graphical illustration of the 1K patient data set represented as a bipartite graph. The size of the vertices are visualised based on their degree, according to which size increases as the degree increases.

The graph constructed from the 1K data set comprises 1 152 patient vertices, 123 condition vertices, and 6 990 edges. The edges correspond to one or more conditions experienced by patients. Patient vertices represent 90.35% of the total vertices, while condition vertices constitute the remaining 9.65%. Patient vertices are only connected to condition vertices, therefore the average degree of patient vertices corresponds to the average number of conditions experienced by patients, equating to 6.07 in the case of the 1K data set. Similarly, the average degree of condition vertices indicates the average number of patients that have experienced this condition which equates to 56.83 in the case of the 1K data set. The 10K data set, on the other hand, contains 12 343 vertices, of which 12 165 vertices (98.55%) are patient vertices and 178 vertices (1.45%) are condition vertices. The average degree of the patient vertices is 9.31, while the average degree of the condition vertex is notably larger (*i.e.* 636.17) due to the large number of patient vertices in relation to the number of condition instances in the 10K data set. Furthermore, degree centrality (discussed in §2.1.2) is computed in order to gain further insight into conditions that are more common amongst the respective patient populations. The ten largest degree centrality scores (in respect of conditions) of the respective data sets are displayed in Figure 5.4. Recall that the 10K data set represents COVID-19 conditions, therefore some insight into prevailing symptoms of the infectious disease may be gleaned from this visualisation.

In Module 9.0, link prediction is performed in order to derive clinical insights from the Medical KG. First, different link prediction algorithms are implemented in Module 9.1, the selection of which can be guided by the literature discussed in §3.2–3.5. In this instantiation of the framework, a bipartite graph representation is adopted and the task of link prediction is formulated as a binary classification problem. For the sake of conducting a comprehensive algorithmic per-

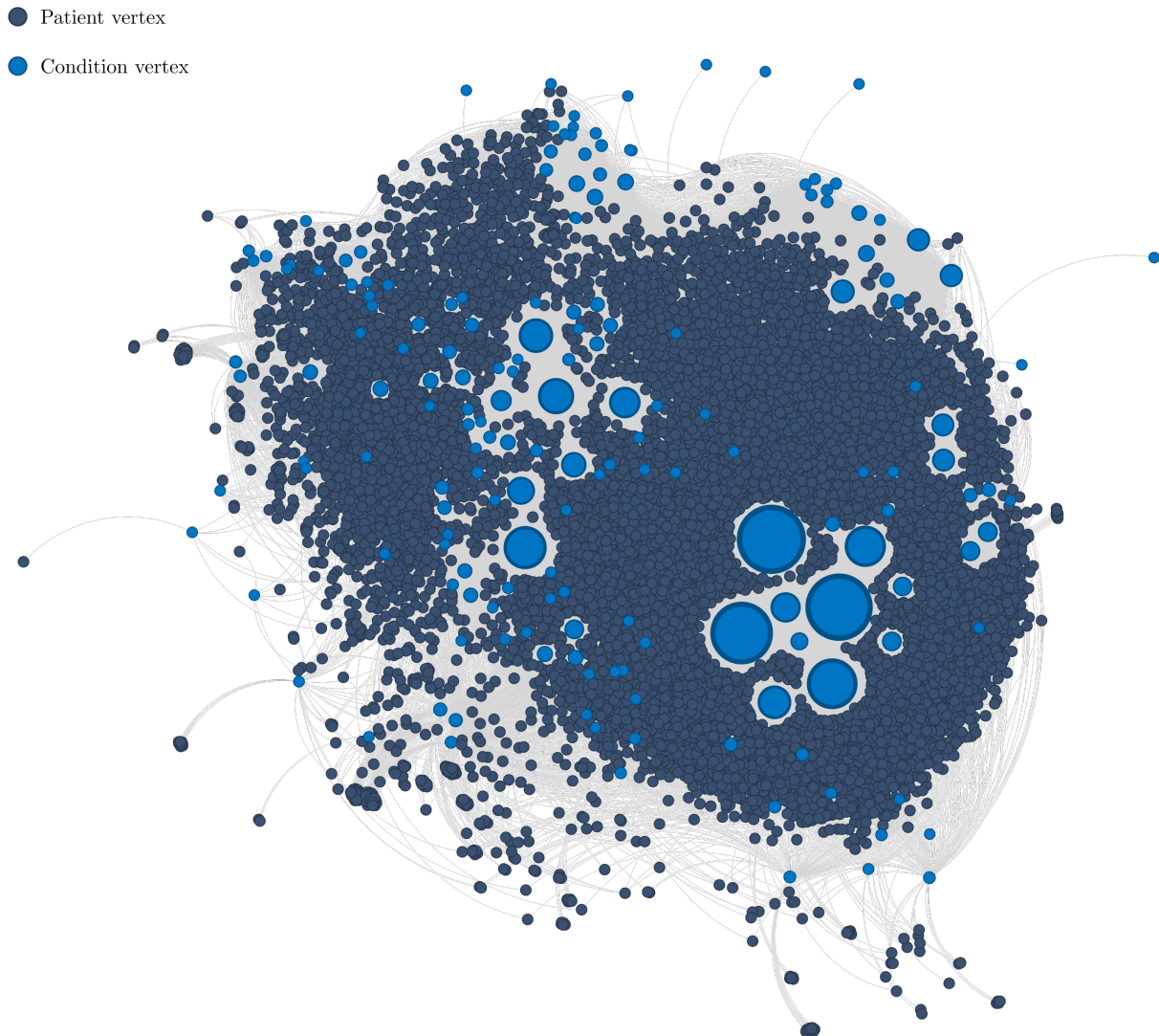


FIGURE 5.3: A graphical illustration of the 10K data set represented as a bipartite graph. The size of the vertices are visualised based on their degree, according to which size increases as the degree increases.

formance analysis, various link prediction approaches<sup>8</sup> were implemented, they are: CN-based algorithms, classifier-based algorithms, and GNN architectures. A summary of the different link prediction algorithms is presented in Table 5.9.

In the case of the CN-based algorithms, each data set was partitioned according to a uniform random distribution such that 80% of the data were employed for training and the remaining 20% were allocated to the probe set. In the case of the classifier-based algorithms, the data set was partitioned according to a uniform random distribution such that 80% of the data were employed for training, whereas the remaining 20% of the data were employed for the test set. The data split employed for the GNNs corresponded to 80% for training, 10% for validation, and for 10% testing, whereas negative edges were generated according to a uniform random distribution. Similarly, the GNN architectures undergo evaluation with respect to the test set. A total of thirty

<sup>8</sup>Recall that an algorithmic verification study was conducted (discussed in §5.1) in order to ensure correct functionality within the computerised environment.

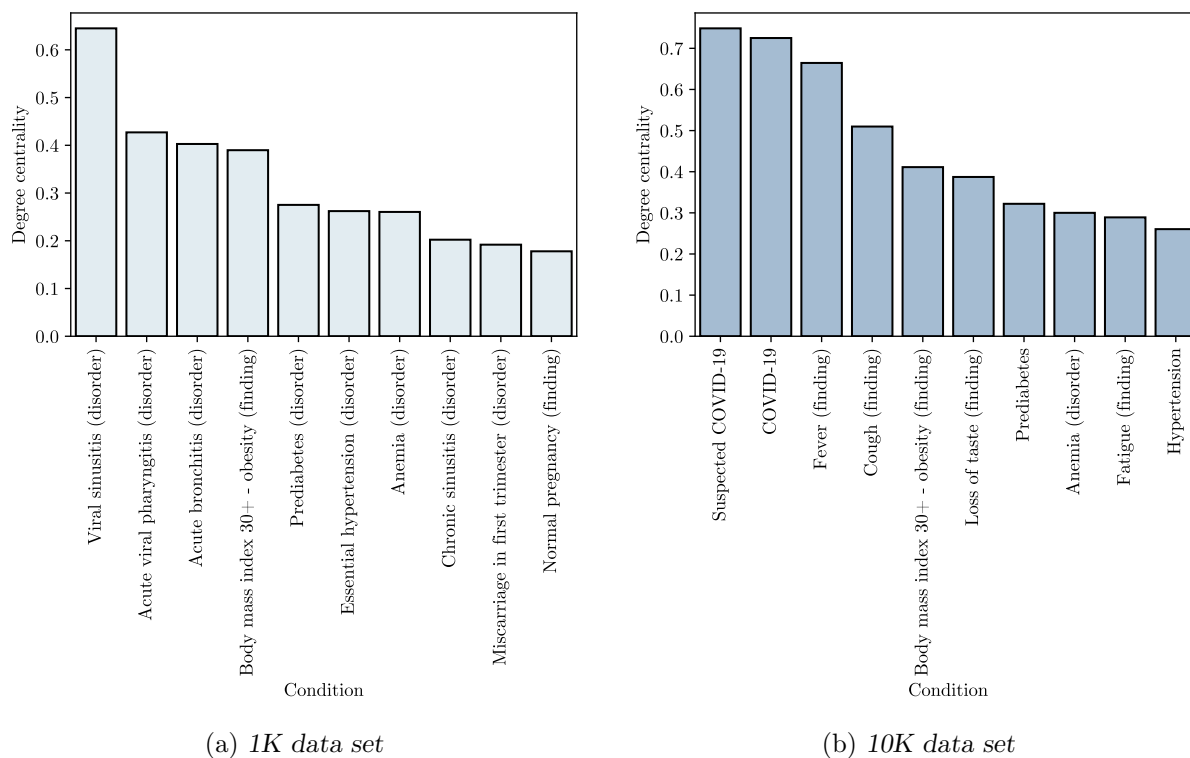


FIGURE 5.4: The top ten conditions based on degree centrality scores for the 1K and 10K data sets.

bootstrap<sup>9</sup> samples were generated from the original data set for the CN- and classifier-based algorithms in order to ensure statistical robustness with respect to performance evaluations. Similarly, in order to account for the randomness inherent in neighbourhood sampling, the GNNs were executed thirty times.

The iterative process of training and evaluation commences by means of Modules 9.2 and 9.3. Module 9.2 involves selecting and subsequently tuning model hyperparameter values in order to maximise the predictive performance of the different algorithmic techniques in respect of the training performance. Module 9.3 involves assessing algorithmic performance by means of the evaluation metrics AUROC and AUPRC. Emphasis is placed on AUPRC due to its pronounced suitability in respect of imbalanced classification problems for which the minority class corresponds to the positive class, as corroborated by Yang *et al.* [306] (discussed in §3.8). AUROC is nevertheless included ascribed to its widespread consideration in various link prediction studies which facilitates improved comparisons. The feature set employed for the classifier-based algorithms includes demographic, domain, and graph-based structural (topological) features. The demographic features included age, gender, race, marital status, as well as the county in which the patient resides while the domain features include healthcare coverage (expressed in monetary terms) and healthcare expense of each patient. Graph-based structural features, on the other hand, include those extracted by the CN-based similarity algorithms of (3.16)–(3.21). The feature set employed for the GNN algorithms included the text embeddings of each condition description as well as the demographic features corresponding to each patient.

The particular set of hyperparameters (together with their value ranges) selected for tuning was guided by findings in the literature [87, 220] and separate pilot studies carried out by the

<sup>9</sup>Bootstrapping is a resampling procedure that employs data from one population in order to generate a sampling distribution by means of extracting repeated random samples from the population, with replacement [221].



TABLE 5.9: A list of the algorithms considered in Module 9.1 for the first instantiation.

Category	Algorithm
CN-based	LCL
	RA
	CRA
	CAA
	CAR
	PRA
Classifier-based	LR
	NB
	$k$ NN
	DT
	RF
	MLP
GNNs	SAGE
	GAT
	GATv2
	GT

author. The CN-based approaches have no hyperparameters and therefore no additional tuning was performed. In the case of the LR model, the hyperparameters selected for tuning were the inverse regularisation<sup>10</sup> strength (denoted by  $C$ ), and the *penalty* term which may either be  $\ell_1$  or  $\ell_2$  regularisation. A notable form of regularisation involves employing parameter norm penalties, in which a regularisation term is added to the loss function in order to constrain the values of the weights [100]. The classifier-based algorithms were implemented using the Scikit-learn library [220] which has certain limitations in respect of the utilisation of the  $\ell_1$  penalty. More specifically,  $\ell_1$  regularisation can only be employed when an appropriate solver is explicitly specified. The default solver for Scikit-learn’s LR algorithm (*i.e.* **LBFSGS**) does not support the  $\ell_1$  penalty, but does permit either the  $\ell_2$  penalty, or no penalty (denoted as **None**). Therefore, only  $\ell_2$  regularisation was considered. The Gaussian NB variant was employed in this instantiation. The only hyperparameter that can be tuned for this variant is the term **alpha** which tends to improve stability during calculations by incorporating a measure of variance [220].

The considered hyperparameters of the  $k$ NN method included the *number of neighbours*  $k$  and the type of *weight function* employed during prediction. In the case of the DT, the *criterion* and *splitter* hyperparameters were selected. The criterion hyperparameter represents the function employed to measure the quality of the split, while the splitter hyperparameter refers to the strategy employed to select the split at each vertex, *i.e.* **best** which involves selecting the best split in respect of the largest decrease in impurity, or **random** which involves selecting the best random split in respect of decreasing impurity. Furthermore, the **ccp\_alpha** parameter was employed in order to mitigate overfitting. A larger value for this parameter results in more aggressive pruning which leads to simpler and shallower trees, therefore improving generalisation by reducing overfitting. The criterion hyperparameter, *number of trees* in the random forest, as well the **ccp\_alpha** parameter were selected as model hyperparameters for the RF classifier. Lastly, the hyperparameters selected for the MLP classifier were the *size of the hidden layers* (*i.e.* the number of hidden neurons), *learning rate*, and *maximum number of iterations* performed during training.

<sup>10</sup>Regularisation is implemented in order to improve generalisation performance.

In the case of the GNNs, the model hyperparameters selected for SAGE were the *number of epochs*, *aggregation function*, and *number of layers*. The candidate aggregator functions employed in SAGE were the mean operator (3.5.4) and the max operator (3.5.4). For the remaining GNN architectures, *i.e.* GAT, GATv2, and GT, the hyperparameters considered were the *number of layers*, *number of epochs*, and the *number of heads*, the latter of which denotes the number of multi-attention heads within the graph attention layers. A summary of the hyperparameters, together with their corresponding values, is presented in Table 5.10. It should be noted that default values (based on the respective libraries [87, 220]) were selected for the remaining hyperparameters.

TABLE 5.10: *The hyperparameters and their associated values employed during the tuning process.*

Algorithm	Hyperparameter	Values
LR	$C$	0.1, 1, 10
	Penalty	$l_2$
	Solver	LBFGS
NB	<b>alpha</b>	0.2, 0.4, 0.6
$k$ NN	Number of neighbours	4, 5, 6
	Weights	uniform, distant
DT	Criterion	gini, entropy, log_loss
	Splitter	best, random
	ccp_alpha	0.1, 0.2, 0.3
RF	Number of trees	50, 100, 150
	Criterion	gini, entropy, log_loss
	ccp_alpha	0.1, 0.2, 0.3
MLP	Size of hidden layers	(10), (15; 10), (20; 15)
	Learning rate	constant, adaptive
	Maximum number of iterations	200, 300
SAGE	Number of layers	2, 3, 4
	Number of epochs	40, 50, 60
	Aggregation	mean, max
GAT	Number of layers	2, 3, 4
	Number of epochs	40, 50, 60
	Number of multi-head-attentions	1, 2
GATv2	Number of layers	2, 3, 4
	Number of epochs	40, 50, 60
	Number of multi-head-attentions	1, 2
GT	Number of layers	2, 3, 4
	Number of epochs	40, 50, 60
	Number of multi-head-attentions	1, 2

Hyperparameter tuning for the classifier-based algorithms was conducted by means of a grid search with respect to both evaluation metrics, the results of which are shown in Tables A.1 and A.2 of Appendix A. The AUPRC metric was prioritised (as discussed earlier) and therefore the set of hyperparameters that yielded the largest AUPRC score for each algorithm was selected. The four GNN architectures were implemented using PyG’s heterogeneous convolution wrapper, *i.e.* **HeteroConv**, which facilitates the definition of custom heterogeneous message and update functions. Accordingly, the message functions are duplicated in order to operate on each edge type individually. Furthermore, PyG’s **RandomLinkSplit** method was employed for the data partitioning process, according to which 30% of the training edges were selected as supervision

edges. The hyperparameter tuning for the four GNN architectures was performed by means of random search in order to address the computational burden associated with the grid search method. Six randomly generated combinations (according to a uniform distribution) of hyperparameter values were selected for the tuning process. The results of this random search in respect of AUROC and AUPRC are presented in Tables A.3 and A.4, respectively. The hyperparameter value combination yielding the largest AUPRC score is considered best. Each link prediction algorithm was subsequently evaluated in respect of the testing partition, whilst employing the best hyperparameter combinations identified. The results, *i.e.* the mean AUROC and AUPRC scores with respect to the thirty runs on the test set, are detailed in Tables 5.11 and 5.12. The algorithm yielding the largest AUPRC score is considered best.

TABLE 5.11: *The mean AUROC and AUPRC scores achieved by the best performing hyperparameter combinations for each algorithm in respect of the 1K data set. The hyperparameter value combination yielding the largest AUPRC score is considered best and its corresponding AUROC and AUPRC scores are highlighted in bold.*

Algorithm	Hyperparameter combination	AUROC	AUPRC
LCL	—	0.8503	0.4661
RA	—	0.8468	0.4466
CRA	—	0.8591	0.4646
CAA	—	0.8562	0.4708
CAR	—	0.8498	0.4653
PRA	—	0.8828	0.5381
LR	$C = 1, \ell_2$	0.6204	0.3615
NB	$\alpha = 0.6$	0.6900	0.4154
$k$ NN	Number of neighbours = 6, distance	0.7676	0.4467
DT	entropy, best, ccp_alpha = 0.1	0.7905	0.4086
RF	Number of trees = 4, entropy, ccp_alpha = 0.1	0.8121	0.4744
MLP	Size of hidden layer = (20, 15), constant, max_iter = 300	0.6926	0.4084
SAGE	Layers = 3, Epochs = 40, Aggregation = max	0.8591	0.7675
GAT	Layers = 4, Epochs = 50, Heads = 2	0.8005	0.6811
GATv2	Layers = 4, Epochs = 50, Heads = 1	0.8433	0.7225
GT	Layers = 4, Epochs = 50, Heads = 1	<b>0.8583</b>	<b>0.7710</b>

In the case of the 1K data set, the overall best performing link prediction algorithm was the GT architecture which achieved an AUROC score<sup>11</sup> of 0.8591 and an AUPRC score of 0.7675. The CN-based algorithms achieved large AUROC scores but relatively small AUPRC scores which indicates that these algorithms failed to discern between the two classes effectively, *e.g.* the CAA algorithm attained an AUROC score of 0.8562 and an AUPRC score of 0.4708. The PRA method consistently outperformed its CN-based counterparts, achieving the largest scores in respect of both AUROC and AUPRC metrics. These results reaffirm the PRA’s efficacy as an effective CN-based bipartite link prediction algorithm, as reported by Aziz *et al.* [17] and observed during the algorithmic verification carried out in §5.1. With respect to the classifier-based algorithms, the RF algorithm performed the best, achieving an AUROC score of 0.8363 and an AUPRC score of 0.5337. Overall, these algorithms achieved the least favourable scores in terms of both evaluation metrics, notably each algorithm (excluding the RF algorithm) achieved an AUPRC score smaller than 0.5. The classifier-based algorithms therefore showcase inferior performance with respect to predicting missing links between patient and condition vertices. The achieved performance of the GNN architectures, however, showcase markedly superior predictive performance. Overall, the

<sup>11</sup>All references to “score” correspond to the mean score, unless stated otherwise.

TABLE 5.12: The mean AUROC and AUPRC scores achieved by the best performing hyperparameter combinations for each algorithm in respect of the 10K data set. The hyperparameter value combination yielding the largest AUPRC score is considered best and its corresponding AUROC and AUPRC scores are highlighted in bold.

Algorithm	Hyperparameter combination	AUROC	AUPRC
LCL	—	0.9153	0.6572
RA	—	0.9180	0.6588
CRA	—	0.9213	0.6630
CAA	—	0.9187	0.6618
CAR	—	0.9160	0.6606
PRA	—	0.9458	0.7288
LR	$C = 1, \ell_2$	0.5215	0.3005
NB	$\alpha = 0.4$	0.9162	0.7603
$k$ NN	Number of neighbours = 6, uniform	0.7295	0.4053
DT	entropy, random, ccp_alpha = 0.1	0.6494	0.3252
RF	Number of trees = 4, entropy, ccp_alpha = 0.1	0.8630	0.5336
MLP	Size of hidden layer = (10), constant, max_iter = 200	0.6696	0.4403
SAGE	Layers = 3, Epochs = 40, Aggregation = max	0.9416	0.9002
GAT	Layers = 4, Epochs = 50, Heads = 2	0.9529	0.9136
GATv2	Layers = 4, Epochs = 50, Heads = 2	<b>0.9618</b>	<b>0.9342</b>
GT	Layers = 2, Epochs = 40, Heads = 2	0.9219	0.8721

GNN architectures exhibit admirable performance (especially in respect of AUPRC) which may be attributed to their ability to leverage both structural properties as well as domain-specific information when predicting the presence of missing links between patient and condition vertices. Furthermore, the inclusion of text embeddings as vertex features, could also be a contributing factor towards its performance superiority.

In the case of the 10K data set, the best performing link prediction algorithm was the GATv2 architecture which achieved an AUROC of 0.9618 and an AUPRC of 0.9342. Similar to the case of the 1K data set, CN-based algorithms attained large AUROC scores but relatively small AUPRC scores which further substantiates the assertion that these algorithms fail to discern between the two classes effectively, regardless of the increase in data. The PRA method once again achieved the largest AUPRC score when compared with its CN-based counterparts. Surprisingly, the arguably rudimentary NB algorithm performed best with respect to the classifier-based algorithms achieving an AUROC score of 0.9162 and an AUPRC score of 0.7603. The RF algorithm, which performed the best with respect to the 1K data set, achieved an AUROC score of 0.8892 and an AUPRC score of 0.6088 with respect to the 10K data set, demonstrating reasonable performance in respect of the link prediction task, albeit notably inferior when compared with the NB algorithm. The remaining classifier-based algorithms, however, achieved relatively small scores for both evaluation metrics. The GNN architectures once again achieved superior levels of performance with respect to both evaluation metrics. In particular, the attention-based architectures, *i.e.* the GAT and GATv2 architectures, performed best overall achieving an AUPRC score of 0.9136 and 0.9342, respectively. The large AUPRC scores achieved by the GNN architectures further substantiate their utility for link prediction. Furthermore, the majority of link prediction algorithms showcased improved predictive capabilities in respect of the 10K data set which may be indicative of the potential benefits associated with additional data instances, although the increase in performance might also be attributable to a less complex predictive

task (given the different clinical context). Nevertheless, this finding underscores the scalability and versatility of GNNs when applied to more complex problem contexts.

In Figure 5.5, box plots are presented showcasing the performance achieved by the best performing algorithms with respect to the AUPRC metric. Once again, notable superiority is achieved by the GNN architectures in respect of AUPRC performance when compared with the best performing CN-based algorithm and classifier-based algorithm for both the 1K and 10K data sets. The performance difference in respect of the 10K data is especially evident. The larger data set exhibits reduced variability in respect of the thirty runs, as shown by the smaller interquartile range. This observation once again indicates that the increased data set size may possibly contribute to improved algorithmic robustness. The performance achieved by each of the sixteen link prediction algorithms is presented in Figures A.1–A.4, respectively.

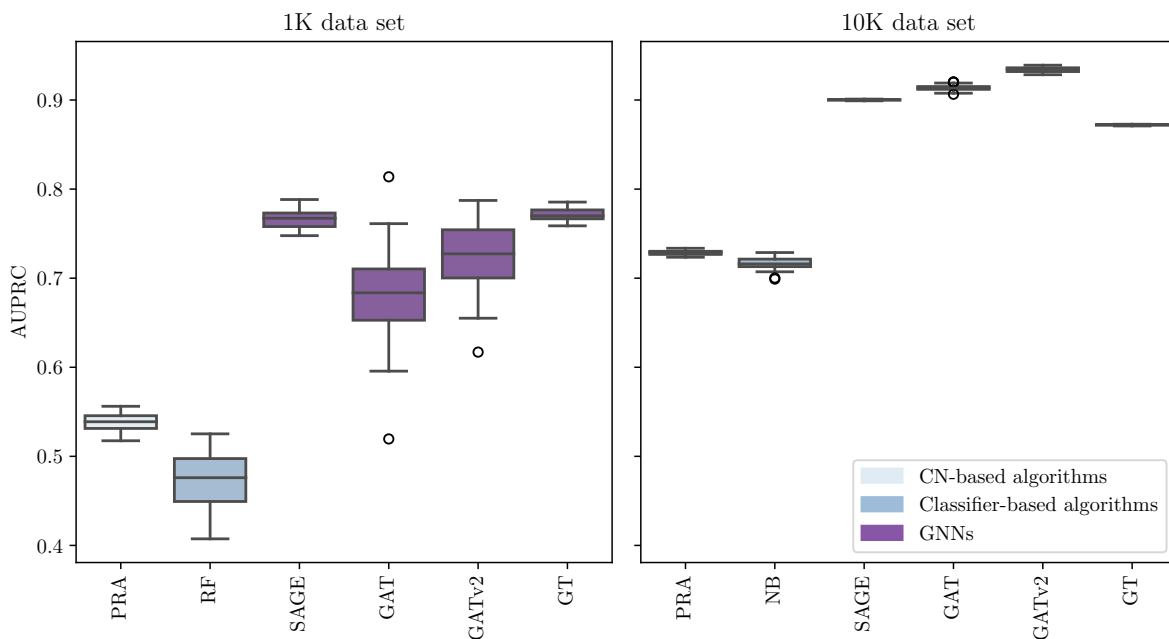


FIGURE 5.5: AUPRC box plots representing the top four algorithms (*i.e.* the GNN architectures) together with the best performing CN-based algorithm and classifier-based algorithm in respect of both the 1K and 10K data sets for the first instantiation.

In Module 10.0, statistical testing is conducted in order to analyse the algorithmic performance data in a more robust manner and subsequently draw the required inferences at a desired level of confidence. Non-parametric statistical tests were applied to the generated performance data — this decision is ascribed to the lack of clear consensus in the literature with respect to the assumptions that can be made with respect to the data’s distribution. More specifically, the non-parametric Friedman test is employed to determine whether a significant statistical difference exists between at least two of the algorithmic performance sample medians<sup>12</sup>. In the case of a difference being observed (at a desired level of confidence), the Nemenyi *post hoc* procedure is performed in order to identify the specific sample pairs in which the differences are present. The Friedman test is conducted in Python using the `scipy.stats` library [283], together with a significance level of  $\alpha = 0.05$ . All sixteen algorithms are subjected to statistical testing. The  $p$ -values obtained after applying the Friedman test to the respective samples are presented in Table 5.13.

<sup>12</sup>While the performance scores presented in this thesis are expressed as mean values, the statistical testing was carried out in respect of median values.

TABLE 5.13: The  $p$ -values obtained in respect of each evaluation metric and data set. A table entry less than 0.05 (indicated in red) denotes a difference at a 5% level of significance.

Data set	AUROC	AUPRC
1K	0	0
10K	0	0

Each of the four  $p$ -values obtained by the Friedman test is smaller than the significance level of  $\alpha = 0.05$  indicating that a statistically significant difference exists between at least two of the sample medians in each case. The Nemenyi *post hoc* procedure is subsequently applied to each instance, the results of which are presented in Tables A.5–A.8.

Towards identifying the best performing link prediction algorithm for each data set, the following procedure is employed. First, the different link prediction algorithms are ranked according to their respective sample medians. Algorithms that are statistically equivalent to the top ranked algorithm are then identified by means of the  $p$ -values obtained by means of the *post hoc* procedure. These algorithms are then considered the best performing link prediction algorithms with respect to the particular data set. In Tables 5.14 and 5.15, the best performing link prediction algorithms with respect to the AUPRC metric are presented (ranked in descending order according to their sample medians). Furthermore, the link prediction algorithms that were statistically equivalent to the best performing algorithm (*i.e.* ranked top) are listed in the last column. In the case of the 1K data set, the GT architecture performed best with respect to the AUPRC metric and its statistical equivalent algorithms were its GNN counterparts, *i.e.* SAGE, GATv2, and GAT. In the case of the 10K data set, the GATv2 architecture yielded the largest AUPRC score and its statistical equivalents were also its GNN counterparts, *i.e.* GAT, SAGE and GT. It may be inferred from these findings that the choice between these four GNN architectures may not significantly impact AUPRC performance. Further investigation is, however, warranted in order to evaluate other important factors that may possibly impact the selection of the most suitable GNN architecture, such as computational efficiency or ease of implementation.

TABLE 5.14: The best performing link prediction algorithms with respect to the AUPRC metric on the 1K data set ranked in descending order according to their respective sample medians.

Rank	Algorithm	Median	Statistically equivalent
1	GT	0.7700	SAGE, GATv2, GAT
2	SAGE	0.7672	
3	GATv2	0.7274	
4	GAT	0.6836	
5	PRA	0.5388	
6	$k$ NN	0.5160	
7	RF	0.4760	
8	LR	0.4727	
9	CAA	0.4668	
10	CAR	0.4622	
11	LCL	0.4615	
12	CRA	0.4606	
13	RA	0.4441	
14	NB	0.4419	
15	MLP	0.4166	
16	DT	0.4096	

TABLE 5.15: *The best performing link prediction algorithms with respect to the AUPRC metric on the 10K data set ranked in descending order according to their respective sample medians.*

Rank	Algorithm	Median	Statistically equivalent
1	GATv2	0.9346	GAT, SAGE, GT
2	GAT	0.9134	
3	SAGE	0.9002	
4	GT	0.8722	
5	PRA	0.7288	
6	NB	0.7160	
7	CRA	0.6631	
8	CAA	0.6618	
9	RA	0.6584	
10	CAR	0.6577	
11	LCL	0.6571	
12	$k$ NN	0.5560	
13	RF	0.5334	
14	MLP	0.4123	
15	LR	0.3570	
16	DT	0.2303	

The final module of the Analysis component, *i.e.* Module 11.0, comprises two child modules which facilitate the extraction of additional insight into algorithmic performance. In Module 11.1, visualisations<sup>13</sup> depicting the performance of the best link prediction algorithms are generated, depicted as precision-recall curves (Figure 5.6) and confusion matrices (Figure 5.12). In Figure 5.6(a), the precision-recall curve of the GT algorithm in respect of the 1K data set is presented, from which a favourable balance between precision and recall at various thresholds may be observed. The precision-recall curve of the GATv2 algorithm in respect of the 10K data set is presented in Figure 5.6(b), from which a considerable improvement may be observed in respect of achieving a favourable balance between precision and recall at various threshold.

The confusion matrices in Figure 5.7 were obtained by identifying a threshold corresponding to a maximum  $F$ -score. In Figure 5.7(a), it may be observed that the smallest value corresponds to instances for which the algorithm incorrectly classifies a positive class as a negative class, *i.e.* an FN. In the case of diagnosis prediction, minimising FNs may be considered a favourable outcome as “missing” a diagnosis can lead to a delay in necessary treatments and interventions, potentially resulting in exacerbated health issues or complications for the patient. The precision corresponding to Figure 5.7(a) is 0.6473, while the precision corresponding to Figure 5.7(b) is 0.8984. The recall corresponding to Figure 5.7(a) is 0.7825, while the recall corresponding to Figure 5.7(b) is 0.8515. A large precision indicates that a large proportion of patients identified by the algorithm as having the condition truly have it, while a large recall indicates that the algorithm correctly identifies a large proportion of all patients who truly have the condition. The algorithms therefore displays considerable improvement with respect to correctly predicting a patient’s diagnosis in the case of the 10K data set. Furthermore, the predicted results are visualised by means of bar plots indicating the top ten most frequently predicted conditions. A similar visualisation is generated for the historical conditions for the respective data sets thereby facilitating a direct comparison between the two sets of conditions. The bar plots corresponding to the 1K data set and 10K data set, are presented in Figures 5.8 and 5.9, respectively.

<sup>13</sup>These visualisations correspond to the experimental run (of which there is a total of thirty) that yielded the largest AUPRC score.

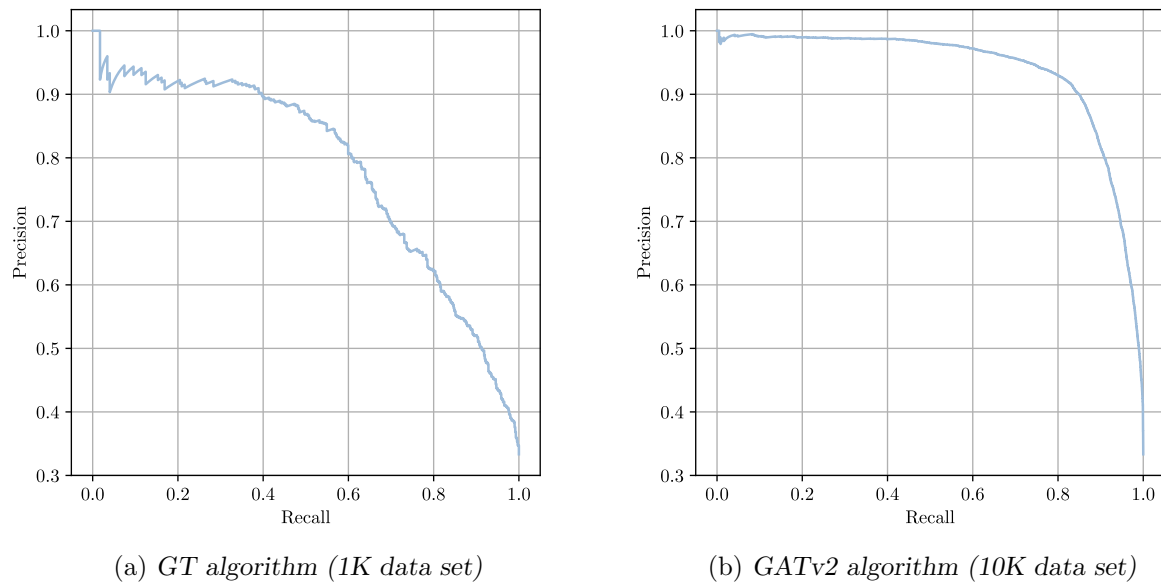


FIGURE 5.6: Precision-recall curves corresponding to the best performing link prediction algorithm in respect of each data set in the first instantiation.

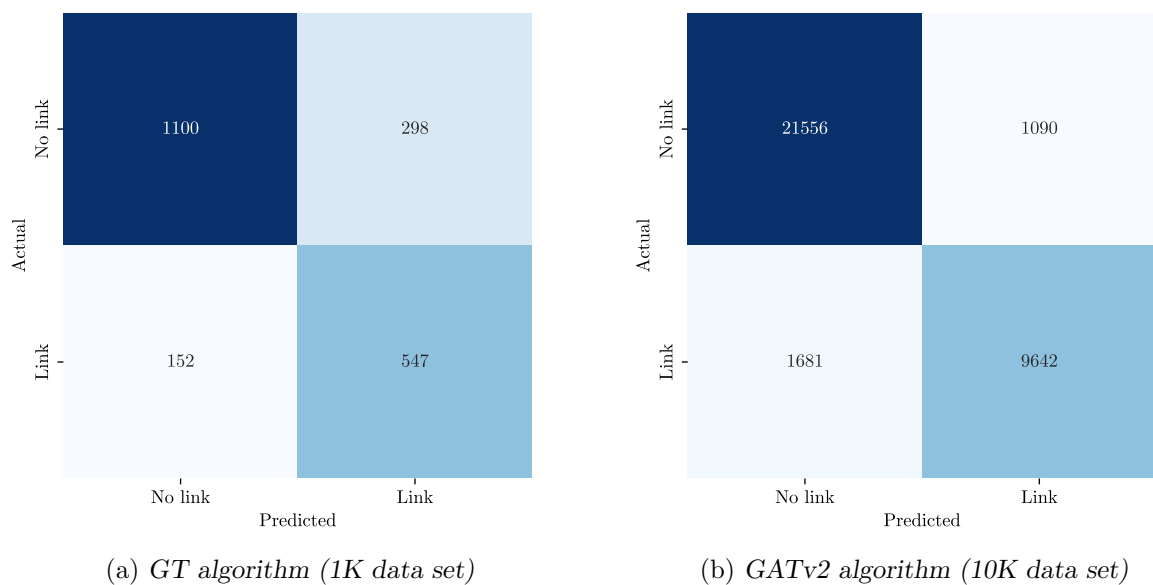


FIGURE 5.7: Confusion matrices corresponding to the best performing link prediction algorithm in respect of each data set in the first instantiation.

In the case of both data sets, the top ten historical conditions correspond to the conditions with the largest degree centrality scores (provided in Figure 5.4) which aligns with expectations as degree centrality typically relates to prominence and interconnectedness of vertices within a graph. In case of the the 1K dataset, both the top ten historical and top ten predicted conditions comprise similar entities, however, the ranking sequence differs notably in some instances. These



results may indicate that while the model accurately identifies the key conditions, their relative importances might not necessarily be captured to the same extent. This might be attributed to the limited sample size of the data set which can lead to insufficient representation of patterns or potential biases in the data set.

In the case of the 10K data set, there is a notable correspondence in respect of both the entities as well as their associated rankings in the historical and predicted plots — the only exception is the first and tenth position. The overall alignment in rankings, except in the case of the first and tenth positions, further underscores the model’s capability to identify not only the prominent conditions but also their relative hierarchical significance correctly. The appearance of “Sputum” instead of “Hypertension” may suggest that the algorithm is capable of inferring underlying patterns concerned with upper respiratory conditions given that the data set pertains to COVID-19 which is an upper respiratory disease. A more in-depth understanding of the changes in predictive rankings require, however, additional domain expertise in order to assimilate and infer the underlying significance and implications of these changes.

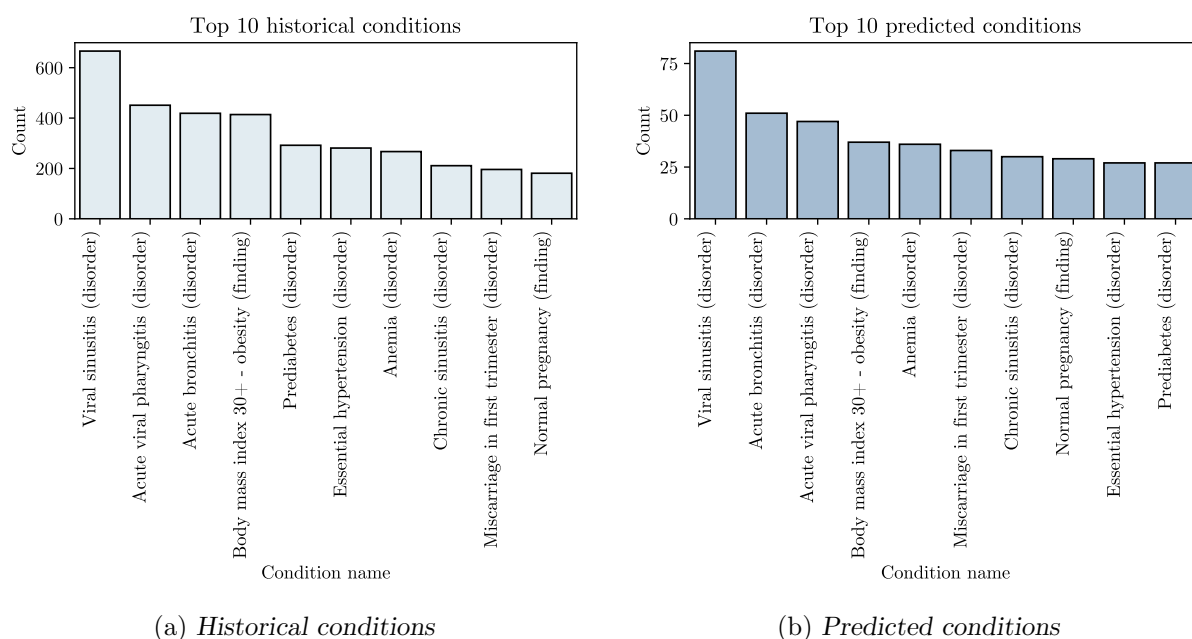


FIGURE 5.8: The top ten most frequent historical and predicted conditions for the 1K data set in respect of the first instantiation.

Lastly, in Module 11.2, the results are transformed into a format that enables healthcare practitioners to view both the historical and predicted conditions, therefore facilitating clinical decision support — an example of which is presented in Table 5.16. These resulting outputs, together with the bar plots in Figures 5.8 and 5.9, represent a more intuitive means towards developing a qualitative understanding, facilitating an insightful perspective into important patterns within the considered clinical context which forms the basis for deriving decision support in a data-driven and systematic manner.

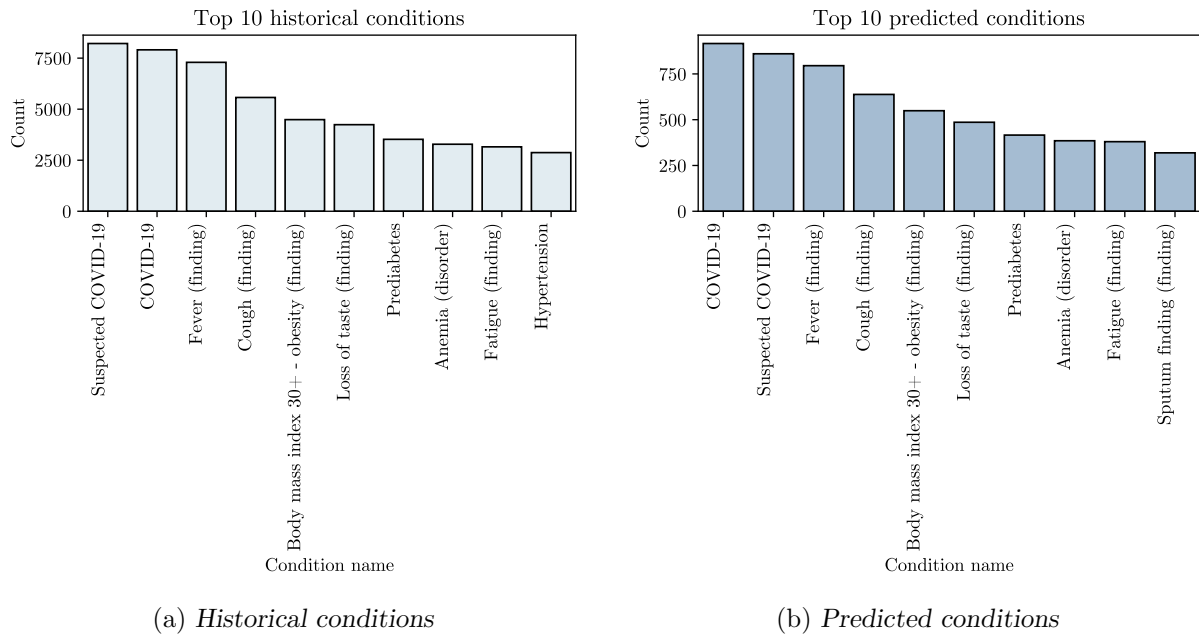


FIGURE 5.9: The top ten most frequent historical and predicted conditions for the 10K data set in the first instantiation.

TABLE 5.16: An extract of missing (undiagnosed) conditions in respect of the GATv2 architecture applied to the 10K data set in a format conducive to providing clinical decision support.

patientID	conditionCode	conditionName	status
0d791...	68235000	Nasal congestion (finding)	Historical
0d791...	59621000	Hypertension	Historical
0d791...	49727002	Cough (finding)	Historical
0d791...	248595008	Sputum finding (finding)	Historical
0d791...	84229001	Fatigue (finding)	Historical
0d791...	386661006	Fever (finding)	Historical
0d791...	840544004	Suspected COVID-19	Historical
0d791...	840539006	COVID-19	Predicted

## 5.4 Second framework instantiation

In this section, a second implementation instantiation of the MEDIKAL framework is considered. The data employed in this instantiation are derived from the COVID-19 10K data set [285]. The aim in this instantiation is to derive inferential patterns from a newly constructed KG towards predicting (or suggesting) relevant medication to prescribe for patients based on their diagnosed conditions. It should be noted that the following discussions are more limited in terms of exposition due to the similarity with respect to the first instantiation. The aim during this instantiation is to demonstrate the framework's utility in respect of prescribing medications to patients based on their current conditions.

### 5.4.1 Processing component

The graph data model in this implementation comprises a patient vertex, a condition vertex, a medication vertex, a HAS relationship, and a PRESCRIBED relationship, as shown in Figure 5.10. Modules 2.0 and 3.0 are executed in the same manner as in the first instantiation for both patient and condition vertices. The raw medication data are contained within the file *medications10K.csv*. An extract of the *medications10K.csv* file is presented in Table 5.17. Each medication is labelled according to a distinct RxNORM [208] code and description. Recall from §2.3.3 that RxNorm is a medical ontology containing standardised names for medications with respect to the active ingredient, strength, and dosage. The various medications prescribed to a patient may be determined by performing a search based on the particular patient’s ID. The raw data are then cleaned according to the child processes numbered 2.1 and 2.2. Module 2.3 is omitted due to the conformity of the *.csv* format in respect of downstream tasks. Thereafter, entity extraction is performed in Modules 3.1 and 3.2 with respect to the medications by identifying each distinct medication (represented by their distinct RxNORM code and description) within the *medications10K.csv* file, and stored as *medication\_nodes10K.csv*. The process of relation extraction is conducted in Module 4.0, whereby all distinct conditions corresponding to each patient ID are grouped, counted, and exported to a file named *has10K.csv* and all distinct medications prescribed to each patient ID are grouped, counted, and exported to a file named *prescribed10K.csv*. Modules 5.1 and 5.2 are subsequently employed for the purpose of structuring the data in a format considered appropriate for the execution of subsequent components.

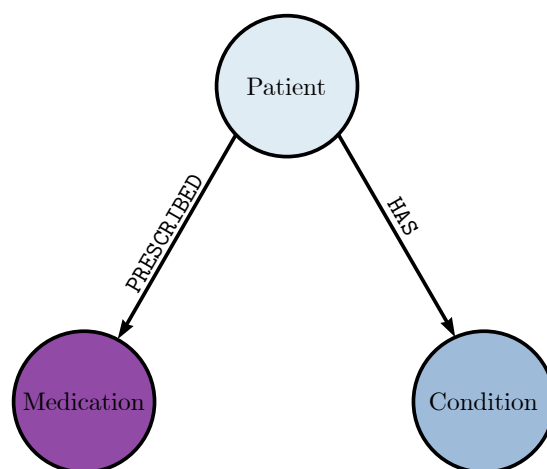


FIGURE 5.10: The graph data model employed in the second instantiation, indicating the triplets {patient, HAS, condition} and {patient, PRESCRIBED, medication}.

TABLE 5.17: An extract of the *medications.csv* file in which the medications experienced by various patients may be observed. The entries corresponding to the “Code” and “Description” column are derived from the RxNORM medical ontology.

Start	Stop	Patient	Code	Description
2019-10-30	2019-11-13	f0...	308182	Amoxicillin 250 MG Oral Capsule
2019-10-30	2019-11-13	f0...	313820	Acetaminophen 160 MG Chewable Tablet
2020-02-12	2020-02-26	06...	313820	Acetaminophen 160 MG Chewable Tablet
2020-04-28	2020-05-08	06...	834061	Penicillin V Potassium 250 MG Oral Tablet

### 5.4.2 KG construction component

The KG construction component is performed in the same manner as in the first instantiation for the patient vertices, condition vertices, and HAS relationship. The input file employed to construct the medication vertices, *i.e.* `medication_nodes10K.csv` file, comprises a `medID` column and a `Description` column, analogous to the `conditionCode` and `conditionName` columns, respectively, of the input file used to construct the condition vertices. Similarly, the distinct RxNORM codes in the `medID` are employed to generate a medication vertex mapping for the medication vertices, while a `Sentence-transformer` encoder is applied to the `Description` column in order to generate text embeddings for each medication’s description. These embeddings represent features for the medication vertices such that similar medication descriptions are represented by similar text embeddings. The `PRESCRIBED` relationship is generated by means of the same function employed to generate the `HAS` relationship. The `prescribed10K.csv` file is presented as input to this function which subsequently links the patient and medication vertices to one another by connecting the patient mappings (source) to the medication mappings (destination) according to the rows of `prescribed10K.csv`. Module 6.4 is omitted due to the absence of relation features, while Module 6.5 is employed towards verifying the graph structure in a similar manner to the initial instantiation. Ontological information is present within the input data, and therefore Modules 7.1–7.7 are not executed explicitly (hence their omission).

### 5.4.3 Analysis component

Module 8.1 is executed in the same manner as the first instantiation by constructing the graph using `NetworkX` and exporting it to `Gephi` as a `.gexf` file. The size of the vertices are visualised based on their degree, according to which the size increases as the degree becomes larger. The Medical KG may be observed in Figure 5.11. Graph feature analysis is then conducted so as to facilitate a quantitative understanding of the graph structure. A summary of the results obtained in Module 8.2 is presented in Table 5.18.

TABLE 5.18: A summary of the graph features computed in Module 8.2 in respect of the second instantiation.

Graph feature	
Number of patient vertices $ \mathcal{V}_p $	12 197
Number of condition vertices $ \mathcal{V}_c $	178
Number of medication vertices $ \mathcal{V}_m $	169
Total number of vertices $ \mathcal{V} $	12 544
Number of HAS edges $ \mathcal{E}_h $	113 239
Number of PRESCRIBED edges $ \mathcal{E}_{pr} $	33 707
Total number of edges $ \mathcal{E} $	146 946
Average number of conditions per patient	9.31
Average number of medications per patient	3.52
Average degree of condition vertices $\bar{d}(\mathcal{V}_c)$	636.17
Average degree of medication vertices $\bar{d}(\mathcal{V}_m)$	199.45

The Medical KG in this instantiation comprises 12 197 patient vertices, 178 condition vertices, and 169 medication vertices. Notably, patient vertices constitute 97.24% of the total vertices. For the sake of interpretability, the average degree of the patient vertex is calculated separately with respect to the condition vertex and the medication vertex. Once again, degree centrality is computed in order to gain further insight into which conditions and medications are more com-

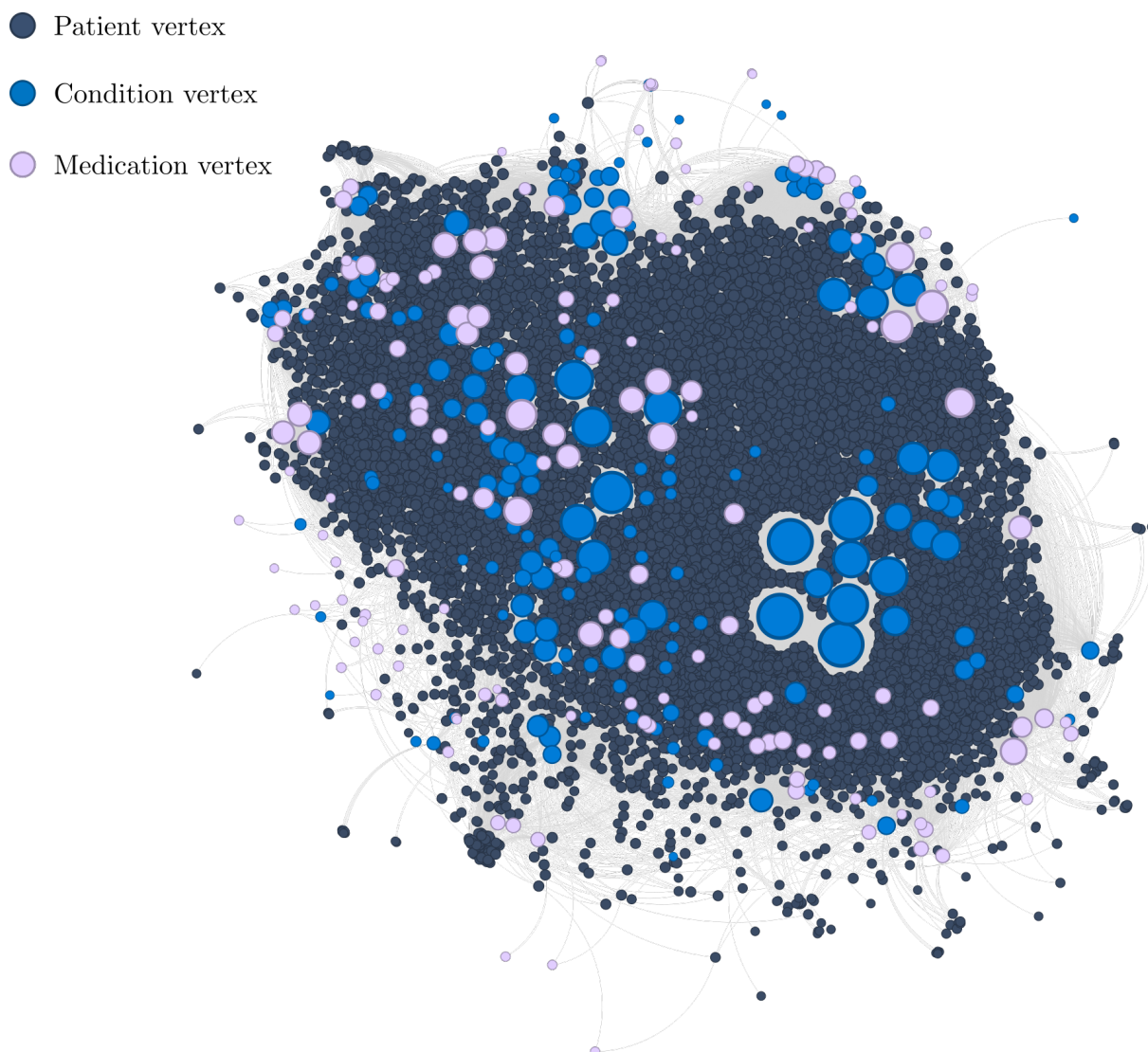


FIGURE 5.11: A graphical illustration of the 10K data set with the additional medication vertex. While certain medications are more frequently prescribed than others, they are not frequent as the condition vertices as indicated by their relatively smaller vertex size.

mon amongst the respective patient populations. Since both condition and medication vertices may only be connected to a patient vertex, the degree centrality for condition and medication vertices may be computed by employing the same approach adopted in the first instantiation. The ten largest degree centrality scores of the condition and medication vertices are presented in Figure 5.12, from which insight may be gleaned with respect to notable conditions and medications prescribed to patients for the purpose of treating COVID-19 and potential-related symptoms.

In Module 9.1, a set of link prediction algorithms is selected and applied. In this instantiation, link prediction is formulated as a binary classification task in which only `PRESCRIBED` edges are considered for prediction. The scope of algorithms considered for the second instantiation is limited to the four GNN architectures employed in the previous instantiation. The reason for limiting selection to GNNs is based on the increased complexity of the graph structure due to the addition of the medication vertex and `PRESCRIBED` relationship, rendering advanced techniques more applicable. Furthermore, the statistically superior performance of the four GNN architec-

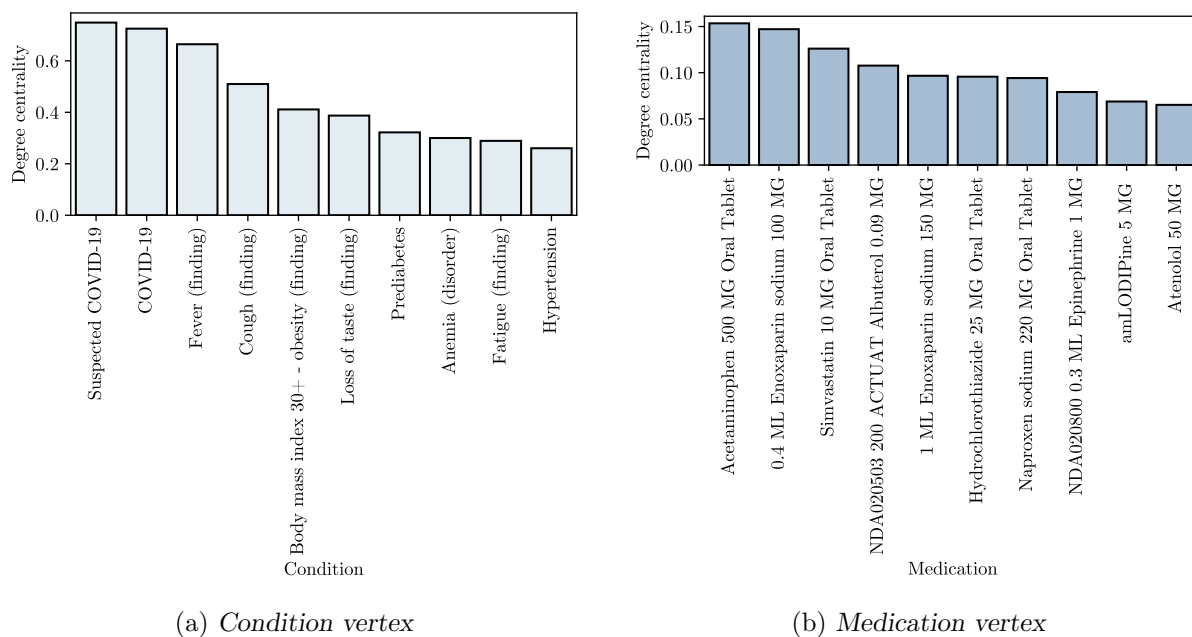


FIGURE 5.12: The top ten condition and medication degree centrality scores in respect of the second instantiation.

tures in respect of the first instantiation further warrants their exclusive consideration. The data split employed was 80% training, 10% validation, and 10% testing, whereas negative edges were generated according to a uniform random distribution. The hyperparameter values selected for each architecture corresponds to the best performing combinations identified during the first instantiation for the 10K data set, as listed in Table 5.12. The mean AUROC and AUPRC scores in respect of thirty runs on the test set are presented in Table 5.19. The best performing link prediction algorithm was the GATv2 architecture which achieved an AUROC score of 0.9505 and an AUPRC score of 0.9067. All four GNN architectures showcased favourable performance in respect of predicting new links between patient and medication vertices underscoring their utility towards clinical decision support for healthcare practitioners based on historical patient data. The AUPRC score achieved by each GNN architecture (in respect of thirty replications) may be observed in the box plots presented in Figure 5.13.

TABLE 5.19: The mean AUROC and AUPRC scores achieved by each GNN algorithm in respect of the second instantiation. The best performing hyperparameter value combinations identified during the first instantiation are employed.

Algorithm	Hyperparameter value combination	AUROC	AUPRC
SAGE	Layers = 3, Epochs = 40, Aggregation = max	0.9463	0.8955
GAT	Layers = 4, Epochs = 50, Heads = 2	0.9356	0.8876
GATv2	Layers = 4, Epochs = 50, Heads = 2	<b>0.9505</b>	<b>0.9067</b>
GT	Layers = 2, Epochs = 40, Heads = 2	0.9471	0.9026

The Friedman test is employed in Module 10.0, based on the same significance level of  $\alpha = 0.05$ . The  $p$ -values achieved for both evaluation metrics are less than the significance level therefore indicating that there is a statistically significant difference between at least two of the sample medians for both evaluation metrics. The Nemenyi *post hoc* procedure is therefore employed with respect to each algorithm, the results of which are presented in Tables 5.20 and 5.21.

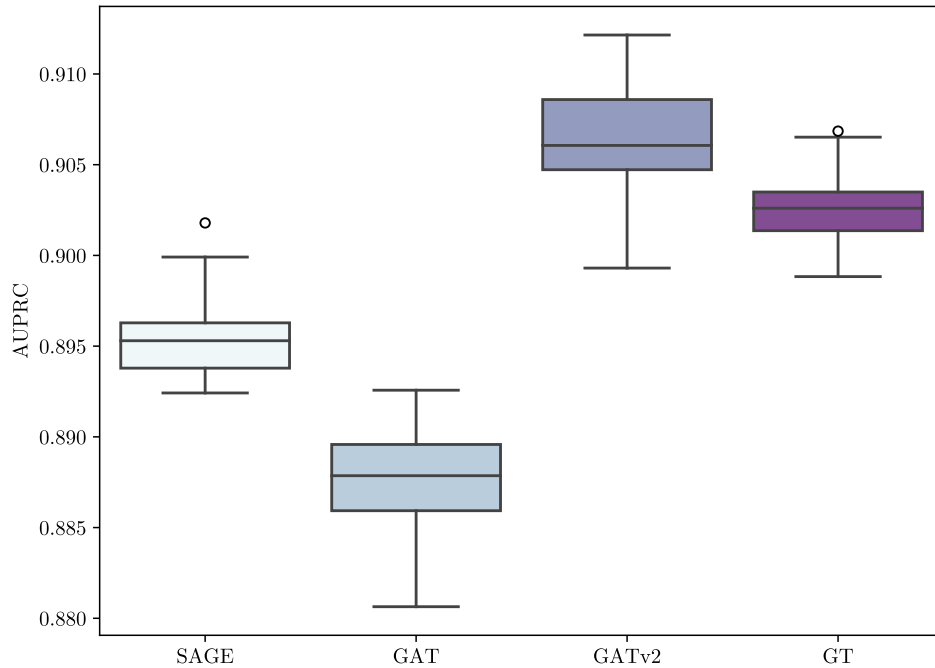


FIGURE 5.13: The AUPRC test performance achieved by the GNNs in respect of the second instantiation.

TABLE 5.20: The  $p$ -values derived from performing the Nemenyi test on the AUROC values obtained by the set of link prediction algorithms with respect to the 10K data set (second instantiation).

	SAGE	GAT	GATv2	GT
SAGE	—	0.0010	0.0010	0.7283
GAT		—	0.0010	0.0010
GATv2			—	0.0010
GT				—

TABLE 5.21: The  $p$ -values derived from performing the Nemenyi test on the AUPRC values obtained by the set of link prediction algorithms with respect to the 10K data set (second instantiation).

	SAGE	GAT	GATv2	GT
SAGE	—	0.0144	0.0010	0.0053
GAT		—	0.0010	0.0010
GATv2			—	0.0772
GT				—

The best performing algorithm with respect to the AUPRC metric is determined by ranking each GNN architecture according to their sample medians and subsequently identifying the GNNs that do not present a significant statistical difference with respect to the best ranked architecture, as presented in Table 5.22. In this instantiation, the GATv2 architecture is deemed the best performing algorithm, whilst its statistically equivalent counterpart is the GT architecture.

After the best performing GNN architecture has been identified, Modules 11.1 and 11.2 may be executed. In Module 11.1, visualisations depicting the performance of the GATv2 are presented

TABLE 5.22: *The best performing link prediction algorithms with respect to the AUPRC metric in the second instantiation.*

Rank	Algorithm	Median	Statistically equivalent
1	GATv2	0.9061	GT
2	GT	0.9026	
3	SAGE	0.8953	
4	GT	0.8879	

in the form of a precision-recall curve and confusion matrix, as presented in Figure 5.14. In Figure 5.14(a), the precision-recall curve of the GATv2 algorithm is presented which showcases a favourable trade-off between precision and recall at various thresholds. The confusion matrix in Figure 5.14(b) was obtained by identifying a threshold corresponding to the maximum  $F$ -score. In the case of prescription prediction, a large precision score indicates that a large proportion of the medications predicted by the model are correct, whereas a large recall score indicates that the model is effectively identifying the correct medication instances corresponding to each patient. The comparatively larger FNs may result in increased mistreatment as these patients are not prescribed the medication that they ought to receive. The precision and recall scores corresponding to the confusion matrix in Figure 5.14(b) are 0.8313 and 0.8281, respectively, which demonstrates reasonable predictive performance. The top ten most frequently predicted medications are compared with the top ten most frequent historic medications in order to gain more insight into the predictions performed by the GATv2 architecture. The visualisations are conducted by means of bar plots, as presented in Figure 5.15. The historical and predicted medications comprise similar entities with the exception of “Acetaminophen 325 MG” in the place of “NDA020800 0.3 ML”, as can be observed in Figure 5.15(b). Furthermore, the ranking sequence corresponding to the historical and predicted medications differ notably in respect of various instances which indicates that although the model can predict prescriptions sufficiently, there remains an opportunity for improvement. Future work may therefore involve conducting additional hyperparameter tuning in respect of the different GNN architectures.

Lastly, in Module 11.2, the results are transformed into a format that enables healthcare practitioners to view both the historical and predicted medications in order to facilitate clinical decision support — an example of which is presented in Table 5.23.

TABLE 5.23: *An extract of outcomes derived from the GATv2 architecture in respect of the 10K COVID-19 data set (of the second instantiation) which is presented in an intuitive format so as to aid clinical decision support.*

patientID	medID	Description	Status
00b5...	705129	Nitroglycerin 0.4 MG/ACTUAT Mucosal Spray	Historical
00b5...	849574	Naproxen sodium 220 MG Oral Tablet	Historical
00b5...	309362	Clopidogrel 75 MG Oral Tablet	Historical
00b5...	197361	Amlodipine 5 MG Oral Tablet	Historical
00b5...	2001499	Vitamin B 12 5 MG/ML Injectable Solution	Historical
00b5...	562251	Amoxicillin 250 MG / Clavulanate 125 MG Oral Tablet	Historical
00b5...	996740	Memantine hydrochloride 2 MG/ML Oral Solution	Historical
00b5...	312961	Simvastatin 20 MG Oral Tablet	Predicted



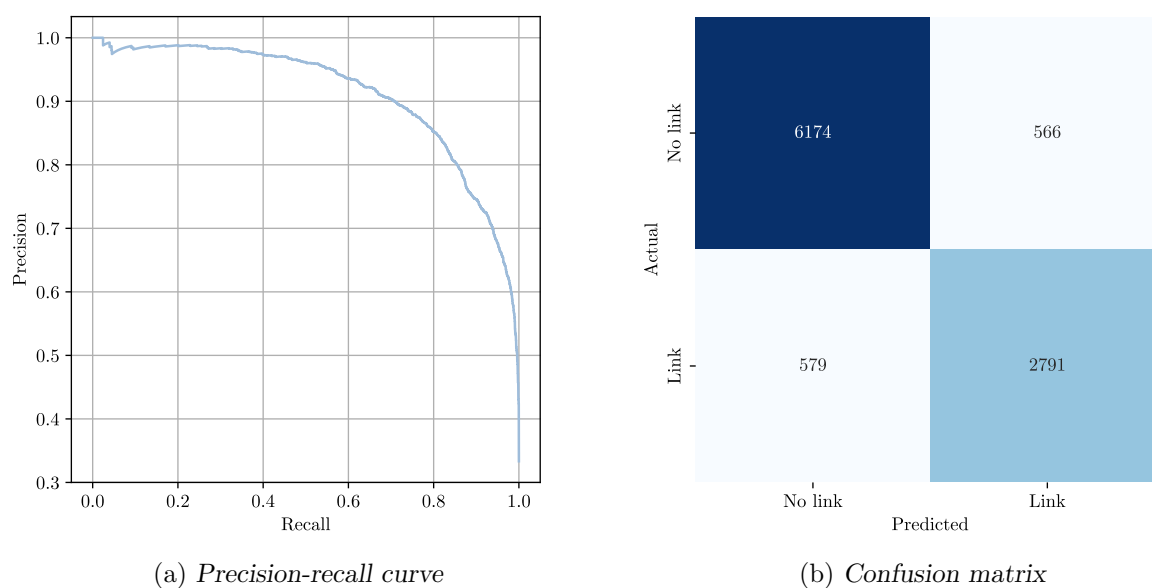


FIGURE 5.14: The precision-recall curve and confusion matrix achieved by the GATv2 algorithm with respect to the experimental run in which the largest AUPRC score was achieved.

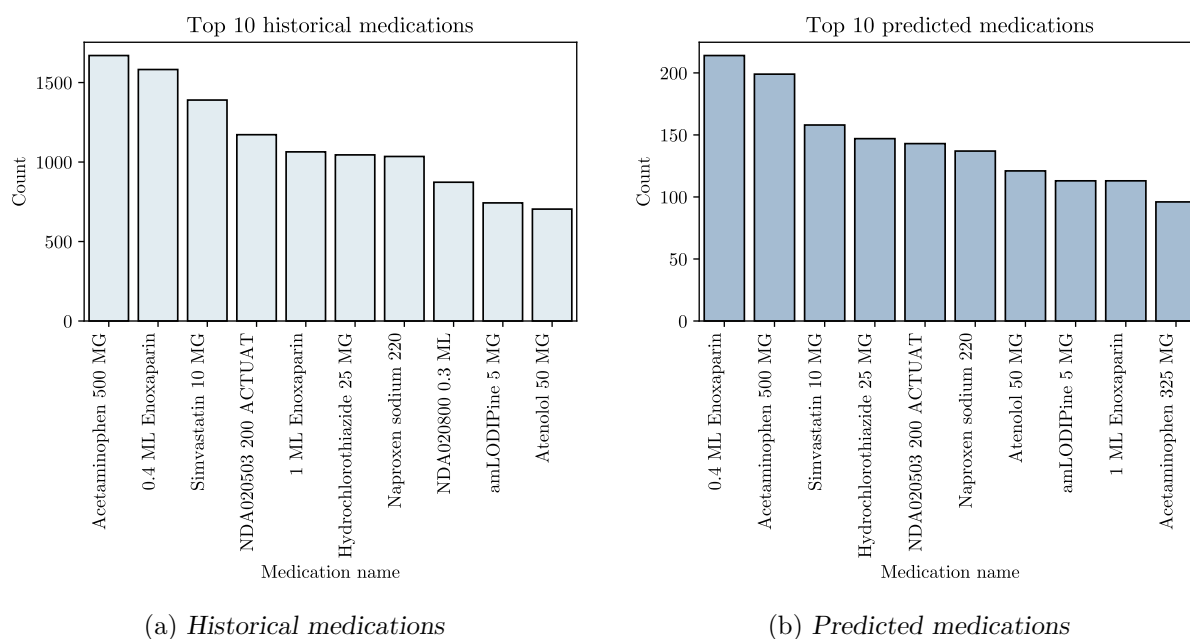


FIGURE 5.15: The top ten most frequent historical and predicted medications for the 10K data set in respect of the second instantiation.

## 5.5 Third framework instantiation

In this section, a third instantiation of the MEDIKAL framework is considered. The aim during this instantiation is to investigate the effect of supplementing the graph (constructed from the *COVID-19 10K data set* [285]) with additional information in respect of the hierarchical medical

ontology provided by SNOMED — this is accomplished by means of an additional *is-a* relation (discussed in §2.3.3). Towards this end, the SNOMED database was modelled as a KG in Neo4j [210] which was facilitated by a dedicated GitHub repository provided by SNOMED [261]. Neo4j is a graph database management system that facilitates the storage and management of data expressed as a graph data model, facilitating efficient representation and querying of complex graph structures. Towards accommodating the markedly extensive volume of data constituting the SNOMED KG, selective information was extracted from the SNOMED KG in order to facilitate the execution of this instantiation. The information retrieved from the SNOMED KG corresponds to all entities (in the SNOMED KG) that are connected to the vertices representing the conditions in the *condition\_nodes10K.csv* file *via* the *is-a* relation. The Medical KG is therefore supplemented and enriched by the medical entities embedded within the SNOMED KG which have been linked to the relevant objects within the Medical KG. The identified entities were exported as a .csv file, called *snomed\_nodes10K.csv*, containing two columns, *i.e.* *entityCode* and *entityName*. The *is-a* relation was exported as a .csv file, called *isa10K.csv*, which represents an edge list formatted in the same manner as *has10K.csv* and *prescribed10K.csv* (as discussed in §5.3 and 5.4). The aim during the subsequent discussions is to highlight certain modules within the MEDIKAL framework that are deemed important in respect of the additional clinical information. The framework’s utility can therefore be demonstrated in respect of prescribing medications to patients by considering both patient-specific conditions as well as related conditions, as supplemented by the SNOMED hierarchy.

### 5.5.1 Processing component

The graph data model corresponding to the third instantiation comprises a patient vertex, a condition vertex, a medication vertex, a HAS relationship, a PRESCRIBED relationship, and an ISA relation, as shown in Figure 5.16. The medical entities extracted from the SNOMED KG pertain to condition vertices and are therefore also represented as condition vertices in the graph. Towards this end, the ISA relation connects condition vertices (*i.e.* in the original data set and in the medical ontology) to one another thereby directly incorporating additional medical ontological information from the SNOMED KG. The remaining modules within the Processing component, *i.e.* Modules 2.0–5.0, are executed in the same manner as the prior instantiations and are therefore omitted from the discussion.

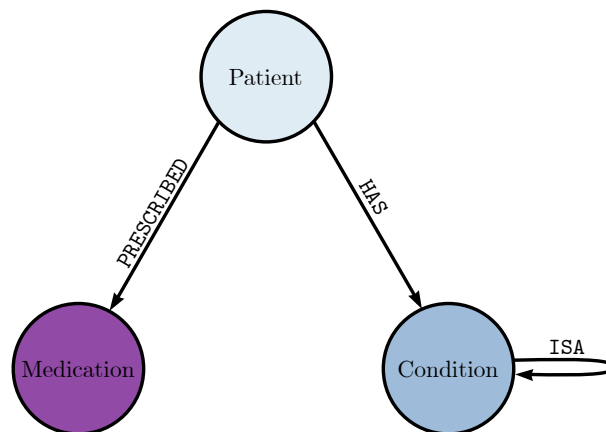


FIGURE 5.16: The graph data model employed in the third instantiation which denotes the triplets  $\{\text{patient}, \text{HAS}, \text{condition}\}$ ,  $\{\text{patient}, \text{PRESCRIBED}, \text{medication}\}$ , and  $\{\text{condition}, \text{ISA}, \text{condition}\}$ .

### 5.5.2 KG construction component

Module 6.0, *i.e.* Construct patient KG, is executed in the same manner as in the previous instantiation and is therefore omitted from this discussion. In order to construct the Medical ontology KG, the *snomed\_nodes10K.csv* and *isa10K.csv* files are employed. These files are already processed and therefore Module 7.1 is bypassed. Due to the fact that the input files of the medical ontology vertices comprise the same structure and data types as the input files of the condition and medication vertices, *i.e.* an `entityCode` column and an `entityName` column, the same functions employed to generate vertex mappings and text embeddings (which serve as vertex features) are employed, therefore completing Module 7.2 and 7.3. Furthermore, the medical ontology vertices are constructed as condition vertices in order to provide additional contextualised information to facilitate the process of predicting links between patients and medications. Similarly, the ISA relation, generated in Module 7.4 is created by means of the same function employed to generate the HAS and PRESCRIBED relations of the Patient KG. The ISA relation contains no features and therefore Module 7.5 is bypassed. Module 7.6 is indirectly executed through Module 7.4, *i.e.* the vertices of the Patient KG are connected to the vertices of the Medical ontology KG *via* the ISA relation, while Module 7.7 is performed in the same manner as for the Patient KG.

### 5.5.3 Analysis component

Module 8.1 is executed in the same manner as in the previous instantiations by constructing the graph by means of NetworkX and exporting it to Gephi as a `.gexf` file. The size of the vertices are once again visualised based on their degree, according to which the size increases as the degree increases. For visualisation purposes, the SNOMED vertices, located at the end vertices of the ISA relation, are partitioned separately in order to distinguish them from the original condition vertices — this facilitates a qualitative understanding of their degree distribution. The Medical KG may be observed in Figure 5.11. The results of Module 8.2 are presented in Table 5.24. The number of condition vertices increases to 727, while a total of 709 ISA relationships are present within the graph. Despite connections between condition vertices and other condition vertices, there are no self-loops within the Medical KG. The ten largest degree centrality scores of the end vertices of the ISA relation are presented in Figure 5.18.

TABLE 5.24: A summary of the graph features computed in Module 8.2 in respect of the third instantiation.

Graph feature	
Number of patient vertices $ \mathcal{V}_p $	12 197
Number of condition vertices $ \mathcal{V}_c $	727
Number of medication vertices $ \mathcal{V}_m $	169
Total number of vertices $ \mathcal{V} $	12 544
Number of HAS edges $ \mathcal{E}_h $	113 239
Number of PRESCRIBED edges $ \mathcal{E}_{pr} $	33 707
Number of ISA edges $ \mathcal{E}_i $	709
Total number of edges $ \mathcal{E} $	146 946
Average number of conditions per patient	9.31
Average number of medications per patient	3.52
Average degree of condition vertices $\bar{d}(\mathcal{V}_c)$	636.17
Average degree of medication vertices $\bar{d}(\mathcal{V}_m)$	199.45

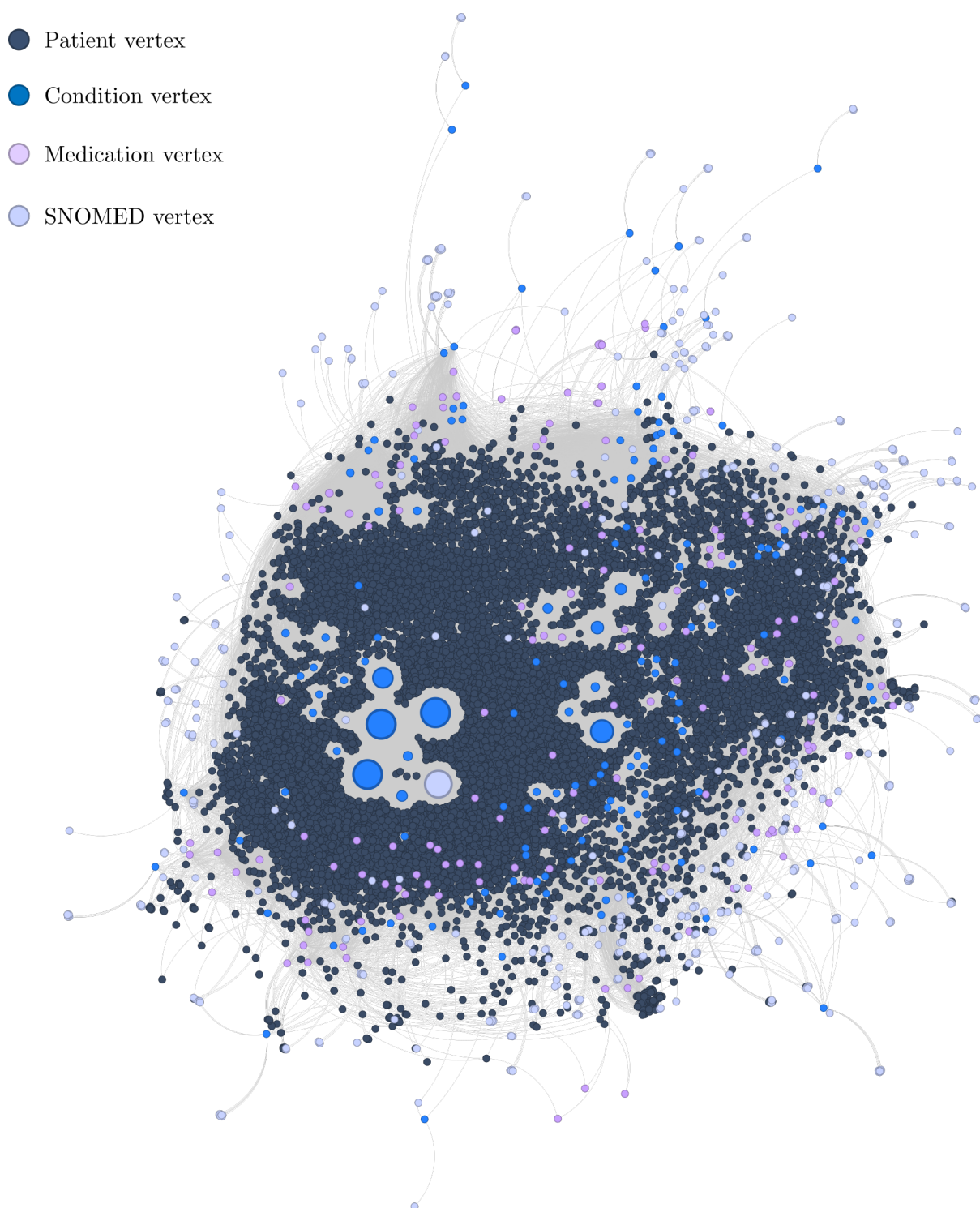


FIGURE 5.17: A graphical illustration of the KG constructed with the additional vertices and ISA relations. The SNOMED vertices are labelled separately in order to facilitate visualisation.

The specifications pertaining to the second instantiation in respect of algorithms, hyperparameters, and data partitioning (as discussed in §3.8.2) are adopted in this instantiation. The mean AUROC and AUPRC test scores with respect to the thirty runs are presented in Table 5.25, while the AUPRC box plots are presented in Figure 5.13. The GT architecture achieved the largest AUPRC score (0.9152). Supplementing the Medical KG with additional condition ver-

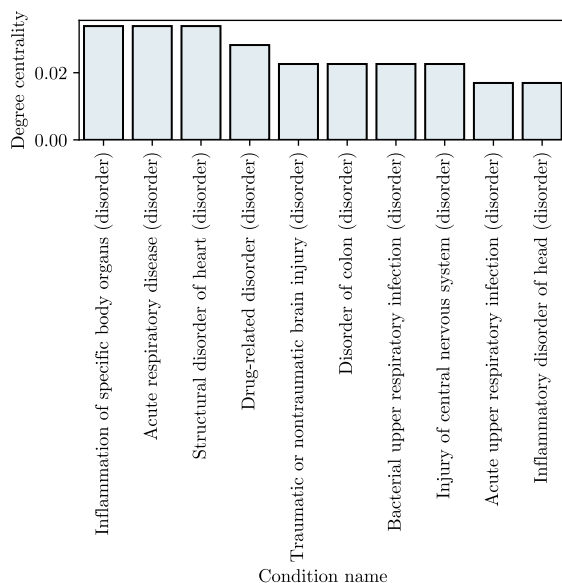


FIGURE 5.18: The top ten degree centrality scores of the end vertices of the ISA relation.

tices (as derived from the SNOMED hierarchy) resulted in a favourable increase in AUPRC performance across the majority of GNN architectures. Increases range from 0.8995 to 0.9070 in respect of the SAGE architecture, from 0.8876 to 0.9124 in respect of the GAT architecture and from 0.9026 to 0.9152 in respect of the GT architecture. The GATv2 architecture, which achieved the largest AUPRC score in the second instantiation (0.9067), did not, however, exhibit an increase in the AUPRC score. These findings indicate that supplementing the KG with additional condition vertices in respect of predicting new links between patient and medication vertices does not guarantee improved performance, however, further work is necessitated in respect of, for example, hyperparameters and their impact on algorithmic performance between different graph data models.

TABLE 5.25: The mean AUROC and AUPRC scores of the best performing hyperparameter combinations for each algorithm on the 10K data set in the third instantiation.

Algorithm	Hyperparameter combination	AUROC	AUPRC
SAGE	Layers = 3, Epochs = 40, Aggregation = <b>max</b>	0.9513	0.9070
GAT	Layers = 4, Epochs = 50, Heads = 2	0.9521	0.9124
GATv2	Layers = 4, Epochs = 50, Heads = 2	0.9477	0.8962
GT	Layers = 2, Epochs = 40, Heads = 2	<b>0.9513</b>	<b>0.9152</b>

Statistical inferential testing is conducted in Module 10.0 by means of the Friedman test. The significance level employed in this instantiation is  $\alpha = 0.05$ . The  $p$ -values achieved by the Friedman test in respect of both evaluation metrics are less than the significance level, indicating that there is a statistically significant difference between at least two of the sample medians for both evaluation metrics. The Nemenyi *post hoc* procedure is subsequently applied to each algorithm — the results of which are presented in Tables 5.26 and 5.27.

As in the previous instantiations, the best performing algorithm with respect to the AUPRC metric is determined by ranking each GNN architecture according to their sample medians and subsequently identifying the GNNs that do not exhibit a significant statistical difference with respect to the best ranked architecture, as presented in Table 5.28. In this instantiation,

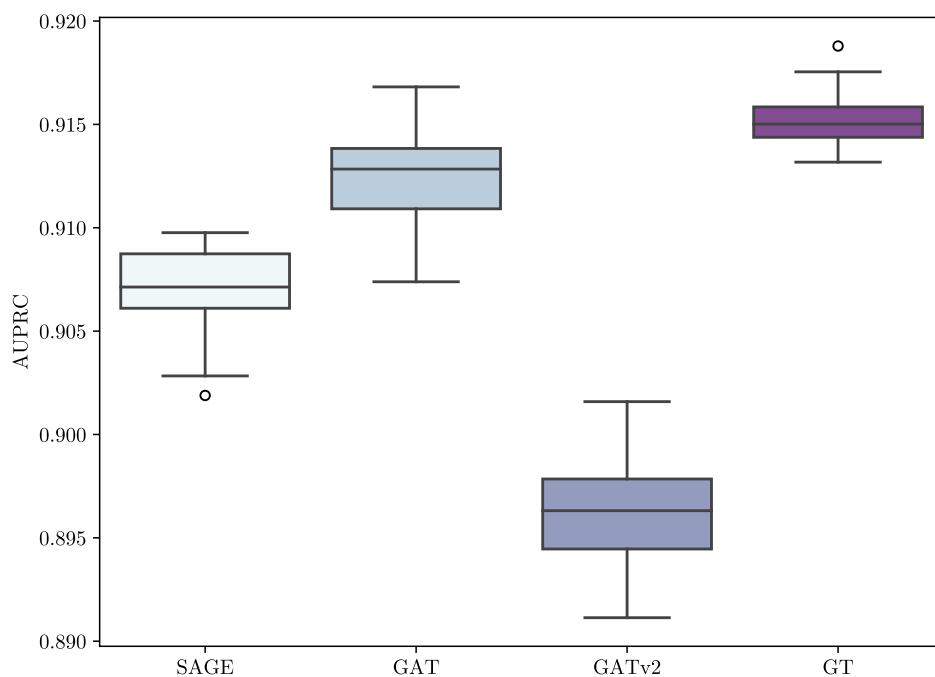


FIGURE 5.19: The AUPRC performance achieved by the GNNs in respect of the third instantiation.

TABLE 5.26: The  $p$ -values derived from performing the Nemenyi test on the AUROC values obtained by the set of link prediction algorithms on the 10K data set (third instantiation).

	SAGE	GAT	GATv2	GT
SAGE	—	0.4992	0.0010	0.9000
GAT		—	0.0010	0.5546
GATv2			—	0.0010
GT				—

TABLE 5.27: The  $p$ -values derived from performing the Nemenyi test on the AUPRC values obtained by the set of link prediction algorithms on the 10K data set (third instantiation).

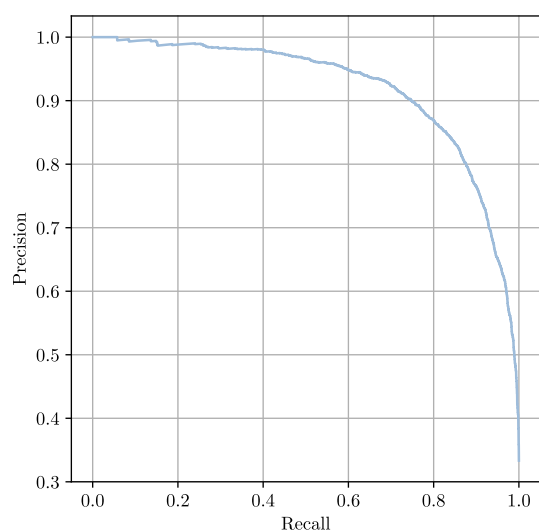
	SAGE	GAT	GATv2	GT
SAGE	—	0.0263	0.0075	0.0100
GAT		—	0.0010	0.0263
GATv2			—	0.0010
GT				—

the GT architecture is deemed the best performing algorithm with no statistically equivalent counterparts.

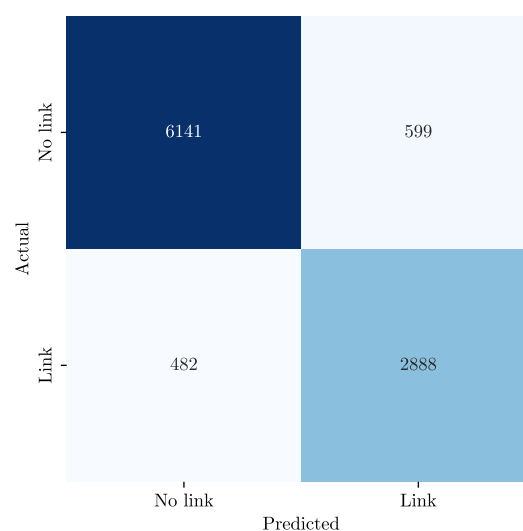
After the best performing GNN architecture has been identified, Modules 11.1 and 11.2 may be executed. In Module 11.1, visualisations (corresponding to the experimental run yielding the largest AUPRC score) of the GT architecture are presented in the form of a precision-recall curve and confusion matrix, *i.e.* Figure 5.20.

TABLE 5.28: The best performing link prediction algorithms with respect to the AUPRC metric in respect of the third instantiation.

Rank	Algorithm	Median	Statistically equivalent
1	GT	0.9150	—
2	GAT	0.9128	
3	SAGE	0.9071	
4	GATv2	0.8963	



(a) The precision-recall curve



(b) The confusion matrix

FIGURE 5.20: The precision-recall curve and confusion matrix of the GT algorithm corresponding to the experimental run which attained the highest AUPRC score.

In Figure 5.20(a), the precision-recall curve of the GT algorithm is presented. The confusion matrix in Figure 5.20(b) was obtained by identifying a threshold corresponding to a maximum  $F$ -score. In this instantiation, the smallest value in the confusion matrix corresponds to FNs which, in the case of prescription prediction, results in a smaller number of mistreatment as less patients are prescribed the incorrect medications. The precision and recall scores corresponding to the confusion matrix in Figure 5.20(b) are 0.8282 and 0.8570, respectively.

The top ten most frequent historic and predicted medications are presented in Figure 5.21. As in the second instantiation, both sets of medications comprise similar entities with the exception of “Vitamin B12” which appears in place of “NDA020800 0.3 ML”. The ranking sequence corresponding to the historical and predicted medications differ only in certain instances indicating slightly improved (ranking) performance when compared with the second instantiation. Lastly, in Module 11.2, the results are transformed into a format that enables healthcare practitioners to assimilate both the historical and predicted conditions in order to facilitate clinical decision support — an example of which is presented in Table 5.29.

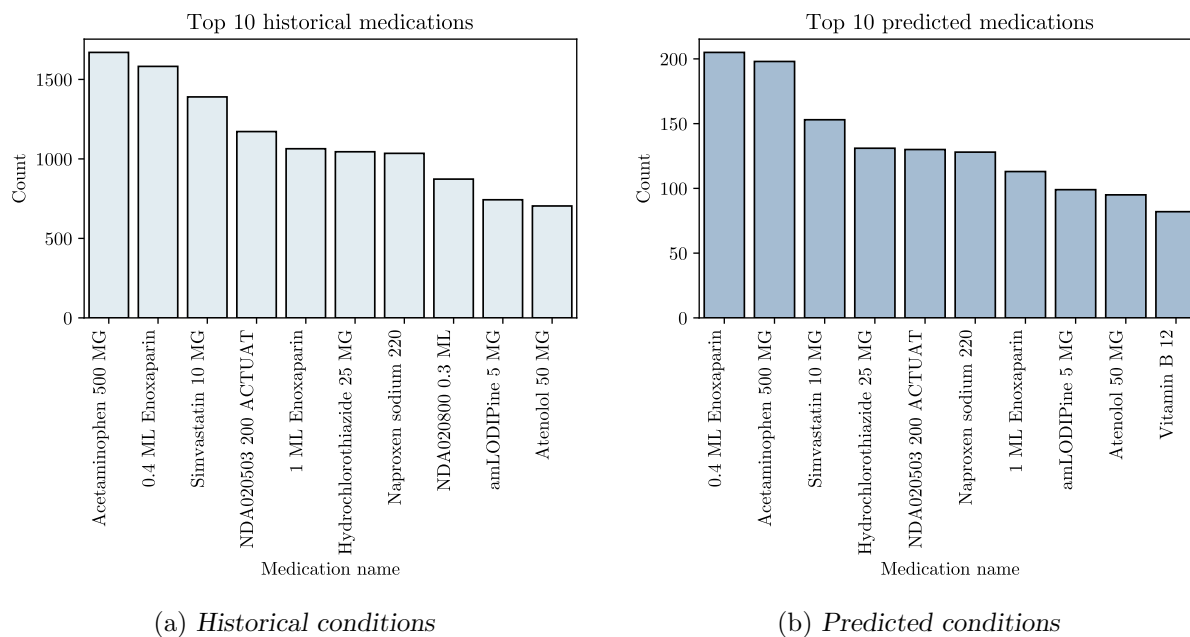


FIGURE 5.21: The top ten most frequent historical and predicted conditions for the 10K data set in the third instantiation.

TABLE 5.29: An extract of outcomes derived from the GT architecture on the 10K COVID-19 data set in an intuitive format so as to aid clinical decision support (third instantiation).

patientID	medID	Description	Status
00b5...	705129	Nitroglycerin 0.4 MG/ACTUAT Mucosal Spray	Historical
00b5...	849574	Naproxen sodium 220 MG Oral Tablet	Historical
00b5...	309362	Clopidogrel 75 MG Oral Tablet	Historical
00b5...	197361	Amlodipine 5 MG Oral Tablet	Historical
00b5...	2001499	Vitamin B 12 5 MG/ML Injectable Solution	Historical
00b5...	562251	Amoxicillin 250 MG / Clavulanate 125 MG Oral Tablet	Historical
00b5...	996740	Memantine hydrochloride 2 MG/ML Oral Solution	Historical
00b5...	312961	Simvastatin 20 MG Oral Tablet	Predicted

## 5.6 Reflection

The third instantiation of the MEDIKAL framework was conducted towards analysing the impact of supplementing the Medical KG with additional ontological information in respect of predicting new links between patient and medication vertices. The SAGE, GAT, and GT architectures achieved improved link prediction performance with respect to the more complex Medical KG, showcasing their ability to leverage the inferential relationships and patterns as a result of the additional ontological information. The GATv2 architecture, however, achieved a smaller AUPRC score. It may be argued that these results indicate that the GATv2 architecture was capable of extracting sufficient information from the simpler graph structure in order to perform predictions, however, it may also be conjectured that additional hyperparameter tuning might have resulted in a different outcome.

The confusion matrix corresponding to the best performing algorithm in respect of the third instantiation (Figure 5.20(b)) comprised 482 FNs, whereas the confusion matrix corresponding to



the best performing algorithm in respect of the second instantiation (Figure 5.14(b)) contained 579 FNs. These findings suggest that in the context of predicting links between patient and medication vertices, the third instantiation may present greater predictive capability by potentially reducing the instances of mistreatment by effectively identifying correct patient-medication correspondences. Consequently, these results could translate to more precise treatment plans. The integration of ontological data into the KG increases the model's inferential capabilities towards discerning relevant patterns and relationships, facilitating a greater understanding of patient-medication dynamics. A KG enriched with such comprehensive data therefore induces the computational opportunity for more sophisticated and precise link prediction outcomes, as it abstracts informative information embedded within the complex graph-based representations.

## 5.7 Methodological utility of MediKAL framework

In order to validate the framework in the context of an industry-based setting, the MEDIKAL framework was presented to Dr. Jacqueline Kazmaier, co-founder of a technology-based healthcare company, called Autoscriber [139]. The services of Autoscriber [15] involve the development of voice recognition software capable of recording, transcribing and extracting clinical information shared between healthcare practitioners and patients during consultations [158]. State-of-the-art speech recognition and NLP approaches are employed towards extracting structured clinical data from consultations between healthcare practitioners and patients thereby reducing administrative efforts by doctors and improving overall healthcare.

One of the main services provided by Autoscriber involves deriving actionable clinical insight from data so as to assist healthcare practitioners in respect of clinical decision making. It may therefore be reasonable to assert that the MEDIKAL framework could potentially provide practical utility towards Autoscriber's operations — the proposed framework facilitates the construction of a graph database comprising different medical-related entities pertaining to a patient population, as well as the subsequent derivation of clinical insights therefrom.

Kazmaier corroborated the MEDIKAL framework's suitability to both Autoscriber's and other related companies' operations [139]. Kazmaier also affirmed the versatility of the MEDIKAL framework, stating that it is sufficiently generic to be applied to various clinical contexts (including theirs). This underscores the framework's guidance towards processing both structured and unstructured clinical data effectively. Furthermore, the emphasis on graph-based abstractions is deemed especially valuable due to the interconnected nature of clinical information and the accompanying complexity of inferring actionable insight therefrom. The framework's utility in respect of both relevance and methodology was therefore confirmed.

As discussed in §4.6.3, the aim during the execution of Module 7.0 is to combine normalised and patient-specific medical information in order to facilitate computational efficacy during analyses, whilst ensuring the data's integrity and representational capabilities are not compromised. Based on her experience, Kazmaier asserts that the inclusion of this module is appropriate as conventional (non-graph) approaches towards representing patient-specific information are seldom sufficient for abstracting the complex interconnectedness of medical concepts. The incorporation of a Medical ontology KG results in further enrichment of contextual interconnectedness, as achieved by the additional level of abstraction. Enhanced utility is therefore demonstrated in respect of the Medical ontology KG module which facilitates the abstraction of inferential relationships embedded within clinical (*i.e.* doctor-patient consultation) data which can be leveraged by the MEDIKAL framework in a systematic and methodologically sound manner.

## 5.8 Chapter summary

In Chapter 5, the MEDIKAL framework's utility was demonstrated by means of various systematic instantiations within a computerised environment. The chapter opened with a discourse pertaining to algorithmic verification in §5.1. Thereafter, background pertaining to the clinical context was detailed in §5.2. Three computerised instantiations were conducted in this chapter. The aim during these instantiations was to analyse the impact of varying a KG's size and complexity in respect of link prediction performance, and to investigate two clinical use cases. In the first instantiation, the link prediction task was conducted with respect to diagnosis prediction on two data set sizes — the results of which were delineated in §5.3. The second and third instantiations, presented in §5.4 and §5.5, respectively, involved link prediction in respect of prescription prediction in order to analyse whether the addition of ontological information is able to enhance link prediction performance. The main findings that emanate from the three instantiations were synthesised and presented in §5.6, which was followed by a discourse pertaining to the validation of the proposed framework by a domain expert in §5.7.



# Part III

# Conclusion



---

---

## CHAPTER 6

---

# Conclusion

### Contents

6.1 Thesis summary . . . . .	141
6.2 Appraisal of thesis contributions . . . . .	142
6.3 Suggestions for future work . . . . .	144

The aim in this chapter is threefold: First, to present a summary of the work carried out in this thesis, secondly, to proffer an appraisal of the contributions of the thesis, and, finally, to recommend a number of extensions with respect to possible follow-up work that may emanate from the contributions made in this thesis.

### 6.1 Thesis summary

In addition to the introductory chapter, this thesis comprises an additional five chapters partitioned into three parts. Part I comprises two chapters, *i.e.* Chapters 2 and 3, which were dedicated to a thorough review of relevant prerequisites and the literature pertaining to the contextual and technical domains of knowledge covered in this thesis, in fulfilment of Objective I of §1.3. In Chapter 2, performed in fulfilment of Objective I(a), a review of the fundamental mathematical, statistical, and clinical prerequisites was presented so as to facilitate an understanding in respect of the relevant topics covered in the subsequent chapters. The chapter opened with an introduction to the basic concepts of graphs which was supplemented by important notational conventions along with various diagrammatic examples. A discussion pertaining to the relevant statistical preliminaries was subsequently presented, followed by a discourse pertaining to EHR data, synthetic data generation, medical ontologies, and clinical KGs.

Chapter 3 was dedicated to a comprehensive review of the field of link prediction, in fulfilment of Objective I(b). The chapter opened with an introduction to the fundamentals of link prediction which represented the foundation of the analytical work conducted in this thesis. Thereafter, an overview of various link prediction algorithms was presented, namely: CN-, classifier-, and embedding-based algorithms. A discussion pertaining to link prediction in respect of bipartite graphs and KGs was subsequently addressed in order to provide necessary technical information pertaining to the specific algorithmic approaches towards link prediction in respect of heterogeneous graph data. Important considerations in respect of performance evaluation and data partitioning were then delineated.

In Part II of this thesis, the design and implementation of the MEDIKAL framework were addressed. Detailed documentation was presented on an end-to-end architectural pipeline for constructing a KG from patient-related data and subsequently analysing the KG in order to derive clinical insights therefrom. More specifically, in Chapter 4, the design of this generic framework was proffered, in fulfilment of Objective II. First, a formal introduction to the notion of a framework as well as an overview of its developmental process was presented. Furthermore, background pertaining to DFDs and the generic data science paradigm was discussed, the latter of which represents the methodological foundation of the MEDIKAL framework. An account of similar frameworks within the literature was presented in order to contextualise the proposed framework. It was reported that related frameworks in the literature are centred towards either constructing a clinical KG from large amounts of biomedical data or analysing established KGs by means of different graph based approaches. Consequently, an opportunity was identified according to which both facets have not been not considered in a holistic manner, particularly in the context of deriving insights by means of link prediction. Appropriately, the proposed framework represents both a more nuanced and comprehensive approach towards graph-based inferential pattern recognition, as it is not restricted to predictions based on a limited set of labels (as in the case of other graph-based analyses, such as node classification) and can instead leverage inferences from various levels of abstraction. Towards this end, the need to develop a unified end-to-end framework was identified for constructing and analysing a medical KG, from which insights can be derived by means of link prediction. A high-level overview of the proposed framework was presented which contained a detailed description of its main functional components and their constituent modules. The descriptions were accompanied by DFDs which facilitated the induction of an intuitive understanding of the information flow within the proposed framework.

Chapter 5 was devoted to a demonstration of the MEDIKAL framework's utility by means of various systematic instantiations within a computerised environment. The chapter opened with a discourse pertaining to algorithmic verification in fulfilment of Objective III. Thereafter, background pertaining to the considered clinical context was detailed. Three computerised instantiations were conducted in this chapter in fulfilment of Objective IV. In the case of the first instantiation, the link prediction task was formulated and conducted with respect to diagnosis prediction in respect of two clinical data sets, each of which differs in respect of complexity and context. In the case of the second and third instantiations, link prediction was conducted in respect of medication prescription in order to further demonstrate the MEDIKAL's framework and to analyse whether the addition of ontological information could enhance algorithmic performance and, if so, to what extent. The main findings that emanated from the three instantiations were subsequently synthesised in fulfilment of Objective V. Lastly, a discourse pertaining to the validation of the proposed framework by a domain expert was presented in fulfilment of Objective VI.

## 6.2 Appraisal of thesis contributions

The main contributions of this thesis are six-fold. In this section, a documentation and appraisal of these contributions are presented.

**Contribution I** *The design, documentation, and development of a generic end-to-end framework for constructing and analysing clinically derived KGs.*

The MEDIKAL framework presented in Chapter 4 represents an attempt towards addressing the shortcomings of current frameworks within the literature by proposing an

end-to-end pipeline that facilitates both the construction and analysis of a KG for the purposes of clinical decision support. The MEDIKAL framework comprises three primary functional components, *i.e.* a Processing component, a KG construction component, and an Analysis component which are integrated into a unified architectural pipeline capable of deriving insights from raw clinical data by means of link prediction. The proposed framework was published in a reputable peer-reviewed journal [219]. A detailed design of the framework was presented in Chapter 4 which was facilitated by means of DFDs depicting the working at various levels of abstraction.

**Contribution II** *A demonstration of the framework’s utility by means of three computerised instantiations.*

The documentation of the aforementioned MEDIKAL framework extended beyond a conceptual level — three practical implementations of the framework were conducted by means of computerised instantiations. Prior to these implementations, however, algorithmic verification was carried out in order to demonstrate the correctness of the computational implementations of the algorithms considered. Detailed considerations were documented during the different instantiations so as to elucidate the practical realisation of the framework — some of the main considerations included descriptions of various data structures, algorithms (including their configurations), and software libraries employed in order to implement the modules constituting the framework. Furthermore, in order to demonstrate the framework’s utility, the three instantiations differed in respect of the data considered and/or the clinical use case.

In the case of the first instantiation, two data sets were considered, differing in respect of size and clinical context, facilitating and enhancing the analyses carried out, as insight can be gleaned in respect of algorithmic performance under different circumstances. In the case of the second instantiation, the considered graph data model comprised an additional vertex and edge type in order to showcase the impact of supplementing the KG with additional clinical information with respect to algorithmic performance (more specifically, GNNs). Lastly, the third instantiation was carried out in a similar manner as the second instantiation, however, it was supplemented with additional vertices derived from the SNOMED ontology.

**Contribution III** *An algorithmic performance analysis of link prediction techniques in respect of KGs at differing levels of abstraction.*

The instantiations conducted in Chapter 5 involved subjecting the link prediction algorithms to KGs of varying complexity. These comparisons involved different configurations of graphs, *e.g.* variations in respect of the number of vertices and edges, as well as the types of vertices and edges. A valuable contribution is therefore made in respect of furthering the understanding of link prediction performance in respect of clinically derived KGs at various levels of abstraction.

**Contribution IV** *A comprehensive comparison of link prediction approaches.*

The first instantiation involved a broad range of link prediction algorithms which included CN-based approaches, classifier-based approaches, and deep-learning methods, *i.e.* GNNs. Furthermore, comprehensive hyperparameter evaluation was carried in respect of all sixteen algorithms, from which further insight into algorithmic performance was gleaned. Based on the numerical results generated, it was reported that contemporary GNNs consistently outperformed their algorithmic counterparts with respect to both the 1K and



10K data sets. The algorithmic prowess of GNNs were demonstrated in both a quantitative and qualitative manner. Furthermore, the superior performance achieved by the GNN architectures highlights the utility of graph-based deep-learning methods towards extracting complex inferential patterns and relationships from graph data. The four GNN architectures were then employed in the second and third instantiations in order to accommodate the increased complexity of the respective KGs within a different clinical context (*i.e.* medication prescription). Statistical analyses were carried out, from which it was observed that algorithmic performance achieved was reasonably consistent in respect of the different GNN architectures. Important algorithmic insight was therefore proffered by means of the analyses carried out in this thesis — a basis for future research endeavours in respect of consolidating efforts pertaining to contemporary GNN approaches.

**Contribution V** *A verification study of CN-based link prediction techniques for bipartite graphs.*

In Chapter 5, an algorithmic verification study was conducted in respect of the comparatively nascent CN-based link prediction approaches for bipartite graphs by replicating (to a reasonable extent) the results achieved by the original authors Aziz *et al.* [17]. The algorithmic performance of the CN-based link prediction approaches were evaluated in respect of four well-known link prediction benchmark data sets — *i.e.* publicly available drug-target interaction networks. The results attained during the verification study were similar to those achieved by Aziz *et al.* which represented a reasonable verification of the author’s computerised instantiation, therefore imbuing the subsequent analyses with credence. The corresponding findings also aid the corroboration of assertions pertaining to CN-based link prediction approaches — enhancing the current body of knowledge.

**Contribution VI** *Face validation by a domain expert.*

The MEDIKAL framework was validated by means of an appropriate subject matter expert, the aim of which was to corroborate the framework’s practical utility. The framework was presented to a co-founder of the healthcare company Autoscriber, Dr. Jacqueline Kazmaier, who corroborated the framework’s suitability to both Autoscriber’s and other related companies’ operations. Kazmaier stated that the MEDIKAL framework’s generic and comprehensive nature renders it applicable to various clinical data (including data considered by the company). Furthermore, Kazmaier highlighted the algorithmic significance of the proposed framework in respect of the benefits associated with graph-based approaches — she stressed that clinical data are inherently interconnected and conventional analyses are seldom sufficient. The MediKAL framework was therefore deemed contextually suitable and sufficiently advanced.

### 6.3 Suggestions for future work

This final section comprises suggestions for five avenues of further investigation as possible follow-up work on the contributions of this thesis. In each case, the suggestion is stated formally and then elaborated upon briefly.

**Suggestion I** *Development of a user interface for the MEDIKAL framework.*

In order to further leverage the benefits of the MEDIKAL framework in a practical setting, the implementation of the proposed pipeline ought to be streamlined by means of a GUI so as to facilitate effective navigation and interaction with the various constituent modules.

The addition of a GUI enhances the scope of engagement by enabling users without a technical background to leverage its benefits which, consequently, improves its practicality with respect to real-world clinical adoption.

**Suggestion II** *Conduct an instantiation of the MEDIKAL framework on a real-world medical data set.*

In this thesis, synthetically generated EHR data were considered as part of the MEDIKAL framework's instantiations, from which methodological insight was gleaned as well as insight into potential use cases. A more representative ascertainment of the framework's efficacy necessitates its application to real-world patient data. Although synthetic datasets are useful in respect of demonstrating conceptual utility, they do not necessarily encompass all of the intricacies embedded within real-world medical data. Furthermore, the algorithmic output and accompanying insights derived from the MEDIKAL framework should be validated by means of a qualified medical practitioner in order to ascertain the efficacy and reliability of the framework in a real-world clinical context.

**Suggestion III** *Tailor the framework to a specific medical domain.*

The medical domain encompasses a broad range of markedly complex dynamics and workings, each of which is accompanied by innate intricacies emanating from the different sub-domains. The framework proposed in this thesis was designed in order to be generic in nature so that it may be applied to various use-cases within the medical domain. Tailoring the MEDIKAL framework to a distinct medical domain, however, may result in enhanced performance — it is proffered that the task of deriving inferential patterns could simplify if the problem scope is narrowed. A domain-specific approach may enable effective and contextual data interpretation and subsequent preparation, potentially resulting in more meaningful and actionable outcomes. Modules pertaining to the KG construction component may be customised in order to extract domain specific entities from large data sets, *e.g.* extracting only neurological conditions from a data set. It should be noted that such an extension should be accompanied by specialised domain expertise during the development and implementation of the modified framework.

**Suggestion IV** *Extending the framework in respect of feature selection.*

Clinical data sets often comprise various features such as patient demographics, imaging data, genetic profiles, and laboratory results, to name but a few. Although the incorporation of these features can contribute towards developing a comprehensive understanding of a patient's medical history, the link prediction task might be rendered more complex due to the addition of redundant information. By extending the MEDIKAL framework to include feature selection, algorithmic performance could potentially be improved by removing inconsequential features. Feature selection techniques can significantly enhance the performance of predictive models by reducing dimensionality, mitigating overfitting, and improving accuracy. Moreover, reduced computational expenditure would also accompany a reduced feature set.

**Suggestion V** *Conduct additional hyperparameter tuning in respect of medication prescription use cases.*

In the case of the second and third instantiations, hyperparameter tuning was not performed — the favourable hyperparameter values identified during the first instantiation

was adopted, due to time restrictions. Consequently, there is additional scope for improvement in respect of each GNN. By conducting additional hyperparameter tuning in respect of the various GNN architectures (for this particular use case), improved algorithmic performance can be achieved and, in doing so, enhance decision support.

---

## References

- [1] ABBAS K, ABBASI A, DONG S, NIU L, YU L, CHEN B, CAI S.-M & HASAN Q, 2021, *Application of network link prediction in drug discovery*, BMC Bioinformatics, **22(187)**, pp. 1–21.
- [2] ABDI H & WILLIAMS LJ, 2010, *Principal component analysis*, Wiley Interdisciplinary Reviews: Computational Statistics, **2(4)**, pp. 433–459.
- [3] ABU-SALIH B, AL-QURISHI M, ALWESHAH M, AL-SMADI M, ALFAYEZ R & SAADEH H, 2023, *Healthcare knowledge graph construction: A systematic review of the state-of-the-art, open issues, and opportunities*, Journal of Big Data, **10(81)**, pp. 1–32.
- [4] ADAMIC LA & ADAR E, 2003, *Friends and neighbors on the web*, Social Networks, **25(3)**, pp. 211–230.
- [5] AGRAWAL M, ZITNIK M & LESKOVEC J, 2018, *Large-scale analysis of disease pathways in the human interactome*, Pacific Symposium on Biocomputing, **23**, pp. 111–122.
- [6] AHALT SC, CHUTE CG, FECHO K, GLUSMAN G, HADLOCK J, TAYLOR CO, PFAFF ER, ROBINSON PN, SOLBRIG H, TA C, TATONETTI N, WENG C & CONSORTIUM TBBDT, 2019, *Clinical data: Sources and types, regulatory constraints, applications*, Clinical and Translational Science, **12(4)**, pp. 329–333.
- [7] AHMED A, SHERVASHIDZE N, NARAYANAMURTHY S, JOSIFOVSKI V & SMOLA AJ, 2013, *Distributed large-scale natural graph factorisation*, Proceedings of the 22<sup>nd</sup> International Conference on World Wide Web, Rio de Janeiro, pp. 37–48.
- [8] ALEMI M, HAGHIGHI H & SHAHRIVARI S, 2017, *CCFinder: Using Spark to find clustering coefficient in big graphs*, The Journal of Supercomputing, **73**, pp. 4683–4710.
- [9] ALJAAF A, AL-JUMEILY D, HUSSAIN A, FERGUS P, AL-JUMAILY M & HAMDAN H, 2016, *Partially synthesised dataset to improve prediction accuracy*, Proceedings of the 12<sup>th</sup> Intelligent Computing Theories and Application, Lanzhou, pp. 855–866.
- [10] ALLUA S & THOMPSON CB, 2009, *Inferential statistics*, Air Medical Journal, **28(4)**, pp. 168–171.
- [11] AMOON A, ARAH O & KHEIFETS L, 2019, *The sensitivity of reported effects of EMF on childhood leukemia to uncontrolled confounding by residential mobility: a hybrid simulation study and an empirical analysis using CAPS data*, Cancer Causes Control, **30(8)**, pp. 901–908.
- [12] ANDERSON R, 2006, *Under threat: patient confidentiality and NHS computing*, Drugs and Alcohol Today, **6(4)**, pp. 13–17.
- [13] AOUAY S, JAMOSSI S & GARGOURI F, 2014, *Feature based link prediction*, Proceedings of the IEEE/ACS 11<sup>th</sup> International Conference on Computer Systems and Applications, Nova Scotia, pp. 523–527.

- [14] APKON M & SINGHAVIRANON P, 2001, *Impact of an electronic information system on physician workflow and data collection in the intensive care unit*, *Intensive Care Medicine*, **27**, pp. 122–30.
- [15] AUTOSCRIBER, 2021, *Ontwikkeld in samenwerking met de arts*, [Online], [Cited March 2022], Available from <https://autoscriber.com/missie.html>.
- [16] AZIZ F, CARDOSO V, BRAVO MERODIO L, RUSS D, PENDLETON S, WILLIAMS J, ACHARJEE A & GKOUTOS G, 2021, *Multimorbidity prediction using link prediction*, [Online], [Cited June 2023], Available from <https://github.com/azizfurqan/LPBN>.
- [17] AZIZ F, CARDOSO V, BRAVO MERODIO L, RUSS D, PENDLETON S, WILLIAMS J, ACHARJEE A & GKOUTOS G, 2021, *Multimorbidity prediction using link prediction*, *Scientific Reports*, **11(16392)**, pp. 1–11.
- [18] BADILLO S, BANFAI B, BIRZELE F, DAVYDOV II, HUTCHINSON L, KAM-THONG T, SIEBOURG-POLSTER J, STEIERT B & ZHANG JD, 2020, *An introduction to machine learning*, *Clinical Pharmacology & Therapeutics*, **107(4)**, pp. 871–885.
- [19] BALAKRISHNAN R & RANGANATHAN K, 2012, *A textbook of graph theory*, 2<sup>nd</sup> Edition, Springer, New York City (NY).
- [20] BANKS J, CARSON J, NELSON B & NICOL D, 2010, *Discrete event system simulation*, 5<sup>th</sup> Edition, Pearson, Upper Saddle River (NJ).
- [21] BARABÁSI A, JEONG H, NÉDA Z, RAVASZ E, SCHUBERT A & VICSEK T, 2002, *Evolution of the social network of scientific collaborations*, *Physica A: Statistical Mechanics and its Applications*, **311(3)**, pp. 590–614.
- [22] BASTIAN M, HEYMANN S & JACOMY M, 2009, *Gephi: An open source software for exploring and manipulating networks*, *Proceedings of the 3<sup>rd</sup> International AAAI Conference on Web and Social Media*, San Jose (CA), pp. 361–362.
- [23] BATINI C, LENZERINI M & NAVATHE SB, 1986, *A comparative analysis of methodologies for database schema integration*, *ACM Computing Surveys*, **18(4)**, pp. 323–364.
- [24] BEAN D, WU H, DZAHINI O, BROADBENT M, STEWART R & DOBSON R, 2017, *Knowledge graph prediction of unknown adverse drug reactions and validation in electronic health records*, *Scientific Reports*, **7(16416)**, pp. 1–11.
- [25] BELKIN M & NIYOGE P, 2001, *Laplacian eigenmaps and spectral techniques for embedding and clustering*, *Advances in Neural Information Processing Systems*, **14(6)**, pp. 1–7.
- [26] BELLMAN R, 1958, *On a routing problem*, *Quarterly of Applied Mathematics*, **16(1)**, pp. 87–90.
- [27] BENCHETTARA N, KANAWATI R & ROUVEIROL C, 2010, *Supervised machine learning applied to link prediction in bipartite social networks*, *Proceedings of the 2010 International Conference on Advances in Social Network Analysis and Mining*, Odense, pp. 326–330.
- [28] BERKSON J, 1944, *Application of the logistic function to bio-assay*, *Journal of the American Statistical Association*, **39(227)**, pp. 357–365.
- [29] BIRKHEAD GS, KLOMPAS M & SHAH NR, 2015, *Uses of electronic health records for public health surveillance to advance public health*, *Annual Review of Public Health*, **36**, pp. 345–359.
- [30] BISHOP CM, 2006, *Pattern recognition and machine learning*, Springer Science & Business Media, New York (NY).

- [31] BODENREIDER O, CORNET R & VREEMAN DJ, 2018, *Recent developments in clinical terminologies - SNOMED CT, LOINC, and RxNorm*, Yearbook of Medical Informatics, **27**, pp. 129–139.
- [32] BOONSTRA A, BODDY D & BELL S, 2008, *Stakeholder management in IOS projects: Analysis of an attempt to implement an electronic patient file*, European Journal of Information Systems, **17(2)**, pp. 100–111.
- [33] BORDES A, USUNIER N, GARCIA-DURAN A, WESTON J & YAKHNENKO O, 2013, *Translating embeddings for modeling multi-relational data*, Proceedings of the 29<sup>th</sup> Conference on Advances in Neural Information Processing Systems, Lake Tahoe (NV), pp. 1–9.
- [34] BORGATTI S & HALGIN D, 2011, *Analyzing affiliation networks*, The Sage Handbook of Social Network Analysis, **1**, pp. 417–433.
- [35] BRANDES U, EIGLSPERGER M, HERMAN I, HIMSOLT M & MARSHALL MS, 2002, *GraphML progress report: Structural layer proposal*, Proceedings of the 9<sup>th</sup> International Symposium on Graph Drawing, Vienna, pp. 501–512.
- [36] BREIMAN L, 2001, *Random Forests*, Machine Learning, **45**, pp. 5–32.
- [37] BRIN S & PAGE L, 1998, *The anatomy of a large-scale hypertextual web search engine*, Proceedings of the 7<sup>th</sup> International Conference on World Wide Web, Brisbane, pp. 107–117.
- [38] BRODY S, ALON U & YAHAV E, 2022, *How attentive are graph attention networks?*, Proceedings of the International Conference on Learning Representations, Virtual, pp. 1–26.
- [39] BRONSTEIN MM, BRUNA J, LECUN Y, SZLAM A & VANDERGHEYNST P, 2017, *Geometric deep learning: Going beyond Euclidean data*, IEEE Signal Processing Magazine, **34(4)**, pp. 18–42.
- [40] CAI H, ZHENG VW & CHANG KC.-C, 2018, *A comprehensive survey of graph embedding: Problems, techniques, and applications*, IEEE Transactions on Knowledge and Data Engineering, **30(9)**, pp. 1616–1637.
- [41] CANNISTRACI C, ALANIS-LOBATO G & RAVASI T, 2013, *From link-prediction in brain connectomes and protein interactomes to the local-community-paradigm in complex networks*, Scientific Reports, **3(1613)**, pp. 1–14.
- [42] CAO S, LU W & XU Q, 2015, *GraRep: Learning graph representations with global structural information*, Proceedings of the 24<sup>th</sup> ACM International Conference on Information and Knowledge Management, Melbourne, pp. 891–900.
- [43] CASTEIGTS A, FLOCCHINI P, QUATTROCIOCCHI W & SANTORO N, 2012, *Time-varying graphs and dynamic networks*, International Journal of Parallel, Emergent and Distributed Systems, **27(5)**, pp. 387–408.
- [44] CHAMI I, ABU-EL-HAJJA S, PEROZZI B, RÉ C & MURPHY K, 2022, *Machine learning on graphs: A model and comprehensive taxonomy*, Journal of Machine Learning Research, **23(89)**, pp. 1–64.
- [45] CHANDRASEKARAN B, JOSEPHSON J & BENJAMINS VR, 1999, *What are ontologies, and why do we need them?*, Intelligent Systems and their Applications, IEEE, **14**, pp. 20–26.
- [46] CHANG Y.-J & KAO H.-Y, 2012, *Link prediction in a bipartite network using Wikipedia revision information*, Proceedings of the 2012 Conference on Technologies and Applications of Artificial Intelligence, Tainan, pp. 50–55.

- [47] CHEN B, LI J, LU G, YU H & ZHANG D, 2020, *Label co-occurrence learning with graph convolutional networks for multi-label chest x-ray image classification*, IEEE Journal of Biomedical and Health Informatics, **24(8)**, pp. 2292–2302.
- [48] CHEN H, CAO G, CHEN J & DING J, 2019, *A practical framework for evaluating the quality of knowledge graph*, Proceedings of the 2019 China Conference on Knowledge Graph and Semantic Computing, Hangzhou, pp. 111–122.
- [49] CHEN J, CHUN D, PATEL M, CHIANG E & JAMES J, 2019, *The validity of synthetic clinical data: a validation study of a leading synthetic data generator (Synthea) using clinical quality measures*, BMC Medical Informatics and Decision Making, **19(44)**, pp. 1–9.
- [50] CHEN L, GAO M, LI B, LIU W & CHEN B, 2018, *Detect potential relations by link prediction in multi-relational social networks*, Decision Support Systems, **115**, pp. 78–91.
- [51] CHEN R, 2018, *Tackling chronic diseases via computational phenotyping: Algorithms, tools and applications*, PhD Dissertation, Georgia Institute of Technology, Atlanta (GA).
- [52] CHEN Y, WU Y, MA S & KING I, 2020, *A literature review of recent graph embedding techniques for biomedical data*, Proceedings of the 27<sup>th</sup> International Conference on Neural Information Processing, Virtual, pp. 21–29.
- [53] CHENG B, ZHANG J, LIU H, CAI M & WANG Y, 2021, *Research on medical knowledge graph for stroke*, Journal of Healthcare Engineering, **2021**, pp. 1–10.
- [54] CHO K, VAN MERRIËNBOER B, GULCEHRE C, BAHDANAU D, BOUGARES F, SCHWENK H & BENGIO Y, 2014, *Learning phrase representations using RNN encoder-decoder for statistical machine translation*, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Doha, pp. 1724–1734.
- [55] CHOI W & LEE H, 2021, *Identifying disease-gene associations using a convolutional neural network-based model by embedding a biological knowledge graph with entity descriptions*, PLoS One, **16(10)**, pp. 1–27.
- [56] COHEN DJ, DORR DA, KNIERIM K, DUBARD CA, HEMLER JR, HALL JD, MARINO M, SOLBERG LI, MCCONNELL KJ & NICHOLS LM, 2018, *Primary care practices' abilities and challenges in using electronic health record data for quality improvement*, Health Affairs, **37(4)**, pp. 635–643.
- [57] CONOVER W, 1998, *Practical nonparametric statistics*, 3<sup>rd</sup> Edition, Wiley, Hoboken (NJ).
- [58] COOLEY P, BROWN S & CAJKA J, 2011, *The role of subway travel in an influenza epidemic: A New York City simulation*, Journal of Urban Health, **88**, pp. 982–995.
- [59] COXETER H, 1969, *Introduction to geometry*, 1<sup>st</sup> Edition, John Wiley & Sons, New York (NY).
- [60] CSÁRDI G & NEPUSZ T, 2006, *The igraph software package for complex network research*, InterJournal Complex Systems, **1695**, pp. 1–10.
- [61] CUKIERSKI W, HAMNER B & YANG B, 2011, *Graph-based features for supervised link prediction*, Proceedings of the 2011 International Joint Conference on Neural Networks, San Jose (CA), pp. 1237–1244.
- [62] DAMINELLI S, THOMAS JM, DURÁN C & CANNISTRACI CV, 2015, *Common neighbours and the local-community-paradigm for topological link prediction in bipartite networks*, [Online], [Cited June 2023], Available from <https://sites.google.com/site/carlovittoriocannistraci/6-datasets-and-matlab-code/cannistraci-based-link-predictors-lcp-corr-and-lcdp-for-bipartite-networks>.

- [63] DAMINELLI S, THOMAS JM, DURÁN C & CANNISTRACI CV, 2015, *Common neighbours and the local-community-paradigm for topological link prediction in bipartite networks*, New Journal of Physics, **17**, pp. 1–11.
- [64] DAVIS DA, CHAWLA NV, BLUMM N, CHRISTAKIS NA & BARABÁSI A.-L, 2008, *Predicting individual disease risk based on medical history*, Proceedings of the 17<sup>th</sup> ACM Conference on Information and Knowledge Management, Napa Valley (CA), pp. 769–778.
- [65] DAVIS J & GOADRICH M, 2006, *The relationship between precision-recall and roc curves*, Proceedings of the 23<sup>rd</sup> International Conference on Machine Learning, Pittsburgh (PA), pp. 233–240.
- [66] DAVIS P, LAY-YEE R & PEARSON J, 2010, *Using micro-simulation to create a synthesised data set and test policy options: The case of health service effects under demographic ageing*, Health Policy, **97(2)**, pp. 267–274.
- [67] DEL VALLE EPG, SANTAMARÍA LP, GARCÍA GL, ZANIN M, RUIZ EM & RODRÍGUEZ-GONZÁLEZ A, 2021, *A meta-path-based prediction method for disease comorbidities*, Proceedings of the 34<sup>th</sup> International Symposium on Computer-Based Medical Systems, Aveiro, pp. 219–224.
- [68] DEO N, 2016, *Graph theory with applications to engineering and computer science*, 1<sup>st</sup> Edition, Dover Publications Inc, Mineola (NY).
- [69] DESMARAIS BA & CRANMER SJ, 2011, *Forecasting the locational dynamics of transnational terrorism: A network analytic approach*, Proceedings of the 1<sup>st</sup> European Intelligence and Security Informatics Conference, Athens, pp. 171–177.
- [70] DI BATTISTA G, 1999, *Graph drawing: Algorithms for the visualisation of graphs*, Prentice Hall, Upper Saddle River (NJ).
- [71] DIJKSTRA EW, 1959, *A note on two problems in connexion with graphs*, Numerische Mathematik, **1(1)**, pp. 269–271.
- [72] DING H, TAKIGAWA I, MAMITSUKA H & ZHU S, 2014, *Similarity-based machine learning methods for predicting drug–target interactions: A brief review*, Briefings in Bioinformatics, **15(5)**, pp. 734–747.
- [73] DIPIETRO R & HAGER GD, 2020, *Chapter 21 - Deep learning: RNNs and LSTM*, pp. 503–519 in ZHOU K, RUECKERT D & FICHTINGER G (EDS), *Handbook of Medical Image Computing and Computer Assisted Intervention*, Academic Press.
- [74] DONG Y, CHAWLA NV & SWAMI A, 2017, *metapath2vec: Scalable representation learning for heterogeneous networks*, Proceedings of the 23<sup>rd</sup> ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, pp. 135–144.
- [75] DONG Y, TANG J, WU S, TIAN J, CHAWLA NV, RAO J & CAO H, 2012, *Link prediction and recommendation across heterogeneous social networks*, Proceedings of the 12<sup>th</sup> International Conference on Data Mining, Brussels, pp. 181–190.
- [76] DONNAT C, ZITNIK M, HALLAC D & LESKOVEC J, 2017, *Learning structural node embeddings via diffusion wavelets*, Proceedings of the 24<sup>th</sup> ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, pp. 1320–1329.
- [77] DUBE K & GALLAGHER T, 2013, *Approach and method for generating realistic synthetic electronic healthcare records for secondary use*, Proceedings of the 3<sup>rd</sup> International Symposium on Foundations of Health Informatics Engineering and Systems, Macau, pp. 69–86.



- [78] EBRAHIMI F & GOLPAYEGANI S, 2016, *Personalized recommender system based on social relations*, Proceedings of the 24<sup>th</sup> Iranian Conference on Electrical Engineering, Shiraz, pp. 218–223.
- [79] EFRON B, 1979, *Bootstrap methods: Another look at the jackknife*, The Annals of Statistics, **7(1)**, pp. 1–26.
- [80] EHNFORSS M, FLORIN J & EHRENBORG A, 2003, *Applicability of the international classification of nursing practice (ICNP) in the areas of nutrition and skin care*, International Journal of Nursing Terminologies and Classifications, **14(1)**, pp. 5–18.
- [81] EL EMAM K, JONKER E, ARBUCKLE L & MALIN B, 2011, *A systematic review of re-identification attacks on health data*, PLoS One, **6(12)**, pp. 1–12.
- [82] ELKABANI I & KHACHFEH RAA, 2015, *Homophily-based link prediction in the facebook online social network: a rough sets approach*, Journal of Intelligent Systems, **24(4)**, pp. 491–503.
- [83] ELLSON J, GANSNER E, KOUTSOFIOS E, NORTH S & WOODHULL G, 2001, *Graphviz – Open source graph drawing tools*, Proceedings of the 2001 Graph Drawing Conference, Vienna, pp. 483–484.
- [84] ENANORIA W, LIU F, ZIPPRICH J, HARRIMAN K, ACKLEY S, BLUMBERG S, WORDEN L & PORCO T, 2016, *The effect of contact investigations and public health interventions in the control and prevention of measles transmission: A simulation study*, PLoS One, **11(12)**, pp. 1–12.
- [85] FADER A, SODERLAND S & ETZIONI O, 2011, *Identifying relations for open information extraction*, Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, Edinburgh, pp. 1535–1545.
- [86] FENSEL D, ŞİMŞEK U, ANGELE K, HUAMAN E, KÄRLE E, PANASIUK O, TOMA I, UMBRICH J, WAHLER A & FENSEL D, 2020, *Introduction: What is a knowledge graph?*, Knowledge graphs: Methodology, tools and selected use cases, pp. 1–10.
- [87] FEY M & LENSSEN JE, 2019, *Fast graph representation learning with PyTorch Geometric*, Proceedings of the ICLR Workshop on Representation Learning on Graphs and Manifolds, La Jolla (CA), pp. 1–9.
- [88] FISHER RA, 1935, *The Design of Experiments*, Oliver and Boyd, Edinburgh.
- [89] FONTOURA M, 2020, *A systematic approach for framework development*, PhD Dissertation, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro.
- [90] FORTUNATO S, 2010, *Community detection in graphs*, Physics Reports, **486(3)**, pp. 75–174.
- [91] FRIEDMAN M, 1940, *A comparison of alternative tests of significance for the problem of  $m$  rankings*, The Annals of Mathematical Statistics, **11(1)**, pp. 86–92.
- [92] FRUCHTERMAN TMJ & REINGOLD EM, 1991, *Graph drawing by force-directed placement*, Software: Practice and Experience, **21(11)**, pp. 1129–1164.
- [93] GAO F, MUSIAL K, COOPER C & TSOKA S, 2015, *Link prediction methods and their accuracy for different social networks and network metrics*, Scientific Programming, **2015**, pp. 1–13.
- [94] GAO J, LYU T, XIONG F, WANG J, KE W & LI Z, 2020, *MGNN: A multimodal graph neural network for predicting the survival of cancer patients*, Proceedings of the 43<sup>rd</sup> International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual, pp. 1697–1700.

- [95] GAO S, DENOYER L & GALLINARI P, 2011, *Link pattern prediction with tensor decomposition in multi-relational networks*, Proceedings of the 2011 IEEE Symposium on Computational Intelligence and Data Mining, Paris, pp. 333–340.
- [96] GARCÍA Y, CHRISTEN J & CAPISTRÁN M, 2015, *A bayesian outbreak detection method for influenza-like illness*, BioMed Research International, **2015(4)**, pp. 1–10.
- [97] GIANFRANCESCO M & GOLDSTEIN N, 2021, *A narrative review on the validity of electronic health record-based research in epidemiology*, BMC Medical Research Methodology, **21**, pp. 1–10.
- [98] GILMER J, SCHOENHOLZ SS, RILEY PF, VINYALS O & DAHL GE, 2017, *Neural message passing for quantum chemistry*, Proceedings of the 34<sup>th</sup> International Conference on Machine Learning, Sydney, pp. 1263–1272.
- [99] GONZALES A, GURUSWAMY G & SMITH S, 2023, *Synthetic data in health care: A narrative review*, PLoS Digital Health, **2**, pp. 1–16.
- [100] GOODFELLOW IJ, BENGIO Y & COURVILLE A, 2016, *Deep learning*, MIT Press, Cambridge (MA).
- [101] GOSSET WS, 1908, *The probable error of a mean*, Biometrika, **6(1)**, pp. 1–25.
- [102] GOYAL P & FERRARA E, 2018, *Graph embedding techniques, applications, and performance: A survey*, Knowledge-Based Systems, **151**, pp. 78–94.
- [103] GRIMSON J, GRIMSON W & HASSELBRING W, 2000, *The SI challenge in health care*, Communications of the ACM, **43(6)**, pp. 49–55.
- [104] GRISHMAN R & SUNDHEIM BM, 1996, *Message understanding conference- 6: A brief history*, Proceedings of the 16<sup>th</sup> International Conference on Computational Linguistics, Hanoi, pp. 466–471.
- [105] GROVER A & LESKOVEC J, 2016, *node2vec: Scalable feature learning for networks*, Proceedings of the International Conference on Knowledge Discovery & Data Mining, San Francisco (CA), pp. 855–864.
- [106] GÜNTHER S, KUHN M, DUNKEL M, CAMPILLOS M, SENGER C, PETSALAKI E, AHMED J, URDIALES EG, GEWIESS A & JENSEN LJ, 2007, *SuperTarget and Matador: Resources for exploring drug-target relationships*, Nucleic Acids Research, **36(1)**, pp. 919–922.
- [107] HAGBERG AA, SCHULT DA & SWART PJ, 2008, *Exploring network structure, dynamics, and function using NetworkX*, Proceedings of the 7<sup>th</sup> Python in Science Conference (SciPy2008), Pasadena (CA), pp. 11–15.
- [108] HAMILTON WL, *Graph representation learning*, Synthesis Lectures on Artificial Intelligence and Machine Learning, **14(3)**, pp. 1–159.
- [109] HAMILTON WL, YING R & LESKOVEC J, 2017, *Representation learning on graphs: Methods and applications*, IEEE Data Engineering Bulletin, **40(3)**, pp. 52–74.
- [110] HAMILTON WL, YING Z & LESKOVEC J, 2017, *Inductive representation learning on large graphs*, Proceedings of the 31<sup>st</sup> Conference on Neural Information Processing Systems, Long Beach (CA), pp. 1025–1035.
- [111] HARNOUNE A, RHANOU M, MIKRAM M, YOUSFI S, ELKAIMBILLAH Z & EL ASRI B, 2021, *BERT based clinical knowledge extraction for biomedical knowledge graph construction and analysis*, Computer Methods and Programs in Biomedicine Update, **1**, pp. 1–12.
- [112] HARPER FM & KONSTAN JA, 2015, *The MovieLens datasets: History and context*, ACM Transactions on Interactive Intelligent Systems, **5(4)**, pp. 1–19.

- [113] HASAN MA, CHAOJI V, SALEM S & ZAKI M, 2006, *Link prediction using supervised learning*, Proceedings of the SDM 06 workshop on Link Analysis, Counterterrorism and Security, Bethesda (MD), pp. 1–10.
- [114] HASHEMIAN M, STANLEY K & OSGOOD N, 2012, *Leveraging H1N1 infection transmission modeling with proximity sensor microdata*, BMC Medical Informatics and Decision Making, **12**, pp. 35.
- [115] HASMAN A, DE BRUIJN L & ARENDS J, 2001, *Evaluation of a method that supports pathology report coding*, Methods of Information in Medicine, **40(4)**, pp. 293–297.
- [116] HÄYRINEN K, SARANTO K & NYKÄNEN P, 2008, *Definition, structure, content, use and impacts of electronic health records: A review of the research literature*, International Journal of Medical Informatics, **77(5)**, pp. 291–304.
- [117] HEINEMAN GT, POLLICE G & SELKOW S, 2008, *Algorithms in a nutshell*, 1<sup>st</sup> Edition, O’Reilly Media Inc, Sebastopol (CA).
- [118] HENNESSY D, SANMARTIN C, EFTEKHARY S, PLAGER L, JONES J, ONATE K, MCEVOY N, HICKS C & DEBER R, 2015, *Creating a synthetic database for use in microsimulation models to investigate alternative health care financing strategies in Canada*, International Journal of Microsimulation, **8**, pp. 41–74.
- [119] HINTON GE & ROWEIS S, 2002, *Stochastic Neighbor Embedding*, Proceedings of the 2002 Conference on Advances in Neural Information Processing Systems, Vancouver, pp. 883–840.
- [120] HODLER A & NEEDHAM M, 2021, *Graph data science for dummies Neo4j special edition*, 1<sup>st</sup> Edition, John Wiley & Sons, Hoboken (NJ).
- [121] HOLDER A & NEEDHAM M, 2019, *Graph algorithms*, 1<sup>st</sup> Edition, O’Reilly Media Inc, Sebastopol (CA).
- [122] HOLLAND JH, 1975, *Adaptation in natural and artificial systems*, 2<sup>nd</sup> Edition, University of Michigan Press, Ann Arbor (MI).
- [123] HOLLANDER M, WOLFE D & CHICKEN E, 2013, *Nonparametric statistical methods*, John Wiley & Sons, New York (NY).
- [124] HUANG L, GUAN H, LIANG Y, GUAN R & FENG X, 2021, *COVID-19 knowledge graph for drug and vaccine development*, Proceedings of the 2021 IEEE International Conference on Bioinformatics and Biomedicine, Virtual, pp. 328–333.
- [125] HUANG Z, YANG J, VAN HARMELEN F & HU Q, 2017, *Constructing knowledge graphs of depression*, Proceedings of the Health Information Science, Moscow.
- [126] HURWITZ J & KIRSCH D, 2018, *Machine learning for dummies, IBM limited edition*, 1<sup>st</sup> Edition, John Wiley & Sons, Hoboken (NJ).
- [127] JACCARD P, 1901, *Distribution de la flore alpine dans le bassin des dranses et dans quelques régions voisines*, Bulletin de la Societe Vaudoise des Sciences Naturelles, **37(140)**, pp. 241–272.
- [128] JAHROMI AH & TAHERI M, 2017, *A non-parametric mixture of Gaussian naive Bayes classifiers based on local independent features*, Proceedings of the Artificial Intelligence and Signal Processing Conference, Indore, pp. 209–212.
- [129] JAIN AK, 2010, *Data clustering: 50 years beyond K-means*, Pattern Recognition Letters, **31(8)**, pp. 651–666.

- [130] JALILI M, OROUSKHANI Y, ASGARI M, ALIPOURFARD N & PERC M, 2017, *Link prediction in multiplex online social networks*, Royal Society Open Science, **4(2)**, pp. 1–11.
- [131] JAMES G, WITTEN D, HASTIE T & TIBSHIRANI R, 2013, *An introduction to statistical learning with applications in R*, Springer, New York (NY).
- [132] JAMISON R, RAYMOND S, LEVINE J, SLAWSBY E, NEDELJKOVIĆ S & KATZ N, 2001, *Electronic diaries for monitoring chronic pain: 1-year validation study*, Pain, **91**, pp. 277–285.
- [133] JAYALATCHUMY D & THAMBIDURAI P, 2017, *Prediction of diseases using hadoop in big data - A modified approach*, Proceedings of the 6<sup>th</sup> Computer Science Online Conference, Virtual, pp. 229–238.
- [134] KAMADA T & KAWAI S, 1989, *An algorithm for drawing general undirected graphs*, Information Processing Letters, **31(1)**, pp. 7–15.
- [135] KAMESHWARAN K & MALARVIZHI K, 2014, *Survey on clustering techniques in data mining*, International Journal of Computer Science and Information Technologies, **5(2)**, pp. 2272–2276.
- [136] KANEHISA M, GOTO S, HATTORI M, AOKI-KINOSHITA KF, ITOH M, KAWASHIMA S, KATAYAMA T, ARAKI M & HIRAKAWA M, 2006, *From genomics to chemical genomics: New developments in KEGG*, Nucleic Acids Research, **34(1)**, pp. D354–D357.
- [137] KATZ L, 1953, *A new status index derived from sociometric analysis*, Psychometrika, **18(1)**, pp. 39–43.
- [138] KAZMAIER J, 2020, *A framework for evaluating unstructured text data using sentiment analysis*, PhD Dissertation, Stellenbosch University, Stellenbosch.
- [139] KAZMAIER J, Chief Executive Officer at Autoscriber, [Personal Communication], Contactable at [jacqueline@autoscriber.com](mailto:jacqueline@autoscriber.com).
- [140] KELLEHER JD, NAMEE BM & D'ARCY A, 2015, *Fundamentals of machine learning for predictive data analytics*, 1<sup>st</sup> Edition, Massachusetts Institute of Technology, Cambridge (MA).
- [141] KELLER JM, GRAY MR & GIVENS JA, 1985, *A fuzzy K-nearest neighbor algorithm*, IEEE Transactions on Systems, Man, and Cybernetics, **15(4)**, pp. 580–585.
- [142] KENNETH E. KENDALL JEK, 2013, *Systems analysis and design*, 9<sup>th</sup> Edition, Pearson, Upper Saddle River (NJ).
- [143] KENTON JD, MING-WEI C & TOUTANOVA LK, 2019, *Bert: Pre-training of deep bidirectional transformers for language understanding*, Proceedings of the 17<sup>th</sup> Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis (MN), pp. 4171–4186.
- [144] KERAMATFAR A, RAFIEE M & AMIRKHANI H, 2022, *Graph neural networks: A bibliometrics overview*, Machine Learning with Applications, **10**, pp. 1–37.
- [145] KHAN A, UDDIN S & SRINIVASAN U, 2019, *Chronic disease prediction using administrative data and graph theory: The case of type 2 diabetes*, Expert Systems with Applications, **136**, pp. 230–241.
- [146] KIM K, HUR Y, KIM G & LIM H, 2020, *GREG: A global level relation extraction with knowledge graph embedding*, Applied Sciences, **10(3)**, pp. 1–12.

- [147] KIPF TN & WELLING M, 2017, *Semi-supervised classification with graph convolutional networks*, Proceedings of the 5<sup>th</sup> International Conference on Learning Representations, Toulon, pp. 11305–11312.
- [148] KNYAZEVA B, TAYLOR GW & AMER M, 2019, *Understanding attention and generalization in graph neural networks*, Proceedings of the 32<sup>nd</sup> Conference on Advances in Neural Information Processing Systems, Vancouver, pp. 4202–4212.
- [149] KONOPKA BM, 2015, *Biomedical ontologies — A review*, Biocybernetics and Biomedical Engineering, **35(2)**, pp. 75–86.
- [150] KOSSINETIS G & WATTS DJ, 2006, *Empirical analysis of an evolving social network*, Science, **311(5757)**, pp. 88–90.
- [151] KOSSINETIS G & WATTS DJ, 2009, *Origins of homophily in an evolving social network*, American Journal of Sociology, **115(2)**, pp. 405–450.
- [152] KRZYWINSKI M & ALTMAN N, 2014, *Visualizing samples with box plots: use box plots to illustrate the spread and differences of samples*, Nature Methods, **11(2)**, pp. 119–121.
- [153] KUMAR A, 2021, *Graph neural networks explained with examples*, [Online], [Cited February 2023], Available from <https://vitalflux.com/graph-neural-networks-explained-with-examples/>.
- [154] KUMAR A, 2021, *Machine learning – feature selection vs feature extraction*, [Online], [Cited July 2022], Available from <https://vitalflux.com/machine-learning-feature-selection-feature-extraction/>.
- [155] KUMAR A, SINGH SS, SINGH K & BISWAS B, 2020, *Link prediction techniques, applications, and performance: A survey*, Physica A: Statistical Mechanics and its Applications, **553**, pp. 1–19.
- [156] KUMAR P & SHARMA D, 2020, *A potential energy and mutual information based link prediction approach for bipartite networks*, Scientific Reports, **10(10)**, pp. 1–15.
- [157] KUNEGIS J, DE LUCA EW & ALBAYRAK S, 2010, *The link prediction problem in bipartite networks*, Proceedings of the Computational Intelligence for Knowledge-Based Systems Design, and 13<sup>th</sup> International Conference on Information Processing and Management of Uncertainty, Dortmund, pp. 380–389.
- [158] LABS L, 2021, *LUMO Labs makes first TTT.ai pre-seed investment in LUMC innovation partner Autoscriber*, [Online], [Cited February 2022], Available from <https://lumolabs.io/lumo-labs-makes-first-ttt-ai-pre-seed-investment-in-lumc-innovation-partner-autoscriber/>.
- [159] LAWRYNOWICZ A & TRESP V, 2014, *Introducing machine learning*, pp. 35–50 in LEHMANN J & VOELKER J (EDS), *Perspectives on Ontology Learning*.
- [160] LEAL-TAIXÉ L & NIESSNER M, 2018, *Introduction to Deep Learning*, Lecture Notes, Department of Informatics, Technical University of Munich, Munich.
- [161] LENSSEN J & FEY M, 2022, *Link prediction on heterogeneous graphs with PyG*, [Online], [Cited January 2023], Available from [https://medium.com/@pytorch\\_geometric/link-prediction-on-heterogeneous-graphs-with-pyg-6d5c29677c70](https://medium.com/@pytorch_geometric/link-prediction-on-heterogeneous-graphs-with-pyg-6d5c29677c70).
- [162] LESKOVEC J, 2021, *Machine learning with graphs*, Lecture slide from CS224W, Stanford University.
- [163] LEVENSTEIN M, 2017, *The researcher passport: Improving data access and confidentiality protection, global, 2017-2018*, Inter-university Consortium for Political and Social Research, Ann Arbor (MI).

- [164] LI L, WANG P, YAN J, WANG Y, LI S, JIANG J, SUN Z, TANG B, CHANG T.-H, WANG S & LIU Y, 2020, *Real-world data medical knowledge graph: Construction and applications*, Artificial Intelligence in Medicine, **103**, pp. 1–10.
- [165] LI X & CHEN H, 2013, *Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach*, Decision Support Systems, **54(2)**, pp. 880–890.
- [166] LI Y & WU H, 2012, *A clustering method based on k-means algorithm*, Physics Procedia, **25**, pp. 1104–1109.
- [167] LIBEN-NOWELL D & KLEINBERG J, 2003, *The link prediction problem for social networks*, Proceedings of the 12<sup>th</sup> International Conference on Information and Knowledge Management, New Orleans (LA), pp. 556–559.
- [168] LICHTENWALTER R, LUSSIER J & CHAWLA N, 2010, *New perspectives and methods in link prediction*, Proceedings of the 16<sup>th</sup> ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington (DC), pp. 243–252.
- [169] LIN Y, LIU Z, SUN M, LIU Y & ZHU X, 2015, *Learning entity and relation embeddings for knowledge graph completion*, Proceedings of the 29<sup>th</sup> AAAI Conference on Artificial Intelligence, Austin (TX), pp. 2181–2187.
- [170] LIU C, CAO W, WU S, SHEN W, JIANG D, YU Z & WONG H.-S, 2022, *Supervised graph clustering for cancer subtyping based on survival analysis and integration of multi-omic tumor data*, IEEE/ACM Transactions on Computational Biology and Bioinformatics, **19(2)**, pp. 1193–1202.
- [171] LIU C, WANG F, HU J & XIONG H, 2015, *Temporal phenotyping from longitudinal electronic health records: a graph based framework*, Proceedings of the 21<sup>st</sup> ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, pp. 705–714.
- [172] LIU T, PAN X, WANG X, FEENSTRA KA, HERINGA J & HUANG Z, 2020, *Exploring the microbiota-gut-brain axis for mental disorders with knowledge graphs*, Journal of Artificial Intelligence for Medical Sciences, **1**, pp. 30–42.
- [173] LIU W & LÜ L, 2010, *Link prediction based on local random walk*, Europhysic Letter, **89**.
- [174] LIU Z, ZHANG Q.-M, LÜ L & ZHOU T, 2011, *Link prediction in complex networks: A local naïve Bayes model*, Europhysics Letters, **96(4)**, pp. 1–6.
- [175] LOCK M & GORDON D, 2012, *Biomedicine examined*, 13<sup>th</sup> Edition, Kluwer Academic Publishers, Dordrecht.
- [176] LU H & UDDIN S, 2023, *Disease prediction using graph machine learning based on electronic health data: A review of approaches and trends*, Healthcare, **11(7)**, pp. 1–21.
- [177] LU H, UDDIN S, HAJATI F, MONI MA & KHUSHI M, 2021, *A patient network-based machine learning model for disease prediction: The case of type 2 diabetes mellitus*, Applied Intelligence, **52**, pp. pages 2411–2422.
- [178] LÜ L, JIN C.-H & ZHOU T, 2009, *Similarity index based on local paths for link prediction of complex networks*, Physical Review E, **80(4)**, pp. 1–10.
- [179] LÜ L & ZHOU T, 2011, *Link prediction in complex networks: A survey*, Physica A: Statistical Mechanics and its Applications, **390(6)**, pp. 1150–1170.
- [180] LV H, ZHANG B, HU S & XU Z, 2022, *Deep link-prediction based on the local structure of bipartite networks*, Entropy, **24(5)**, pp. 1–11.

- [181] MA X, WANG M, LIU X, YANG Y, ZHENG Y & WANG S, 2022, *InDISP: An interpretable model for dynamic illness severity prediction*, Proceedings of the 27<sup>th</sup> International Conference on Database Systems for Advanced Applications, Virtual, pp. 631–638.
- [182] MA Y, LIANG X, HUANG J & CHENG G, 2017, *Intercity transportation construction based on link prediction*, Proceedings of the 29<sup>th</sup> International Conference on Tools with Artificial Intelligence, Boston (MA), pp. 1135–1138.
- [183] MAAS AL, HANNUN AY & NG AY, 2013, *Rectifier nonlinearities improve neural network acoustic models*, Proceedings of the 30<sup>th</sup> International Conference on Machine Learning, Atlanta (GA), pp. 1–6.
- [184] MALIK KM, KRISHNAMURTHY M, ALOBAIDI M, HUSSAIN M, ALAM F & MALIK G, 2020, *Automated domain-specific healthcare knowledge graph curation framework: Subarachnoid hemorrhage as phenotype*, Expert Systems with Applications, **145**, pp. 1–15.
- [185] MALLAWAARACHCHI V, 2020, *Inductive vs. transductive learning*, [Online], [Cited July 2022], Available from <https://towardsdatascience.com/inductive-vs-transductive-learning-e608e786f7d>.
- [186] MANN M, ILIEVSKI F, ROSTAMI M, AASTHA A & SHBITA B, 2021, *Open drug knowledge graph*, Proceedings of the 2<sup>nd</sup> International Workshop on Knowledge Graph Construction, Virtual, pp. 1–17.
- [187] MARILL KA, GAUHAROU ES, NELSON BK, PETERSON MA, CURTIS RL & GONZALEZ MR, 1999, *Prospective, randomized trial of template-assisted versus undirected written recording of physician records in the emergency department*, Annals of Emergency Medicine, **33(5)**, pp. 500–509.
- [188] MCGUINNESS DL & VAN HARMELEN F, 2004, *OWL web ontology language overview*, W3C recommendation, **10(10)**, pp. 1–19.
- [189] MCINNES L, HEALY J, SAUL N & GROSSBERGER L, 2018, *UMAP: Uniform manifold approximation and projection*, Journal of Open Source Software, **3(29)**, pp. 861.
- [190] MCPHERSON M, SMITH-LOVIN L & COOK JM, 2001, *Birds of a feather: Homophily in social networks*, Annual Review of Sociology, **27(1)**, pp. 415–444.
- [191] METZGER S, 2022, *A beginner's guide to tokens, vectors, and embeddings in NLP*, [Online], [Cited November 2023], Available from <https://medium.com/@saschametzger/what-are-tokens-vectors-and-embeddings-how-do-you-create-them-e2a3e698e037>.
- [192] MIKKELSEN G & AASLY J, 2001, *Concordance of information in parallel electronic and paper based patient records*, International Journal of Medical Informatics, **63**, pp. 123–31.
- [193] MILOSEVIC N & THIELEMANN W, 2023, *Comparison of biomedical relationship extraction methods and models for knowledge graph creation*, Journal of Web Semantics, **75**, pp. 1–12.
- [194] MITCHELL TM, 1997, *Machine learning*, McGraw-Hill Science/Engineering/Math, New York (NY).
- [195] MITRA S & PAL S, 1995, *Fuzzy multi-layer perceptron, inferencing and rule generation*, IEEE Transactions on Neural Networks, **6(1)**, pp. 51–63.
- [196] MOČKUS J, 1975, *On bayesian methods for seeking the extremum*, Proceedings of the 1975 Optimization Techniques IFIP Technical Conference, Novosibirsk, pp. 400–404.
- [197] MOHAMMADHASSANZADEH H, ABIDI SR, VAN WOENSEL W & ABIDI SSR, 2018, *Investigating plausible reasoning over knowledge graphs for semantics-based health data analytics*, Proceedings of the 27<sup>th</sup> International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, Paris, pp. 148–153.

- [198] MOHAPATRA A & WEXLER P, 2009, *Chapter 68 - Web-based databases*, pp. 619–630 in WEXLER P, GILBERT SG, HAKKINEN PJ & MOHAPATRA A (EDS), *Information Resources in Toxicology (Fourth Edition)*, Academic Press, San Diego (CA).
- [199] AL-MOSLMI T, GALLOFRÉ OCAÑA M, L.OPDAHL A & VERES C, 2020, *Named entity extraction for knowledge graphs: A literature overview*, IEEE Access, **8**, pp. 32862–32881.
- [200] MOUMTZOGLU AS, “Facets of the Evolving Healthcare Management Model”, in: *Quality of Healthcare in the Aftermath of the COVID-19 Pandemic*, IGI Global, 2022, pp. 303–321.
- [201] MOVIG K, LEUFKENS H, LENDERINK A & EGBERTS A, 2003, *Validity of hospital discharge International Classification of Diseases (ICD) codes for identifying patients with hyponatremia*, Journal of Clinical Epidemiology, **56(6)**, pp. 530–535.
- [202] MWOGI TS, BIONDICH PG & GRANNIS SJ, 2014, *An evaluation of two methods for generating synthetic HL7 Segments reflecting real-world health information exchange transactions*, Proceedings of the 2014 AMIA Annual Symposium Proceedings, Washington (DC), pp. 1855–1863.
- [203] NACHMAN B & SHIH D, 2020, *Anomaly detection with density estimation*, Physical Review D, **101(7)**, pp. 1–16.
- [204] NAYLOR T & FINGER J, 1967, *Verification of computer simulation models*, Management Science, **14(2)**, pp. 92–106.
- [205] NEL G, 2021, *A brief history and overview of machine learning and artificial intelligence*, (Unpublished) Technical Report 2021-02, Stellenbosch Unit for Operations Research in Engineering, Department of Industrial Engineering, Stellenbosch University, Stellenbosch.
- [206] NELDER JA & WEDDERBURN RW, 1972, *Generalized linear models*, Journal of the Royal Statistical Society Series A: Statistics in Society, **135(3)**, pp. 370–384.
- [207] NELSON C, BOVE R, BUTTE A & BARANZINI S, 2021, *Embedding electronic health records onto a knowledge network recognizes prodromal features of multiple sclerosis and predicts diagnosis*, Journal of the American Medical Informatics Association, **29**, pp. 424–434.
- [208] NELSON SJ, ZENG K, KILBOURNE J, POWELL T & MOORE R, 2011, *Normalized names for clinical drugs: RxNorm at 6 years*, Journal of the American Medical Informatics Association, **18(4)**, pp. 441–448.
- [209] NEMENYI PB, 1963, *Distribution-free multiple comparisons*, Princeton University, Princeton (NJ).
- [210] NEO4J, 2022, *Neo4j graph database*, [Online], [Cited November 2022], Available from <https://neo4j.com/product/neo4j-graph-database/>.
- [211] NEWMAN MEJ, 2001, *Clustering and preferential attachment in growing networks*, Physical Review E, **64**, pp. 1–13.
- [212] NGUFOR C, VAN HOUTEN H, CAFFO BS, SHAH ND & MCCOY RG, 2019, *Mixed effect machine learning: A framework for predicting longitudinal change in hemoglobin A1c*, Journal of Biomedical Informatics, **89**, pp. 56–67.
- [213] NGUYEN TA, PERKINS WA, LAFFEY TJ & PECORA D, 1987, *Knowledge base verification*, AI Magazine, **8(2)**, pp. 69–75.
- [214] NICKEL M, MURPHY KP, TRESP V & GABRILOVICH E, 2015, *A review of relational machine learning for knowledge graphs*, IEEE, **104**, pp. 11–33.



- [215] NICKEL M, TRESP V & KRÖGER P, 2011, *A three-way model for collective learning on multi-relational data*, Proceedings of the 28<sup>th</sup> International Conference on Machine Learning, Bellevue (WAS), pp. 809–816.
- [216] OU M, CUI P, PEI J, ZHANG Z & ZHU W, 2016, *Asymmetric transitivity preserving graph embedding*, Proceedings of the 22<sup>nd</sup> ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco (CA), pp. 1105–1114.
- [217] OU Q, JIN Y.-D, ZHOU T, WANG B.-H & YIN B.-Q, 2007, *Power-law strength-degree correlation from resource-allocation dynamics on weighted networks*, Physical Review E, **75(2)**, pp. 1–6.
- [218] PANDIT S, CHAU P, WANG S & FALOUTSOS C, 2007, *Netprobe: A fast and scalable system for fraud detection in online auction networks*, Proceedings of the 16<sup>th</sup> International World Wide Web Conference, WWW2007, Bannf, pp. 201–210.
- [219] PARSHOTAM H & NEL GS, 2023, *Diagnosis prediction using knowledge graphs*, South African Journal of Industrial Engineering, **34(3)**.
- [220] PEDREGOSA F, VAROQUAUX G, GRAMFORT A, MICHEL V, THIRION B, GRISEL O, BLONDEL M, PRETTENHOFER P, WEISS R, DUBOURG V, VANDERPLAS J, PASSOS A, COURNAPEAU D, BRUCHER M, PERROT M & DUCHESNAY E, 2011, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research, **12**, pp. 2825–2830.
- [221] PENNSTATE, 2023, *Bootstrapping Methods*, Lecture Notes Stat 500, PennState, State College (PA).
- [222] PEROZZI B, AL-RFOU R & SKIENA S, 2014, *Deep walk*, Proceedings of the 20<sup>th</sup> International Conference on Knowledge Discovery and Data Mining, New York (NY), pp. 701–710.
- [223] PHAM T, TAO X, ZHANG J, YONG J, LI Y & XIE H, 2022, *Graph-based multi-label disease prediction model learning from medical data and domain knowledge*, Knowledge-Based Systems, **235**, pp. 1–15.
- [224] PHAM T, TAO X, ZHANG J, YONG J, ZHOU X & GURURAJAN R, 2019, *MeKG: Building a medical knowledge graph by data mining from MEDLINE*, Proceedings of the 12<sup>th</sup> International Conference on Brain Informatics, Haikou, pp. 159–168.
- [225] PREIM B & BOTHA CP, 2013, *Visual Computing for Medicine: Theory, Algorithms, and Applications*, 2<sup>nd</sup> Edition, Morgan Kaufmann Publishers Inc., San Francisco (CA).
- [226] PRIETO-DIAZ R, 1990, *Domain analysis: An introduction*, ACM SIGSOFT Software Engineering Notes, **15(2)**, pp. 47–54.
- [227] PYG, 2023, *Heterogeneous Graph Learning*, [Online], [Cited July 2023], Available from <https://pytorch-geometric.readthedocs.io/en/latest/tutorial/heterogeneous.html>.
- [228] QUINLAN JR, 1986, *Induction of decision trees*, Machine Learning, **1**, pp. 81–106.
- [229] RAGHAVAN UN, ALBERT R & KUMARA S, 2007, *Near linear time algorithm to detect community structures in large-scale networks*, Physical Review E, **76(3)**, pp. 1–11.
- [230] RAGHUNATHAN TE, REITER JP & RUBIN DB, 2003, *Multiple imputation for statistical disclosure limitation*, Journal of Official Statistics, **19**, pp. 1–16.
- [231] RAUT P, KHANDELWAL H & VYAS G, 2020, *A comparative study of classification algorithms for link prediction*, Proceedings of the 2<sup>nd</sup> International Conference on Innovative Mechanisms for Industry Applications, Bangalore, pp. 479–483.

- [232] REED S, OORD A, KALCHBRENNER N, COLMENAREJO SG, WANG Z, CHEN Y, BELOV D & FREITAS N, 2017, *Parallel multiscale autoregressive density estimation*, Proceedings of the 34<sup>th</sup> International Conference on Machine Learning, Sydney, pp. 2912–2921.
- [233] REIMERS N & GUREVYCH I, 2019, *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*, Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Hong Kong, pp. 3982–3992.
- [234] REINER BENAÏM A, ALMOG R, GORELIK Y, HOCHBERG I, NASSAR L, MASHIACH T, KHAMAISSI M, LURIE Y, AZZAM ZS, KHOURY J, KURNIK D & BEYAR R, 2020, *Analyzing medical research results based on synthetic data and their relation to real data results: Systematic comparison from five observational studies*, JMIR Medical Informatics, **8(2)**, pp. 16492.
- [235] REITER JP, 2003, *Inference for partially synthetic, public use microdata sets*, Survey Methodology, **29(2)**, pp. 181–188.
- [236] REYNOLDS D, 2009, *Gaussian mixture models*, pp. 659–663 in LI SZ & JAIN AK (EDS), *Encyclopedia of Biometrics*, Springer US, Boston (MA).
- [237] RISH I, 2001, *An empirical study of the naïve Bayes classifier*, Proceedings of the 2001 IJCAI Workshop on Empirical Methods in Artificial Intelligence, Seattle (WA), pp. 41–46.
- [238] ROBBINS H & MONRO S, 1951, *A stochastic approximation method*, The Annals of Mathematical Statistics, **22(3)**, pp. 400–407.
- [239] ROBINSON I, WEBBER J & EIFREM E, 2015, *Graph databases*, 2<sup>nd</sup> Edition, O’Reily Media Inc, Sebastopol (CA).
- [240] ROCHA R, HUFF S, HAUG P, EVANS D & BRAY B, 1998, *Evaluation of a semantic data model for chest radiology: application of a new methodology*, Methods of Information in Medicine, **37(4)**, pp. 477–490.
- [241] RODRIGUEZ MA & NEUBAUER P, 2012, *The graph traversal pattern*, pp. 29–46 in FLETCHER G, HIDDERS J & LARRIBA-PEY JL (EDS), *Graph Data Management: Techniques and Applications*, IGI Global, Hershey (PA).
- [242] ROSENBLATT M, 1956, *Remarks on some nonparametric estimates of a density function*, The Annals of Mathematical Statistics, pp. 832–837.
- [243] ROSSETTI G, BERLINGERIO M & GIANNOTTI F, 2011, *Scalable link prediction on multidimensional networks*, Proceedings of the 11<sup>th</sup> IEEE International Conference on Data Mining, Vancouver, pp. 979–986.
- [244] ROTMENSCH M, HALPERN Y, TLIMAT A, HORNG S & SONTAG D, 2017, *Learning a health knowledge graph from electronic medical records*, Scientific Reports, **7**, pp. 1–11.
- [245] RUSSEL S & NORVIG P, 2009, *Artificial intelligence: A modern approach*, 3<sup>rd</sup> Edition, Pearson Education, London.
- [246] SACHDEV K & GUPTA MK, 2019, *A comprehensive review of feature based methods for drug-target interaction prediction*, Journal of Biomedical Informatics, **93**, pp. 103159.
- [247] SANH V, DEBUT L, CHAUMOND J & WOLF T, 2019, *DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter*, Proceedings of the 5<sup>th</sup> Workshop on Energy Efficient Machine Learning and Cognitive Computing - Poster presentation.
- [248] SANTOS A, COLAÇO A, NIELSEN A, LILI N, STRAUSS M, GEYER P, COSCIA F, WEWER ALBRECHTSEN N, MUNDT F, JENSEN L & MANN M, 2022, *A knowledge graph to interpret clinical proteomics data*, Nature Biotechnology, **40**, pp. 692–702.

- [249] SCARSELLI F, GORI M, TSOI AC, HAGENBUCHNER M & MONFARDINI G, 2009, *The graph neural network model*, IEEE Transactions on Neural Networks, **20**(1), pp. 61–80.
- [250] SCELLATO S, NOULAS A & MASCOLO C, 2011, *Exploiting place features in link prediction on location-based social networks*, Proceedings of the 17<sup>th</sup> ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego (CA), pp. 1046–1054.
- [251] SCHLICHTKRULL M, KIPF TN, BLOEM P, VAN DEN BERG R, TITOV I & WELLING M, 2017, *Modeling relational data with graph convolutional networks*, Proceedings of the 14<sup>th</sup> Extended Semantic Web Conference, Portorož, pp. 593–607.
- [252] SCHOMBURG I, CHANG A, EBELING C, GREMSE M, HELDT C, HUHNS G & SCHOMBURG D, 2004, *BRENDA, the enzyme database: Updates and major new developments*, Nucleic Acids Research, **32**(1), pp. 431–433.
- [253] SCHRIGER D, BARAFF L, ROGERS W & CRETIN S, 1997, *Implementation of clinical guidelines using a computer charting system. Effect on the initial care of health care workers exposed to body fluids*, Journal of the American Medical Association, **278**(19), pp. 1585–1590.
- [254] SCOTT DW, 1979, *On optimal and data-based histograms*, Biometrika, **66**(3), pp. 605–610.
- [255] SELSAM D, LAMM M, BÜNZ B, LIANG P, DE MOURA L & DILL D, 2019, *Learning a SAT solver from single-bit supervision*, Proceedings of the 2019 International Conference on Learning Representations, New Orleans (LA), pp. 1–12.
- [256] SHANNON P, MARKIEL A, OZIER O, BALIGA NS, WANG JT, RAMAGE D, AMIN N, SCHWIKOWSKI B & IDEKER T, 2003, *Cytoscape: a software environment for integrated models of biomolecular interaction networks*, Genome Research, **13**(11), pp. 2498–2504.
- [257] SHEN W, HAN J & WANG J, 2014, *A probabilistic model for linking named entities in web text with heterogeneous information networks*, Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, Snowbird (UT), pp. 1199–1210.
- [258] SHEN W, WANG J & HAN J, 2015, *Entity linking with a knowledge base: Issues, techniques, and solutions*, IEEE Transactions on Knowledge and Data Engineering, **27**(2), pp. 443–460.
- [259] SHI L, LI S, YANG X, QI J, PAN G & ZHOU B, 2017, *Semantic health knowledge graph: Semantic integration of heterogeneous medical knowledge and services*, BioMed research international, **2017**, pp. 1–12.
- [260] SHI Y, ZHENGJIE H, FENG S, ZHONG H, WANG W & SUN Y, 2021, *Masked label prediction: Unified message passing model for semi-supervised classification*, Proceedings of the 30<sup>th</sup> International Joint Conference on Artificial Intelligence, Montreal, pp. 1548–1554.
- [261] SNOMED, 2017, *snomed-database-loader*, [Online], [Cited May 2022], Available from <https://github.com/IHTSDO/snomed-database-loader>.
- [262] STANOJEVIĆ V, VLAJIĆ S, MILIĆ M & OGNJANOVIĆ M, 2011, *Guidelines for framework development process*, Proceedings of the 7<sup>th</sup> Central and Eastern European Software Engineering Conference, Moscow, pp. 1–9.
- [263] STONE M, 1974, *Cross-validatory choice and assessment of statistical predictions*, Journal of the Royal Statistical Society: Series B (Methodological), **36**(2), pp. 111–133.
- [264] SUN W, CAI Z, LI Y, LIU F, FANG S & WANG G, 2018, *Data processing and text mining technologies on electronic medical records: A review*, Journal of Healthcare Engineering, **2018**, pp. 1–9.

- [265] SUN Z, WANG F & HU J, 2015, *LINKAGE: An approach for comprehensive risk prediction for care management*, Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, pp. 1145–1154.
- [266] SUN Z, YIN H, CHEN H, CHEN T, CUI L & YANG F, 2021, *Disease prediction via graph neural networks*, IEEE Journal of Biomedical and Health Informatics, **25(3)**, pp. 818–826.
- [267] SURENDRA H & MOHAN H, 2017, *A review of synthetic data generation methods for privacy preserving data publishing*, International Journal of Scientific & Technology Research, **6**, pp. 95–101.
- [268] SUTTON RS & BARTO AG (EDS), 1998, *Reinforcement learning: An introduction*, MIT Press, Cambridge (MA).
- [269] SWEENEY L, ABU A & WINN J, 2013, *Identifying participants in the personal genome project by name*, URL: <http://dataprivacylab.org/projects/pgp/>.
- [270] SYMONDS P, HUTCHINSON E, IBBETSON A, TAYLOR J, MILNER J, CHALABI Z, DAVIES M & WILKINSON P, 2019, *MicroEnv: A microsimulation model for quantifying the impacts of environmental policies on population health and health inequalities*, Science of the Total Environment, **697**, pp. 1–10.
- [271] TANGE HJ, HASMAN A, DE VRIES ROBBÉ PF & SCHOUTEN HC, 1997, *Medical narratives in electronic medical records*, International Journal of Medical Informatics, **46(1)**, pp. 7–29.
- [272] TATE A, BELOFF N, AL-RADWAN B, WICKSON J, PADMANABHAN S, WILLIAMS T, STAA T & BLEACH A, 2013, *Exploiting the potential of large databases of electronic health records for research using rapid search algorithms and an intuitive query interface*, Journal of the American Medical Informatics Association, **21**, pp. 1–9.
- [273] TAYEFI M, NGO P, CHOMUTARE T, DALIANIS H, SALVI E, BUDRIONIS A & GODTLIEBSEN F, 2021, *Challenges and opportunities beyond structured data in analysis of electronic health records*, WIREs Computational Statistics, **13(6)**, pp. 1–19.
- [274] TEXIER G, JACKSON ML, SIWE L, MEYNARD JB, DEPARIS X & CHAUDET H, 2017, *Building test data from real outbreaks for evaluating detection algorithms*, PLoS One, **12(9)**, pp. 1–17.
- [275] TIBSHIRANI R, 1996, *Regression shrinkage and selection via the lasso*, Journal of the Royal Statistical Society Series B: Statistical Methodology, **58(1)**, pp. 267–288.
- [276] TRAN HN, 2020, *Multi-relational embedding for knowledge graph representation and analysis*, PhD Dissertation, The Graduate University for Advanced Studies, Hayama.
- [277] TROUILLON T, WELBL J, RIEDEL S, GAUSSIER E & BOUCHARD G, 2016, *Complex embeddings for simple link prediction*, Proceedings of the 33<sup>rd</sup> International Conference on Machine Learning, New York City (NY), pp. 2071–2080.
- [278] VAN ROSSUM G & DRAKE FL, 2009, *Python 3 Reference Manual*, CreateSpace, Scotts Valley (CA).
- [279] VAN VUUREN JH & HENNING MA, 2022, *Graph and network theory: An applied approach using MATHEMATICA*, 1<sup>st</sup> Edition, Springer, Berlin.
- [280] VAPNIK V & CORTES C, 1995, *Support-vector networks*, Machine Learning, **20**, pp. 273–297.

- [281] VASWANI A, SHAZEER N, PARMAR N, USZKOREIT J, JONES L, GOMEZ AN, KAISER L & POLOSUKHIN I, 2017, *Attention is all you need*, Proceedings of the 30<sup>th</sup> Conference on Advances in Neural Information Processing Systems, Long Beach (CA), pp. 1–11.
- [282] VELIČKOVIĆ P, CUCURULL G, CASANOVA A, ROMERO A, LIÒ P & BENGIO Y, 2018, *Graph attention networks*, Proceedings of the 6<sup>th</sup> International Conference on Learning Representations, Vancouver, pp. 1–12.
- [283] VIRTANEN P, GOMMERS R, OLIPHANT TE, HABERLAND M, REDDY T, COURNAPEAU D, BUROVSKI E, PETERSON P, WECKESSER W, BRIGHT J, VAN DER WALT SJ, BRETT M, WILSON J, MILLMAN KJ, MAYOROV N, NELSON ARJ, JONES E, KERN R, LARSON E, CAREY CJ, POLAT İ, FENG Y, MOORE EW, VANDERPLAS J, LAXALDE D, PERKTOLD J, CIMRMAN R, HENRIKSEN I, QUINTERO EA, HARRIS CR, ARCHIBALD AM, RIBEIRO AH, PEDREGOSA F, VAN MULBREGT P & SCIPY 1.0 CONTRIBUTORS, 2020, *SciPy 1.0: Fundamental algorithms for scientific computing in Python*, Nature Methods, **17**, pp. 261–272.
- [284] WALONOSKI J, KRAMER M, NICHOLS J, QUINA A, MOESEL C, HALL D, DUFFETT C, DUBE K, GALLAGHER T & MCLACHLAN S, 2018, *Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record*, Journal of the American Medical Informatics Association, **25(3)**, pp. 230–238.
- [285] WALONOSKI J, KLAUS S, GRANGER E, HALL D, GREGOROWICZ A, NEYARAPALLY G, WATSON A & EASTMAN J, 2020, *Synthea™ Novel coronavirus (COVID-19) model and synthetic data set*, Intelligence-Based Medicine, **1**, pp. 1–8.
- [286] WANG AY.-T, MURDOCK RJ, KAUWE SK, OLIYNYK AO, GURLO A, BRGOCH J, PERSOSON KA & SPARKS TD, 2020, *Machine learning for materials scientists: An introductory guide toward best practices*, Chemistry of Materials, **32(12)**, pp. 4954–4965.
- [287] WANG C, GUO J, ZHAO N, LIU Y, LIU X, LIU G & GUO M, 2020, *A cancer survival prediction method based on graph convolutional network*, IEEE Transactions on NanoBioscience, **19(1)**, pp. 117–126.
- [288] WANG D, LIN J, CUI P, JIA Q, WANG Z, FANG Y, LIU X, YANG J & LIU H, 2019, *A semi-supervised graph attentive network for financial fraud detection*, Proceedings of the 2019 International Conference on Data Mining, Beijing, pp. 598–607.
- [289] WANG M, MA X, SI J, TANG H, WANG H, LI T, OUYANG W, GONG L, YONGZHONG T, HE X, HUANG W & LIU X, 2021, *Adverse drug reaction discovery using a tumor-biomarker knowledge graph*, Frontiers in Genetics, **11**, pp. 1–11.
- [290] WANG Q, LI M, WANG X, PARULIAN N, HAN G, MA J, TU J, LIN Y, ZHANG RH, LIU W, CHAUHAN A, GUAN Y, LI B, LI R, SONG X, FUNG Y, JI H, HAN J, CHANG S.-F, PUSTEJOVSKY J, RAH J, LIEM D, ELSAYED A, PALMER M, VOSS C, SCHNEIDER C & ONYSHKEYVYCH B, 2021, *COVID-19 Literature Knowledge Graph Construction and Drug Repurposing Report Generation*, Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations, Virtual, pp. 66–77.
- [291] WANG R, CHANG M.-C & RADIGAN M, 2020, *Modeling latent comorbidity for health risk prediction using graph convolutional network*, Proceedings of the 33<sup>rd</sup> International Flairs Conference, Miami (FL), pp. 1–6.
- [292] WANG T, QIU RG, YU M & ZHANG R, 2020, *Directed disease networks to facilitate multiple-disease risk assessment modeling*, Decision Support Systems, **129**, pp. 1–14.

- [293] WANG X, CHAI Y, LI H & WU D, 2021, *Link prediction in heterogeneous information networks: An improved deep graph convolution approach*, *Decision Support Systems*, **141**, pp. 1–12.
- [294] WANG Z, ZHANG J, FENG J & CHEN Z, 2014, *Knowledge graph embedding by translating on hyperplanes*, *Proceedings of the 28<sup>th</sup> AAAI Conference on Artificial Intelligence*, Québec, pp. 1112–1119.
- [295] WARD W, 2001, *Speech recognition and production by machines*, pp. 14882–14887 in SMELSER N & BALTES PB (EDS), *International Encyclopedia of the Social & Behavioral Sciences*, Pergamon, Oxford.
- [296] WILSON R, 1996, *Introduction to graph theory*, 4<sup>th</sup> Edition, Longman, Harlow.
- [297] WISHART DS, KNOX C, GUO AC, CHENG D, SHRIVASTAVA S, TZUR D, GAUTAM B & HASSANALI M, 2008, *DrugBank: A knowledgebase for drugs, drug actions and drug targets*, *Nucleic Acids Research*, **36(1)**, pp. 901–906.
- [298] WU H, SONG C, GE Y & GE T, 2022, *Link prediction on complex networks: An experimental survey*, *Data Science and Engineering*, **7**, pp. 253–278.
- [299] WU M, HUANG Y, ZHAO L & HE Y, 2018, *Link prediction based on random forest in signed social networks*, *Proceedings of the 10<sup>th</sup> International Conference on Intelligent Human-Machine Systems and Cybernetics*, Hangzhou, pp. 251–256.
- [300] XIA S, DAI B, LIM E.-P, ZHANG Y & XING C, 2012, *Link prediction for bipartite social networks: The role of structural holes*, *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining*, Istanbul, pp. 153–157.
- [301] XIAO H, HUANG M, HAO Y & ZHU X, 2015, *TransA: An adaptive approach for knowledge graph embedding*, arXiv preprint arXiv:1509.05490.
- [302] XU S, YANG C, SHI C, FANG Y, GUO Y, YANG T, ZHANG L & HU M, 2021, *Topic-aware heterogeneous graph neural network for link prediction*, *Proceedings of the 30<sup>th</sup> ACM International Conference on Information and Knowledge Management*, Gold Coast, pp. 2261–2270.
- [303] XU Z, GLASS K, LAU C, GEARD N, GRAVES P & CLEMENTS A, 2017, *A synthetic population for modelling the dynamics of infectious disease transmission in American Samoa*, *Scientific Reports*, **7**, pp. 1–9.
- [304] YAMADA Y, 2008, *The electronic health record as a primary source of clinical phenotype for genetic epidemiological studies*, *The HUGO Journal*, **2(5)**.
- [305] YANG B, YIH W.-t, HE X, GAO J, DENG L, MESNIL G, HE M, SMOLA AJ & HOVY E, 2015, *Embedding entities and relations for learning and inference in knowledge bases*, *Proceedings of the 54<sup>th</sup> Annual Meeting of the Association for Computational Linguistics*, Berlin, pp. 17–26.
- [306] YANG Y, LICHTENWALTER R & CHAWLA N, 2015, *Evaluating link prediction methods*, *Knowledge and Information Systems*, **45**, pp. 751–782.
- [307] YE Q, HSIEH C.-Y, YANG Z, KANG Y, CHEN J, CAO D, HE S & HOU T, 2021, *A unified drug–target interaction prediction framework based on knowledge graph and recommendation system*, *Nature Communications*, **12(6775)**, pp. 1–12.
- [308] YING R, HE R, CHEN K, EKSOMBATCHAI P, HAMILTON WL & LESKOVEC J, 2018, *Graph convolutional neural networks for web-scale recommender systems*, *Proceedings of the 24<sup>th</sup> ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, London, pp. 974–983.

- [309] YOUNG J, RAIN N & TAGKOPOULOS I, 2022, *Knowledge integration and decision support for accelerated discovery of antibiotic resistance genes*, Nature Communications, **13**, pp. 1–11.
- [310] YU G, TABATABAEI M, MEZEI J, ZHONG Q, CHEN S, LI Z, LI J, SHU L & SHU Q, 2022, *Improving chronic disease management for children with knowledge graphs and artificial intelligence*, Expert Systems with Applications, **201**, pp. 1–12.
- [311] YU LR & VEENSTRA TD, 2010, *Essentials of genomic and personalized medicine*, 1<sup>st</sup> Edition, Academic Press, Amsterdam.
- [312] YU T, YIN H & ZHU Z, 2018, *Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting*, IEEE Transactions on Intelligent Transportation Systems, **19**(1), pp. 285–296.
- [313] YUAN YC & GAY G, 2006, *Homophily of network ties and bonding and bridging social capital in computer-mediated distributed teams*, Journal of Computer-Mediated Communication, **11**(4), pp. 1062–1084.
- [314] ZAVERI A, RULA A, MAURINO A, PIETROBON R, LEHMANN J & AUER S, 2015, *Quality assessment for linked data: A survey*, Semantic Web, **7**, pp. 63–93.
- [315] ZENG X, TU X, LIU Y, FU X & SU Y, 2022, *Toward better drug discovery with knowledge graph*, Current Opinion in Structural Biology, **72**, pp. 114–126.
- [316] ZHANG A, LIPTON ZC, LI M & SMOLA AJ, 2023, *Dive into deep learning*, Cambridge University Press, Cambridge.
- [317] ZHANG C, SONG D, HUANG C, SWAMI A & CHAWLA NV, 2019, *Heterogeneous graph neural network*, Proceedings of the 25<sup>th</sup> ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Anchorage (AK), pp. 793–803.
- [318] ZHANG F, LIU X, TANG J, DONG Y, YAO P, ZHANG J, GU X, WANG Y, SHAO B, LI R & WANG K, 2019, *OAG: Toward linking large-scale heterogeneous entity graphs*, Proceedings of the 25<sup>th</sup> ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage (AK), pp. 2585–2595.
- [319] ZHANG S, LIN X & ZHANG X, 2021, *Discovering DTI and DDI by knowledge graph with MHRW and improved neural network*, Proceedings of the 2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Virtual, pp. 588–593.
- [320] ZHANG Y, SHENG M, ZHOU R, WANG Y, HAN G, ZHANG H, XING C & DONG J, 2020, *HKGB: An inclusive, extensible, intelligent, semi-auto-constructed knowledge graph framework for healthcare with clinicians' expertise incorporated*, Information Processing & Management, **57**(6), pp. 1–16.
- [321] ZHANG Y, BAI R, CHEN Q, ZHANG Y & FENG M, 2022, *Causal discovery and knowledge linkage in scientific literature: A case study in biomedicine*, Proceedings of the Information for a Better World: Shaping the Global Future: 17<sup>th</sup> International Conference, Virtual, pp. 319–328.
- [322] ZHENG M, WANG F, HU X, MIAO Y, CAO H & TANG M, 2022, *A method for analyzing the performance impact of imbalanced binary data on machine learning models*, Axioms, **11**(11), pp. 1–19.
- [323] ZHENG S, RAO J, SONG Y, ZHANG J, XIAO X, FANG EF, YANG Y & NIU Z, 2021, *PharmKG: A dedicated knowledge graph benchmark for biomedical data mining*, Briefings in Bioinformatics, **22**, pp. 1–15.

- 
- [324] ZHOU D, BOUSQUET O, LAL T, WESTON J & SCHÖLKOPF B, 2003, *Learning with local and global consistency*, Proceedings of the 2003 Conference on Advances in Neural Information Processing Systems, Vancouver, pp. 1–8.
- [325] ZHOU J, WU F, ZHU X, CRASWELL N & SONG L, 2018, *Graph neural networks: A review of methods and applications*, IEEE Transactions on Knowledge and Data Engineering, **32(1)**, pp. 1–21.
- [326] ZHOU T, LÜ L & ZHANG Y.-C, 2009, *Predicting missing links via local information*, The European Physical Journal B, **71**, pp. 623–630.
- [327] ZHOU Y, WANG F, TANG J, NUSSINOV R & CHENG F, 2020, *Artificial intelligence in COVID-19 drug repurposing*, Lancet Digit Health, **2(12)**, pp. 667–676.
- [328] ZHU F & UKKUSURI SV, 2014, *Accounting for dynamic speed limit control in a stochastic traffic environment: A reinforcement learning approach*, Transportation Research Part C: Emerging Technologies, **41**, pp. 30–47.
- [329] ZHU X, GHAHRAMANI Z & LAFFERTY JD, 2003, *Semi-supervised learning using gaussian fields and harmonic functions*, Proceedings of the 20<sup>th</sup> International Conference on Machine Learning, Washington (DC), pp. 912–919.
- [330] ZIETSMAN H, 2021, *The data science process — From data preparation to model evaluation and deployment*, (Unpublished) Technical Report 2021-01, Stellenbosch Unit for Operations Research in Engineering, Department of Industrial Engineering, Stellenbosch University, Stellenbosch.





---



---

APPENDIX A

---

## Numerical results

In this appendix, detailed results pertaining to the instantiations conducted in Chapter 5 are presented. First, a detailed account of the hyperparameter tuning results obtained in respect of the 1K and 10K data sets are presented in Tables A.1–A.4. Furthermore, box plots corresponding to the three instantiations are presented in Figure A.1–A.6. Lastly, the  $p$ -values of the Nemenyi *post hoc* procedure are presented in respect of the first instantiation, as shown in Tables A.5–A.8.

TABLE A.1: *Hyperparameter tuning results for classifier-based algorithms with respect to the 1K data set of the first instantiation. The hyperparameter value combination yielding the largest AUPRC score is considered best and its corresponding AUROC and AUPRC scores are highlighted in bold.*

Algorithm	Hyperparameter combination	AUROC	AUPRC
LR	$C = 0.1, \ell_2$	0.9143	0.9293
	$C = 1, \ell_2$	<b>0.9143</b>	<b>0.9293</b>
	$C = 10, \ell_2$	0.9143	0.9293
NB	alpha = 0.2	0.8767	0.9002
	alpha = 0.4	0.8857	0.9073
	alpha = 0.6	<b>0.8913</b>	<b>0.9116</b>
$k$ NN	Number of neighbours = 4, uniform	0.9185	0.9013
	Number of neighbours = 4, distance	0.9234	0.9162
	Number of neighbours = 5, uniform	0.9203	0.9081
	Number of neighbours = 5, distance	0.9276	0.9250
	Number of neighbours = 6, uniform	0.9201	0.9109
	Number of neighbours = 6, distance	<b>0.9295</b>	<b>0.9292</b>
DT	gini, best, ccp_alpha = 0.1	0.8918	0.8167
	gini, random, ccp_alpha = 0.1	0.8130	0.7923
	entropy, best, ccp_alpha = 0.1	<b>0.9394</b>	<b>0.9139</b>
	entropy, random, ccp_alpha = 0.1	0.8758	0.8516
	log_loss, best, ccp_alpha = 0.1	0.9394	0.9139
	log_loss, random, ccp_alpha = 0.1	0.8545	0.8311
	gini, best, ccp_alpha = 0.2	0.8918	0.8167
	gini, random, ccp_alpha = 0.2	0.8130	0.7923
	entropy, best, ccp_alpha = 0.2	0.8929	0.8098
	entropy, random, ccp_alpha = 0.2	0.8387	0.8151
	log_loss, best, ccp_alpha = 0.2	0.8929	0.8098
	log_loss, random, ccp_alpha = 0.2	0.8274	0.7958
	gini, best, ccp_alpha = 0.3	0.8918	0.8167

TABLE A.1: Hyperparameter tuning results for classifier-based algorithms with respect to the 1K data set of the first instantiation. The hyperparameter value combination yielding the largest AUPRC score is considered best and its corresponding AUROC and AUPRC scores are highlighted in bold.

Algorithm	Hyperparameter combination	AUROC	AUPRC
RF	gini, random, ccp_alpha = 0.3	0.5793	0.5198
	entropy, best, ccp_alpha = 0.3	0.8929	0.8098
	entropy, random, ccp_alpha = 0.3	0.8130	0.7923
	log_loss, best, ccp_alpha = 0.3	0.8929	0.8098
	log_loss, random, ccp_alpha = 0.3	0.8130	0.7923
	Number of trees = 2, gini, ccp_alpha = 0.1	0.8928	0.8212
	Number of trees = 4, gini, ccp_alpha = 0.1	0.8990	0.8366
	Number of trees = 2, entropy, ccp_alpha = 0.1	0.9143	0.8600
	Number of trees = 4, entropy, ccp_alpha = 0.1	<b>0.9349</b>	<b>0.8978</b>
	Number of trees = 2, log_loss, ccp_alpha = 0.1	0.8944	0.8388
	Number of trees = 4, log_loss, ccp_alpha = 0.1	0.9242	0.892
	Number of trees = 2, gini, ccp_alpha = 0.2	0.8085	0.7349
	Number of trees = 4, gini, ccp_alpha = 0.2	0.9073	0.8437
	Number of trees = 2, entropy, ccp_alpha = 0.2	0.8081	0.7352
	Number of trees = 4, entropy, ccp_alpha = 0.2	0.8904	0.8144
	Number of trees = 2, log_loss, ccp_alpha = 0.2	0.8677	0.8014
	Number of trees = 4, log_loss, ccp_alpha = 0.2	0.8954	0.8209
	Number of trees = 2, gini, ccp_alpha = 0.3	0.5773	0.5179
	Number of trees = 4, gini, ccp_alpha = 0.3	0.7371	0.6713
	Number of trees = 2, entropy, ccp_alpha = 0.3	0.7345	0.6671
Number of trees = 4, entropy, ccp_alpha = 0.3	0.8901	0.8212	
Number of trees = 2, log_loss, ccp_alpha = 0.3	0.882	0.8077	
Number of trees = 4, log_loss, ccp_alpha = 0.3	0.9071	0.8395	
MLP	Size of hidden layer = (10), constant, max_iter = 200	0.8259	0.7819
	Size of hidden layer = (10), constant, max_iter = 300	0.8349	0.8184
	Size of hidden layer = (10), adaptive, max_iter = 200	0.7322	0.6546
	Size of hidden layer = (10), adaptive, max_iter = 300	0.8522	0.8396
	Size of hidden layer = (15, 10), constant, max_iter = 200	0.8030	0.7885
	Size of hidden layer = (15, 10), constant, max_iter = 300	0.7680	0.7354
	Size of hidden layer = (15, 10), adaptive, max_iter = 200	0.8357	0.8142
	Size of hidden layer = (15, 10), adaptive, max_iter = 300	0.8318	0.8223
	Size of hidden layer = (20, 15), constant, max_iter = 200	0.8200	0.8045
	Size of hidden layer = (20, 15), constant, max_iter = 300	<b>0.8390</b>	<b>0.8424</b>
	Size of hidden layer = (20, 15), adaptive, max_iter = 200	0.7638	0.7599
	Size of hidden layer = (20, 15), adaptive, max_iter = 300	0.7911	0.7535

TABLE A.2: Hyperparameter tuning results for classifier-based algorithms with respect to the 10K data set of the first instantiation. The hyperparameter value combination yielding the largest AUPRC score is considered best and its corresponding AUROC and AUPRC scores are highlighted in bold.

Algorithm	Hyperparameter combination	AUROC	AUPRC
LR	$C = 0.1, \ell_2$	0.9348	0.9480
	$C = 1, \ell_2$	<b>0.9348</b>	<b>0.9480</b>
	$C = 10, \ell_2$	0.9348	0.9480
NB	alpha = 0.2	0.9683	0.9670
	alpha = 0.4	<b>0.9684</b>	<b>0.9670</b>

TABLE A.2: Hyperparameter tuning results for classifier-based algorithms with respect to the 10K data set of the first instantiation. The hyperparameter value combination yielding the largest AUPRC score is considered best and its corresponding AUROC and AUPRC scores are highlighted in bold.

Algorithm	Hyperparameter combination	AUROC	AUPRC
<i>k</i> NN	alpha = 0.6	0.9684	0.9670
	Number of neighbours = 4, uniform	0.9492	0.9289
	Number of neighbours = 4, distance	0.9491	0.9354
	Number of neighbours = 5, uniform	0.9544	0.9388
	Number of neighbours = 5, distance	0.9543	0.9446
	Number of neighbours = 6, uniform	0.9578	0.9454
	Number of neighbours = 6, distance	<b>0.9576</b>	<b>0.9507</b>
DT	gini, best, ccp_alpha = 0.1	0.9121	0.8670
	gini, random, ccp_alpha = 0.1	0.8510	0.8345
	entropy, best, ccp_alpha = 0.1	0.9118	0.8671
	entropy, random, ccp_alpha = 0.1	<b>0.9209</b>	<b>0.9055</b>
	log_loss, best, ccp_alpha = 0.1	0.9118	0.8671
	log_loss, random, ccp_alpha = 0.1	0.8719	0.8541
	gini, best, ccp_alpha = 0.2	0.8510	0.8345
	gini, random, ccp_alpha = 0.2	0.9118	0.8671
	entropy, best, ccp_alpha = 0.2	0.8510	0.8345
	entropy, random, ccp_alpha = 0.2	0.9118	0.8151
	log_loss, best, ccp_alpha = 0.2	0.8929	0.8671
	log_loss, random, ccp_alpha = 0.2	0.8510	0.8345
	gini, best, ccp_alpha = 0.3	0.9121	0.8670
	gini, random, ccp_alpha = 0.3	0.5000	0.4444
	entropy, best, ccp_alpha = 0.3	0.9118	0.8671
	entropy, random, ccp_alpha = 0.3	0.8510	0.8345
	log_loss, best, ccp_alpha = 0.3	0.9118	0.8671
	log_loss, random, ccp_alpha = 0.3	0.8510	0.8345
	RF	Number of trees = 2, gini, ccp_alpha = 0.1	0.8813
Number of trees = 4, gini, ccp_alpha = 0.1		0.9140	0.8603
Number of trees = 2, entropy, ccp_alpha = 0.1		0.9176	0.8855
Number of trees = 4, entropy, ccp_alpha = 0.1		<b>0.9383</b>	<b>0.9066</b>
Number of trees = 2, log_loss, ccp_alpha = 0.1		0.9086	0.8572
Number of trees = 4, log_loss, ccp_alpha = 0.1		0.9189	0.8730
Number of trees = 2, gini, ccp_alpha = 0.2		0.8156	0.7555
Number of trees = 4, gini, ccp_alpha = 0.2		0.9136	0.8579
Number of trees = 2, entropy, ccp_alpha = 0.2		0.8868	0.8132
Number of trees = 4, entropy, ccp_alpha = 0.2		0.9181	0.8625
Number of trees = 2, log_loss, ccp_alpha = 0.2		0.8852	0.8172
Number of trees = 4, log_loss, ccp_alpha = 0.2		0.9056	0.8421
Number of trees = 2, gini, ccp_alpha = 0.3		0.5822	0.5286
Number of trees = 4, gini, ccp_alpha = 0.3		0.7470	0.6975
Number of trees = 2, entropy, ccp_alpha = 0.3		0.8777	0.8030
Number of trees = 4, entropy, ccp_alpha = 0.3		0.8943	0.8262
Number of trees = 2, log_loss, ccp_alpha = 0.3		0.8883	0.8208
Number of trees = 4, log_loss, ccp_alpha = 0.3	0.8122	0.7495	
MLP	Size of hidden layer = (10), constant, max_iter = 200	<b>0.8925</b>	<b>0.8785</b>
	Size of hidden layer = (10), constant, max_iter = 300	0.8437	0.8468

TABLE A.2: Hyperparameter tuning results for classifier-based algorithms with respect to the 10K data set of the first instantiation. The hyperparameter value combination yielding the largest AUPRC score is considered best and its corresponding AUROC and AUPRC scores are highlighted in bold.

Algorithm	Hyperparameter combination	AUROC	AUPRC
	Size of hidden layer = (10), adaptive, max_iter = 200	0.9168	0.8666
	Size of hidden layer = (10), adaptive, max_iter = 300	0.8960	0.8267
	Size of hidden layer = (15, 10), constant, max_iter = 200	0.8761	0.8367
	Size of hidden layer = (15, 10), constant, max_iter = 300	0.8635	0.8498
	Size of hidden layer = (15, 10), adaptive, max_iter = 200	0.8522	0.8351
	Size of hidden layer = (15, 10), adaptive, max_iter = 300	0.8511	0.8345
	Size of hidden layer = (20, 15), constant, max_iter = 200	0.8703	0.8578
	Size of hidden layer = (20, 15), constant, max_iter = 300	0.8899	0.8713
	Size of hidden layer = (20, 15), adaptive, max_iter = 200	0.8377	0.8054
	Size of hidden layer = (20, 15), adaptive, max_iter = 300	0.8353	0.8346

TABLE A.3: Hyperparameter tuning results for the GNN architectures with respect to the 1K data set of the first instantiation. The hyperparameter value combination yielding the largest AUPRC score is considered best and its corresponding AUROC and AUPRC scores are highlighted in bold.

Algorithm	Hyperparameter combination	AUROC	AUPR
GraphSAGE	Layers = 2, Epochs = 40, Aggregation = max	0.8616	0.7761
	Layers = 3, Epochs = 40, Aggregation = max	<b>0.8733</b>	<b>0.7957</b>
	Layers = 4, Epochs = 50, Aggregation = mean	0.8610	0.7686
	Layers = 2, Epochs = 60, Aggregation = mean	0.8676	0.7801
	Layers = 2, Epochs = 40, Aggregation = mean	0.8729	0.7955
	Layers = 2, Epochs = 60, Aggregation = max	0.8627	0.7693
GATConv	Layers = 2, Epochs = 60, Heads = 1	0.7636	0.6633
	Layers = 3, Epochs = 50, Heads = 1	0.7955	0.6310
	Layers = 4, Epochs = 50, Heads = 2	<b>0.8583</b>	<b>0.7663</b>
	Layers = 3, Epochs = 60, Heads = 1	0.7673	0.5896
	Layers = 2, Epochs = 40, Heads = 2	0.8114	0.7051
	Layers = 4, Epochs = 50, Heads = 1	0.8346	0.7329
GATv2Conv	Layers = 2, Epochs = 60, Heads = 1	0.8142	0.7038
	Layers = 3, Epochs = 50, Heads = 1	0.8314	0.6868
	Layers = 4, Epochs = 50, Heads = 2	0.8559	0.7560
	Layers = 3, Epochs = 60, Heads = 1	0.8330	0.7129
	Layers = 2, Epochs = 40, Heads = 2	0.8298	0.7284
	Layers = 4, Epochs = 50, Heads = 1	<b>0.8430</b>	<b>0.7711</b>
Graph transformer	Layers = 2, Epochs = 60, Heads = 1	0.8653	0.7802
	Layers = 3, Epochs = 50, Heads = 1	0.8683	0.7878
	Layers = 4, Epochs = 50, Heads = 2	0.8729	0.7959
	Layers = 3, Epochs = 60, Heads = 1	0.8685	0.7813
	Layers = 2, Epochs = 40, Heads = 2	0.8773	0.8011
	Layers = 4, Epochs = 50, Heads = 1	<b>0.8739</b>	<b>0.8066</b>

TABLE A.4: Hyperparameter tuning results for the GNN architectures with respect to the 10K data set of the first instantiation. The hyperparameter value combination yielding the largest AUPRC score is considered best and its corresponding AUROC and AUPRC scores are highlighted in bold.

Algorithm	Hyperparameter combination	AUROC	AUPR
GraphSAGE	Layers = 2, Epochs = 40, Aggregation = max	0.9332	0.8740
	Layers = 3, Epochs = 40, Aggregation = max	<b>0.9319</b>	<b>0.8811</b>
	Layers = 4, Epochs = 50, Aggregation = mean	0.9029	0.8443
	Layers = 2, Epochs = 60, Aggregation = mean	0.9046	0.8413
	Layers = 2, Epochs = 40, Aggregation = mean	0.9138	0.8539
	Layers = 2, Epochs = 60, Aggregation = max	0.9134	0.8492
GATConv	Layers = 2, Epochs = 60, Heads = 1	0.9405	0.8825
	Layers = 3, Epochs = 50, Heads = 1	0.9170	0.8391
	Layers = 4, Epochs = 50, Heads = 2	<b>0.9521</b>	<b>0.9191</b>
	Layers = 3, Epochs = 60, Heads = 1	0.9103	0.8240
	Layers = 2, Epochs = 40, Heads = 2	0.9429	0.8891
	Layers = 4, Epochs = 50, Heads = 1	0.9413	0.8908
GATv2Conv	Layers = 2, Epochs = 60, Heads = 1	0.9422	0.8894
	Layers = 3, Epochs = 50, Heads = 1	0.9334	0.8725
	Layers = 4, Epochs = 50, Heads = 2	<b>0.9538</b>	<b>0.9204</b>
	Layers = 3, Epochs = 60, Heads = 1	0.9406	0.8870
	Layers = 2, Epochs = 40, Heads = 2	0.9477	0.8987
	Layers = 4, Epochs = 50, Heads = 1	0.9423	0.8878
Graph transformer	Layers = 2, Epochs = 60, Heads = 1	0.9175	0.8627
	Layers = 3, Epochs = 50, Heads = 1	0.9131	0.8596
	Layers = 4, Epochs = 50, Heads = 2	0.9144	0.8664
	Layers = 3, Epochs = 60, Heads = 1	0.9043	0.8403
	Layers = 2, Epochs = 40, Heads = 2	<b>0.9291</b>	<b>0.8847</b>
	Layers = 4, Epochs = 50, Heads = 1	0.9083	0.8528

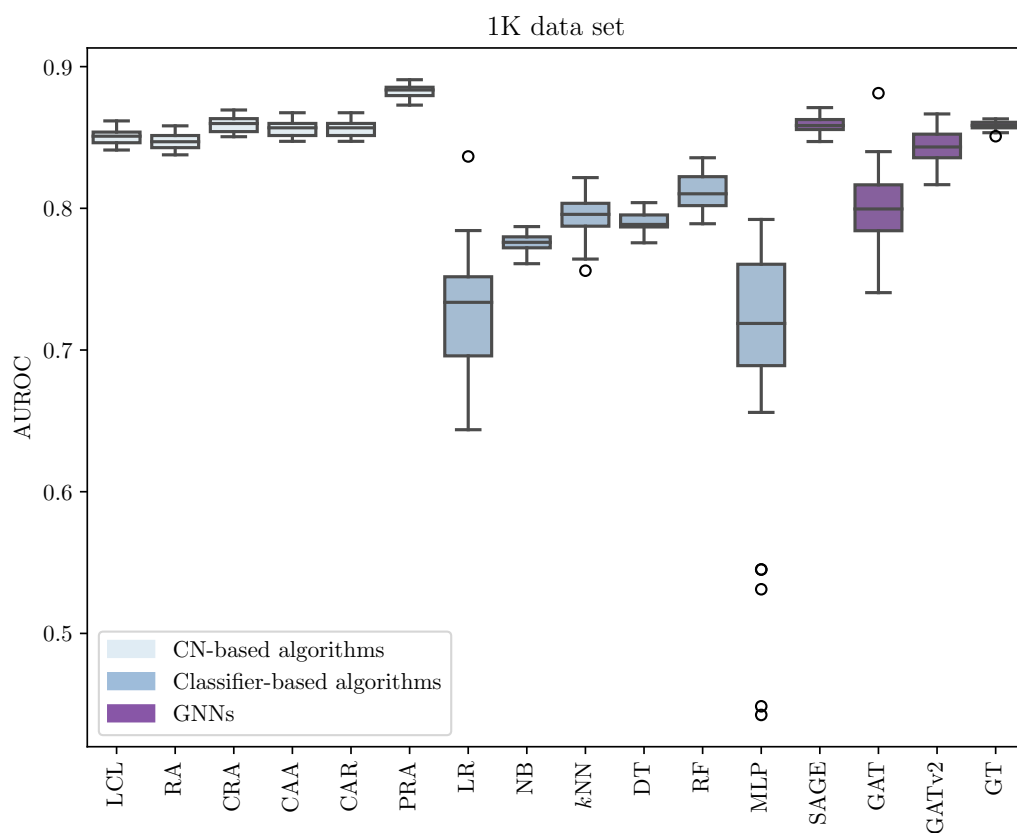


FIGURE A.1: The AUROC performance achieved by the different algorithms in respect of the 1K testing data set for the first instantiation.

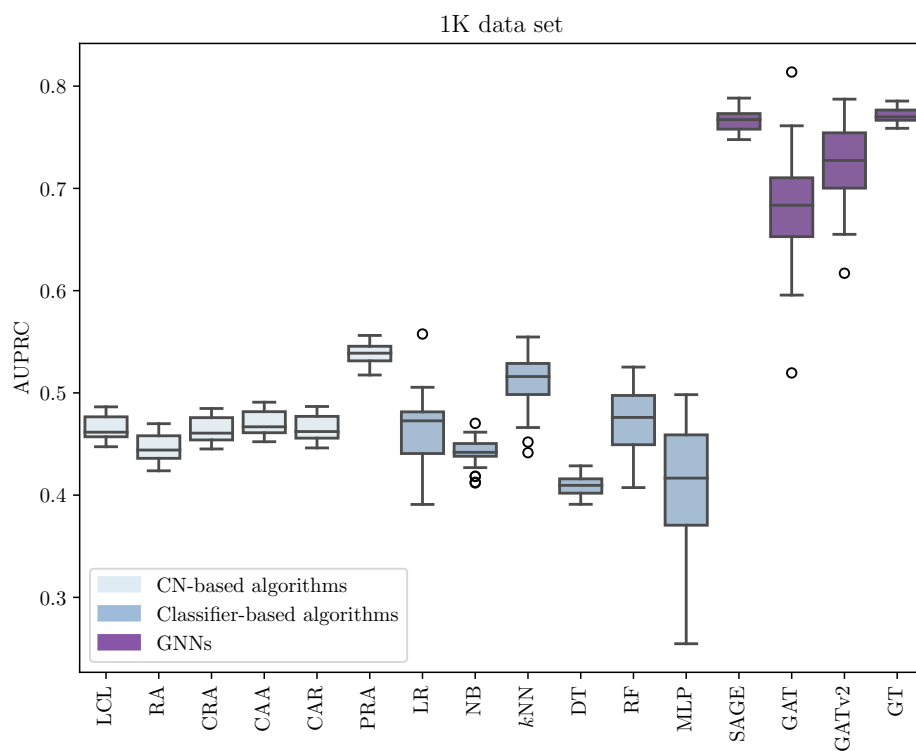


FIGURE A.2: The AUPRC performance achieved by the different algorithms in respect of the 1K testing data set for the first instantiation.



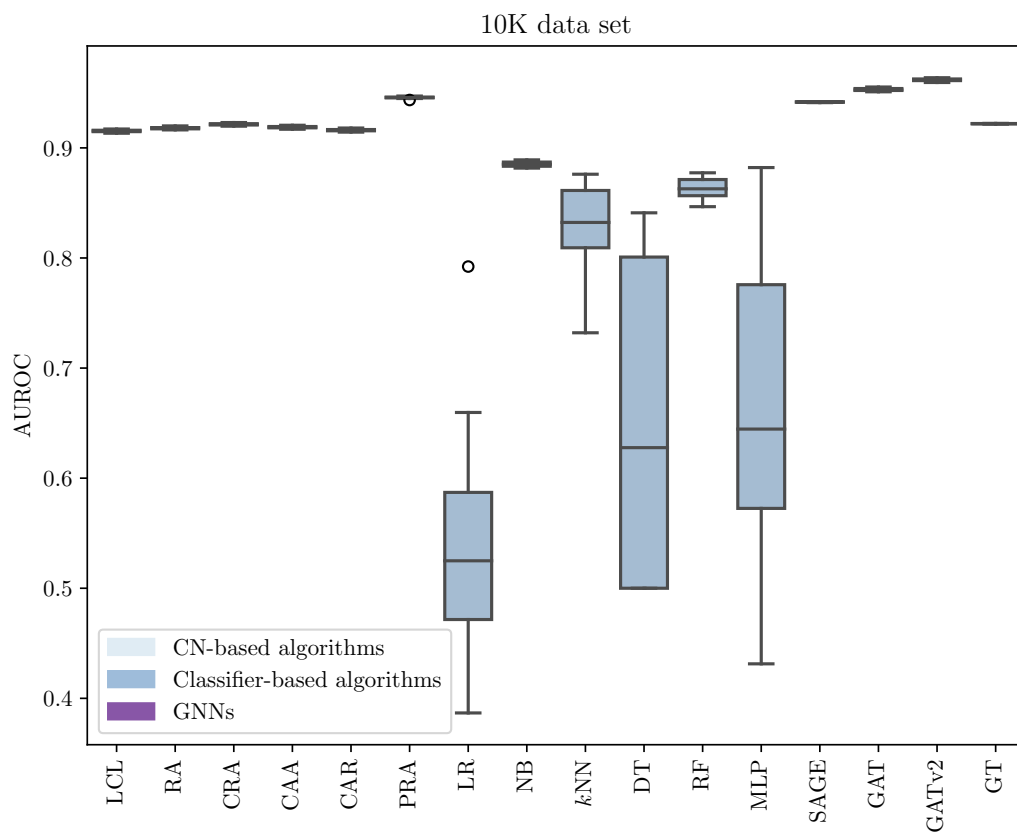


FIGURE A.3: The AUROC performance achieved by the different algorithms in respect of the 10K testing data set for the first instantiation.

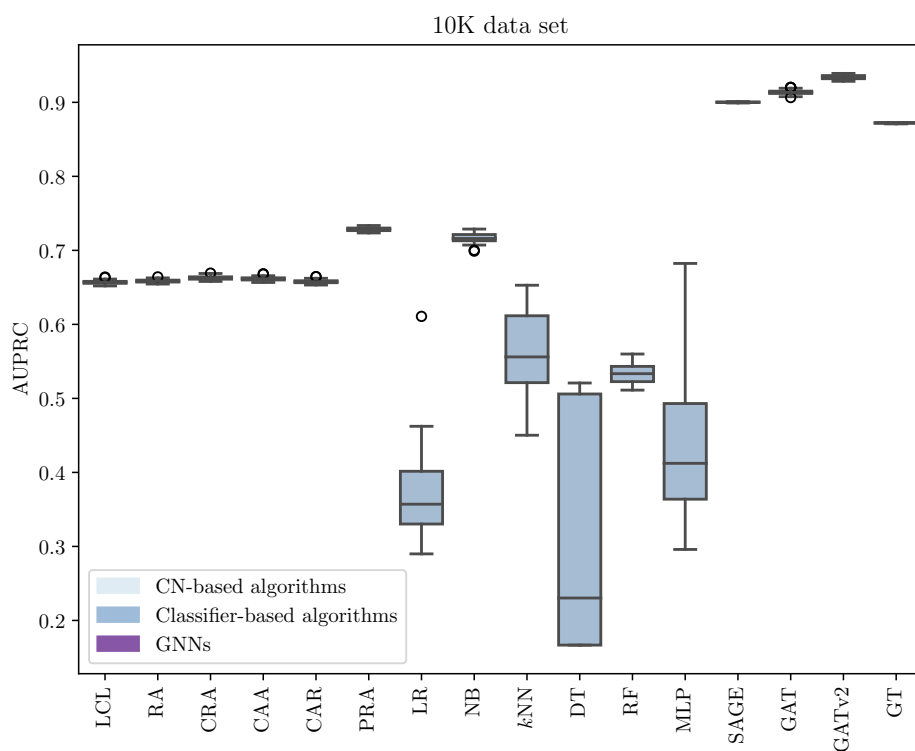


FIGURE A.4: The AUPRC performance achieved by the different algorithms in respect of the 10K testing data set for the first instantiation.

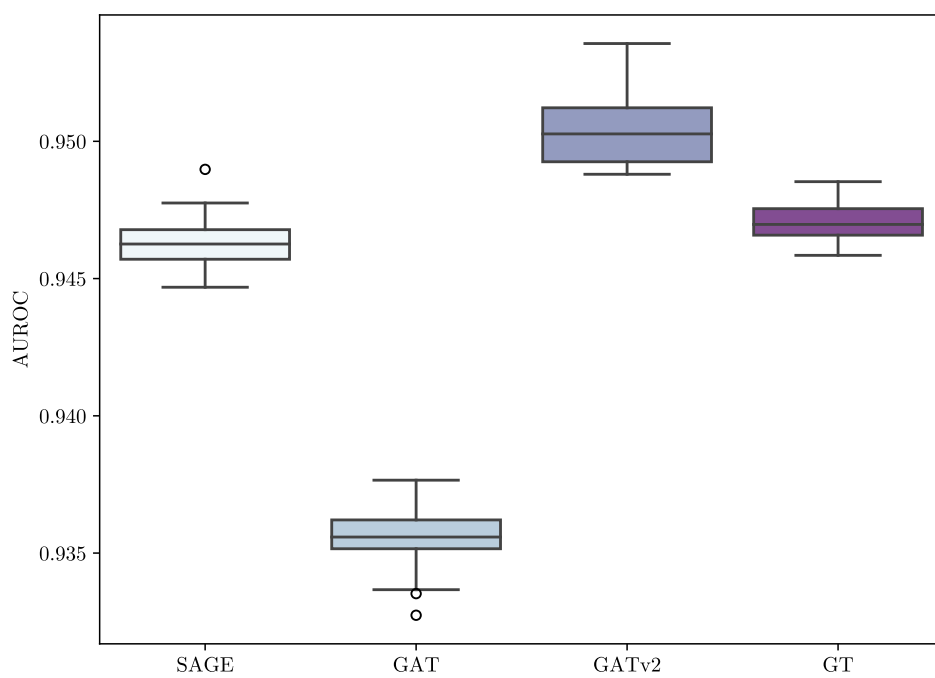


FIGURE A.5: The AUROC performance achieved by the GNNs in respect of the second instantiation.

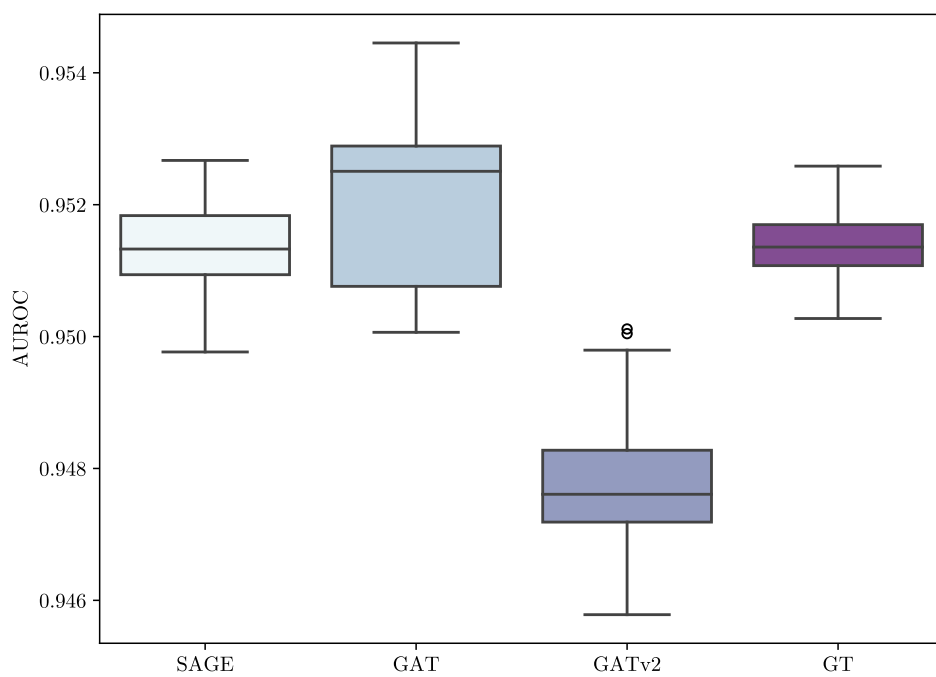


FIGURE A.6: *The AUROC performance achieved by the GNNs in respect of the third instantiation.*







