Gottfried Wilhelm Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik

EVALUATING HYBRID AI FOR PREDICTION OVER LUNG CANCER KNOWLEDGE GRAPHS

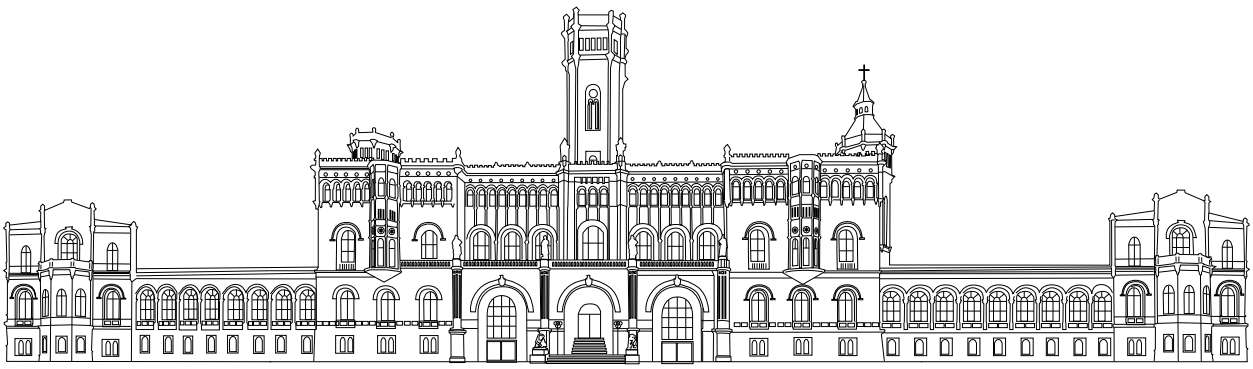
A thesis submitted in fulfillment of the requirements for the degree of
Master of Computer Science

BY

Sahar Safaei
Matriculation number: 10035219
E-mail: sahar.safaei@stud.uni-hannover.de

First evaluator: Prof. Dr. Maria-Esther Vidal
Second evaluator: Prof. Dr. Sören Auer
Supervisor: M.Sc. Emetis Niazmand

January 29, 2024



Declaration of Authorship

I, Sahar Safaei, declare that this thesis titled, 'EVALUATING HYBRID AI FOR PREDICTION OVER LUNG CANCER KNOWLEDGE GRAPHS ' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Sahar Safaei

Signature: _____

Date: _____

Acknowledgements

I would like to express my sincere gratitude to Prof. Dr. Maria-Esther Vidal for her invaluable guidance, unwavering support and insightful feedback throughout the completion of this Master's thesis. Her expertise and encouragement have been instrumental in shaping the direction of my research and enhancing the overall quality of this thesis.

I am also deeply thankful to my supervisor, M.Sc. Emetis Niazmand, for her guidance and constructive criticism. She has been crucial in navigating the challenges of the research process and refining the content of this thesis. I appreciate the time and effort she invested in helping me improve my research skills.

I am grateful to the entire academic community at TIB-Leibniz Information Center for Sciences and Technology University Library at Leibniz University Hannover for providing a conducive environment for learning and research. The resources and facilities at the institute have been indispensable in carrying out this study.

Finally, I would like to thank my family and my beloved husband for their unwavering support and encouragement throughout this academic journey. Their understanding and encouragement have been a source of strength, and I am truly appreciative of their role in my success.

Abstract

Link prediction is of great importance in the field of knowledge graphs, as it plays a key role in facilitating knowledge discovery and supporting decision-making, especially in healthcare. Although knowledge graphs provide a structured representation of data, challenges arise from data integration and quality assurance issues. The presence of inaccuracies, outdated information and inconsistencies poses a threat to data quality, requiring ongoing efforts to address incomplete or missing data. The challenges posed by data quality issues are multifaceted and contribute to an overall reduction in the reliability of information. In the era of big data and artificial intelligence, dealing with incomplete information and missing data is a challenge. Inductive learning, a form of machine learning that involves making generalizations based on specific examples, can be a valuable approach for link prediction to overcome some obstacles associated with knowledge graphs in healthcare. In response to these challenges, link prediction is becoming as a valuable technique to improve the quality of knowledge graphs by filling in missing links. The state-of-the-art proposes various approaches for knowledge graph completion and link predictions involves the evaluation of different embeddings and symbolic learning models. Experimental benchmarks are designed to evaluate different models and relations types and provide insights into their effectiveness. This research aims to develop a framework for evaluation of hybrid AI models over lung cancer knowledge graph. The primary objectives include comparative analysis of embeddings and symbolic learning models, investigation of the impact of data modelling, exploration of the influence of relation types, and evaluation of the impact of knowledge graph enhancing.

Contents

1 Introduction **1**
- 1.1 Overview of the Document . . . 1
- 1.2 Motivating Example . . . 3
- 1.3 Our Proposed Approach . . . 6
- 1.4 Contribution . . . 7
- 1.5 Structure of the Document . . . 8
- 1.6 Summary of the Chapter . . . 9

2 Background **10**
- 2.1 Preliminaries of Knowledge Graphs . . . 10
  - 2.1.1 RDF . . . 11
  - 2.1.2 OWL . . . 11
  - 2.1.3 Data Modeling . . . 11
  - 2.1.4 SPARQL . . . 12
  - 2.1.5 Data Integration . . . 12
  - 2.1.6 Mapping Languages . . . 13
  - 2.1.7 Closed and Open World Assumptions . . . 13
- 2.2 A Lung Cancer Knowledge Graph . . . 14
- 2.3 Inductive Knowledge . . . 15
  - 2.3.1 Terminology . . . 15
- 2.4 Models for Knowledge Graph Embeddings . . . 19
  - 2.4.1 Translational Models . . . 21
  - 2.4.2 Tensor Decomposition Models . . . 24
  - 2.4.3 Neural Models . . . 27
- 2.5 Symbolic Learning . . . 28
  - 2.5.1 Rule Mining . . . 28
- 2.6 Link Prediction . . . 31
  - 2.6.1 Evaluation Metrics for Link Prediction . . . 32

2.7 Summary of the Chapter 33

3 Related Work **34**
3.1 Knowledge Graph Completion Methods 34
3.1.1 Impact of Data Redundancy 35
3.1.2 Impact of Cartesian Relations 35
3.2 Summary of the Chapter 35

4 Our Approach and Proposed Solution **36**
4.1 Problem Statement 36
4.2 Proposed Approach 37
4.2.1 Methodology 38
4.3 Summary of the Chapter 51

5 Experimental Evaluation **52**
5.1 Experimental Setup 52
5.2 Benchmarks 54
5.3 PreProcessing Steps 56
5.4 Evaluation Metrics 59
5.5 Results 59
5.5.1 Analyze Different Models 61
5.5.2 Analyze Performance of Different Models by Enhancing Data . 62
5.5.3 Analyze Performance with Different Data Schemas 64
5.5.4 Analyze Different Type of Relations 65
5.6 Summary of the Chapter 68

6 Conclusions and Future Work **69**
6.1 Conclusions 69
6.2 Limitations 70
6.3 Future Work 71
6.4 Summary of the Chapter 71

Bibliography **72**

List of Figures

1.1 **Motivating example.** The figure depicts 3 patients in the lung cancer KG. `Patient1` and `Patient2` are existing patients with their stage of diagnosis. `Patient3` is a new patient for whom our aim is to diagnose the stage based on his characteristics. Stage III is predicated on `Data Schema A` and Stage II in the `Data Schema B`, which `Data Schema B` predicate correctly. 4

2.1 Conceptual overview of popular inductive techniques for KGs in terms of type of representation generated (Numeric/Symbolic) and type of paradigm used (Unsupervised/Self-supervised/Supervised) 16
2.2 Example of mediator (Compound Value Type (CVT)) nodes in lung cancer KG 19
2.3 Simple illustration of TransE that connect s (subject) and o (object) with relation (p) 22
2.4 Simple illustration of TransH that connect a subject (s) and object (o with relation (r_p) 22
2.5 Simple illustration of TransR that connect s (subject) and o (object) with relation (r) 23
2.6 Simple illustration of TransD that connect s (subject) and o (object) with relation(r) 24
2.7 Visualization of the TuckER architecture.s. It is taken from [3] 26
2.8 Prediction under incompleteness [15] 29

4.1 The illustration of our approach that defines as a Black Box to evaluate hybrid AI for prediction over lung cancer KG 38
4.2 The workflow visualization of the approach. The workflow uses two different arrow colors. Orange arrows signify the workflow of base data, while blue arrows represent the workflow of preprocessed data with new data schema. 39
4.3 The workflow visualization of data preprocessing 40
4.4 The workflow visualization of prepare data for different framework 41
4.5 The workflow visualization of AMIE (Rule Mining) process 42
4.6 The workflow visualization of AMIE-LC process 44

4.7 The workflow visualization of TuckER process showing how the TuckER model is executed through our benchmarks 45
4.8 The workflow visualization of ConvE process showing how the ConvE model is executed through our benchmarks 46
4.9 The workflow visualization of OpenKE process showing how we prepare OpenKE framework to run other models through our benchmarks 47
4.10 The workflow visualization of TransE process showing how the TransE model is executed through our benchmarks 47
4.11 The workflow visualization of TransH process showing how the TransH model is executed through our benchmarks 48
4.12 The workflow visualization of TransR process showing how the TransR model is executed through our benchmarks 49
4.13 The workflow visualization of TransD process showing how the TransD model is executed through our benchmarks 49
4.14 The workflow visualization of RotatE process showing how the RotatE model is executed through our benchmarks 50
4.15 The workflow visualization of DistMult process showing how the DistMult model is executed through our benchmarks 50
4.16 The workflow visualization of ComplEx process showing how the ComplEx model is executed through our benchmarks 51

5.1 Preprocessing portion to change data schema in KG 55
5.2 Example of duplicates 56
5.3 Example of mediator (CVT) node in lung cancer KG 57
5.4 Illustration of the removal of the CVT node in the lung cancer KG and the establishment of connections between two entities, namely familyType and has familyCancerType, aimed at introducing more informative relations. . . 57
5.5 Evaluation models based on Hit@k(↑) metrics over BaseBiomarker. Better performance is indicated by a higher Hit@k value. 60
5.6 Evaluation models based on Hit@k(↑) metrics over Biomarker. Better performance is indicated by a higher Hit@k value. 61
5.7 Evaluation models based on Hit@k(↑) metrics over BaseRelapse. Better performance is indicated by a higher Hit@k value. 62
5.8 Evaluation models based on Hit@k(↑) metrics over Relapse. Better performance is indicated by a higher Hit@k value. 62
5.9 Compare MR(↓) between BaseBiomarker and BaseRelapse. Better performance is indicated by a lower MR value. 63
5.10 Compare MR(↓) between Biomarker and Relapse. Better performance is indicated by a lower MR value. 63
5.11 Compare MRR(↑) by enhancing data in BaseBiomarker and Biomarker. Better performance is indicated by a higher MRR value. 64

5.12 Evaluation models based on MRR(↑) metrics over Biomarker by modifying data schema. Better performance is indicated by a higher MRR value. . . . 64
5.13 Evaluation models based on MRR(↑) metrics over Relapse by modifying data schema. Better performance is indicated by a higher MRR value. . . . 65
5.14 Number of relation types in different Benchmarks 65
5.15 Evaluation MRR(↑) over Biomarker with different relation types. Better performance is indicated by a higher MRR value. 66
5.16 Evaluation MRR(↑) over Relapse with different relation types. Better performance is indicated by a higher MRR value. 66

List of Tables

1.1 Embeddings for `Data Schema A` 5
1.2 Embeddings for `Data Schema B` 5
1.3 Computed scores for triples `(Patient3, diagnosisStage, II)` and `(Patient3, diagnosisStage, III)` 6

5.1 Hyperparameters for OpenKE 53
5.2 Hyperparameters for models used OpenKE 53
5.3 Hyperparameters for ConvE 53
5.4 Hyperparameters for TuckER 54
5.5 Statistics of evaluation datasets 54
5.6 Establishment of relations used to create new data schema in lung cancer KG 58
5.7 Right_Link prediction result on BaseBiomarker (BB) and Biomarker (B). Better performance is indicated by a lower MR value, along with higher values for MRR and Hits@k. 59
5.8 Right_Link prediction result on BaseRelapse (BR) and Relapse (R). Better performance is indicated by a lower MR value, along with higher values for MRR and Hits@k. 60
5.9 Right_Link prediction result on BaseBiomarker over different relations types. Better performance is indicated by a lower MR value, along with higher values for MRR and Hits@k. 67
5.10 Right_Link prediction result on Biomarker over different relations types. Better performance is indicated by a lower MR value, along with higher values for MRR and Hits@k. 67
5.11 Right_Link prediction result on BaseRelapse over different relations types. Better performance is indicated by a lower MR value, along with higher values for MRR and Hits@k. 68
5.12 Right_Link prediction result on Relapse over different relations types. Better performance is indicated by a lower MR value, along with higher values for MRR and Hits@k. 68

Acronyms

AI Artificial Intelligence

CVT Compound Value Type

CWA Closed World Assumption

DIS Data Integration System

ILP Inductive Logic Programming

JSON-LD JSON for Linking Data

KB Knowledge Base

KG Knowledge Graph

OWA Open World Assumption

OWL Web Ontology Language

R2RML RDB to RDF Mapping Language

RDF Resource Description Framework

RML RDF Mapping Language

Chapter 1

Introduction

1.1 Overview of the Document

Knowledge Graph (KG)s are critical to many enterprises today: They provide the structured data and factual knowledge that drive many products and make them more intelligent. In general, a KG is a structured representation of knowledge that encompasses entities, their attributes, and the relationships between them. It serves as a way to organize and connect information in a meaningful way, allowing more efficient and comprehensive knowledge retrieval [24].
KGs provide a powerful framework for organizing, integrating, and leveraging healthcare knowledge, ultimately contributing to improved patient care, research outcomes, and healthcare decision-making. For example, KGs can help healthcare professionals make informed decisions by providing a comprehensive understanding of patient histories, treatment options, and relevant medical research. This can improve diagnostic accuracy and treatment planning [30].
Although KGs offer numerous benefits in organizing and connecting information, there are challenges associated with their integration. Some common problems include the following.

- **Data Heterogeneity**: Information in healthcare often comes from diverse sources, such as electronic health records, medical literature, and different databases. Integrating heterogeneous data into a unified KG can be challenging due to variations in formats, terminologies, and data structures.

- **Data Quality and Accuracy**: Ensuring the quality and accuracy of the data is crucial for KGs. Inaccurate or incomplete information can lead to unreliable

insight and decisions. Cleaning and validating data from various sources can be time-consuming.

- **Semantic Compatibility**: Achieving semantic interoperability involves mapping and aligning concepts from different sources to a common ontology. Differences in terminologies and the lack of standardized data schema can hinder the seamless integration of data in a KG.

Inaccurate and inconsistent data reduces the quality of the data. Dealing with incomplete information and missing data is therefore an ongoing challenge in big data and Artificial Intelligence (AI) [13]. Inductive learning, a form of machine learning that involves making generalizations based on specific examples, can be a valuable approach for link prediction to overcome some obstacles associated with KGs in healthcare. Link prediction is a valuable technique in KGs that can improve overall quality by filling in missing links in KGs.
The state of the art [1] proposes different approaches for KG completion and link prediction based on inductive learning. Akrami et al. [1] show that inductive learning models (e.g., KG embeddings or symbolic learning) are affected by certain types of relation (1-to-1, 1-to-n, n-to-n and n-to-1). In addition, KG embedding models are very sensitive to the representation and population of data within the KGs. Similarly, in symbolic models, e.g., AMIE [15], existing relationships between entities affect the type of rules learned by the symbolic learning. However, it is not clear how the modeling of KGs affects the performance of the methods, even in the comparison of diverse schemas and different types of relations.
In our study, we investigate the applicability of features identified in existing benchmarks for link prediction in KGs for application in the medical domain, focusing on a KG related to lung cancer. In particular, we will evaluate inductive learning models over lung cancer KG. Furthermore, our research addresses the challenge of KG completion by incorporating symbolic learning. We aim to develop a framework for predicting links in a KG using horn rules extracted from symbolic learning. This approach is implemented by considering an open-world assumption in KG.
In summary, the main goal of our work is to evaluate various aspects of link prediction within a KG. First, our objective is to evaluate the efficiency of our proposed link prediction approach compared to numerical models within KG. Furthermore, we are trying to understand the impact of enhancing KG, particularly in the context of integrating a Biomarker KG with a Relapse KG, on the link prediction task. Another aspect of our research is exploring ways to enhance the performance of the link prediction task through modifications to the data schema within the KG. Finally, our objective is to investigate whether the models learned for the link prediction task

have different scores based on different types of relation within a KG.

1.2 Motivating Example

We explore a motivating example to provide a clear picture of the issues this work aims to resolve.
Cancer, a pervasive and devastating disease, claims millions of lives each year and represents a significant challenge to medical research. Among the various types of cancer, lung cancer is one of the leading causes of cancer-related deaths worldwide. The prognosis for patients with lung cancer is highly dependent on the stage of the disease at the time of diagnosis. Early detection significantly increases treatment success, while late diagnosis limits options and outlook.
Figure 1.1 shows two different data schemas for three patients in a lung cancer KG. In these schemas, `Patient1` and `Patient2` are existing patients whose diagnosis stage is available in KG. `Patient3` is a new patient for whom we aim to diagnose the stage. In Figure 1.1 we consider two different data schemas. One difference between the two data schemas is that `Data Schema A` represents the relations `hasFamilyHistory_ familyrelationDegree`, `hasFamilyHistory_ familyGender`, `hasFamilyHistory_ familyType`, and `hasFamilyHistory_ cancerType` individually and links them to the patients. In contrast, `Data Schema B` merges the relations between `hasFamilyHistory_ familyrelationDegree` and `hasFamilyHistory_ familyGender`, as well as hasFamilyHistory_ familyType and `hasFamilyHistory_ cancerType`.
There are certain cases where different instances are referred to by different names. For example, in `Data Schema A`, the terms `Medium` and `Moderate` represent the same smoking habit. In Figure 1.1, the instance `Medium` is linked to the instance `Moderate` through the `owl:sameAs` property (e.g., `(Medium, owl:sameAs, Moderate)`), indicating that they are the same. In `Data Schema A`, `Patient2` is connected to `Medium` using the `smokerType` property, while `Patient3` is connected to `Moderate` using the same property. This inconsistency in naming instances within the KG brings challenges for prediction tasks. Therefore, ensuring consistency among instances within KGs is important. In `Data Schema B`, we directly connect `Patient2` to `Moderate` using the `smokerType` property (e.g., `(Patient2, smokerType, Moderate)`).
Further differences between two data Schemas is using categorization of values instead of values. Effective characterization is also an essential aspect for KGs. In `data Schema B`, we use the category `Old` for patients over 50 years old. This categorization leads to be `Patient2` and `Patient3` in the same age group. However, in `Data Schema A`, just `Patient1` and `Patient3` share a similar age of 70, in `Data Schema B` making all three patients share a similar category of age.

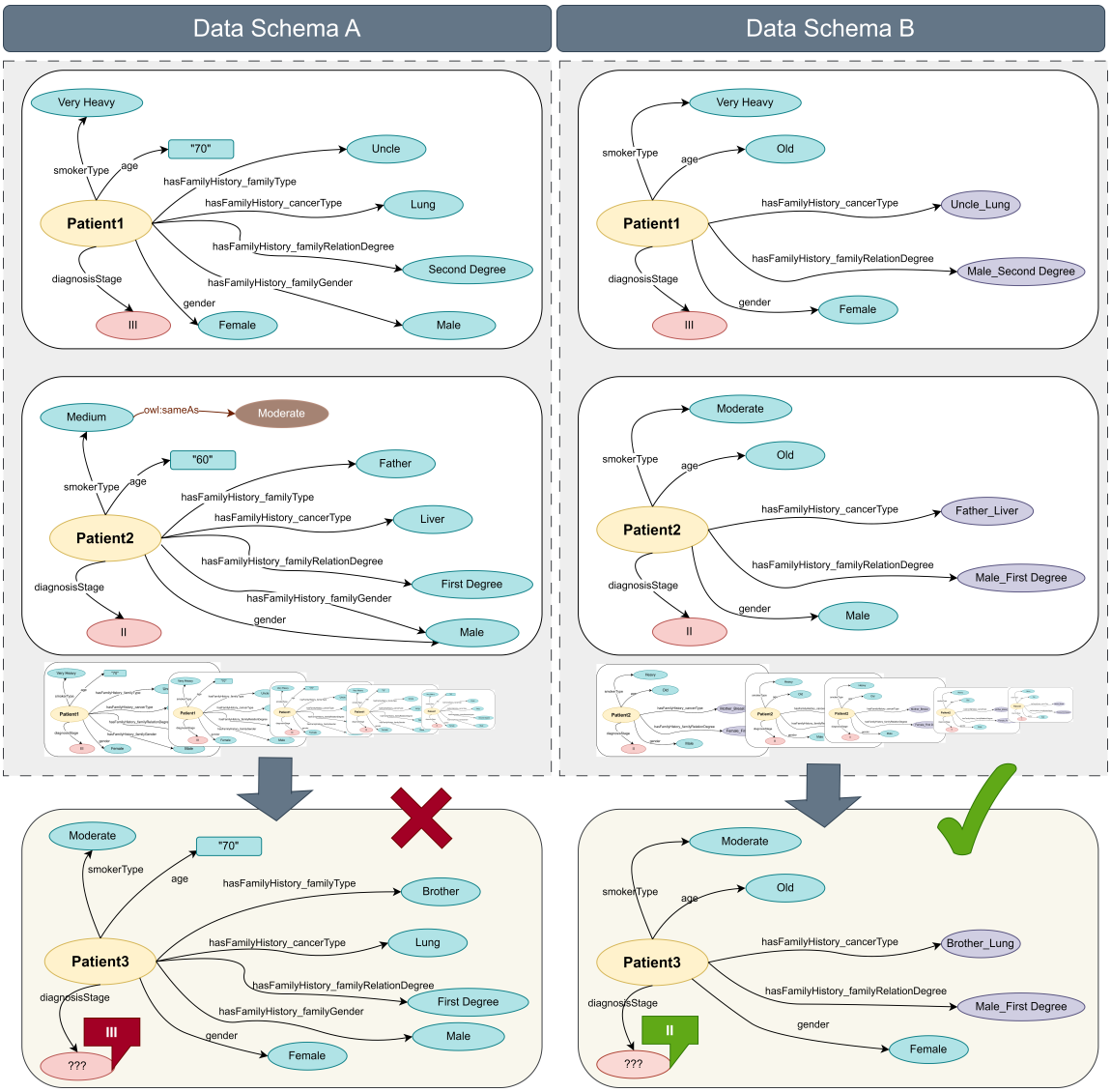


Figure 1.1: **Motivating example.** The figure depicts 3 patients in the lung cancer KG. `Patient1` and `Patient2` are existing patients with their stage of diagnosis. `Patient3` is a new patient for whom our aim is to diagnose the stage based on his characteristics. Stage III is predicated on `Data Schema A` and Stage II in the `Data Schema B`, which `Data Schema B` predicate correctly.

The main goal of the example shown in Figure 1.1 is to predict missing links buy discoverying the missing objects in the triplet `(Patient3, diagnosisStage,`

`?satge)`. Inductive learning uses AI techniques to make predictions about missing connections between instances within KG. Inductive learning can be broadly categorized into numerical and symbolic learning approaches, depending on the types of representations and methods used to generalize knowledge from data. One group is based on latent feature models, also known as knowledge graph embedding models. The other group is based on observed feature models that exploit observable properties of a KG, also known as symbolic models.
KG embedding models transform nodes and edges in a knowledge graph into a low dimensional continuous vector space that preserves KGs structure. In Tables 1.1 and 1.2, the embedding vector of entities and property required to predicate stage of `Patient3` is given. In Table 1.3, we compute the scores for two triples `(Patient3, diagnosisStage, II)` and `(Patient3, diagnosisStage, III)` in order to predicate the stage with higher score. The models resorts to the scoring function $\phi = -||e_s + r_p - e_o||_2$ for learning the vector representations of the nodes and edeges. The objective of a plausibility score function is to assign high values of ϕ the triples that are likely to be true and low scores to triples that are likely to be false.

Instances in Motivating Example KG		Embedding
Property	diagnosisStage	[0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1]
Entity	II	[0 1 0 0 1 0 1 0 0 1 0 1 2 0 2 1 1 2 1 2]
Entity	III	[1 0 0 1 0 1 0 0 1 0 1 0 1 1 1 2 2 1 1 2]
Entity	Patient3	[0 0 1 1 0 0 0 1 1 0 0 1 1 1 0 0 0 0 1 1]

Table 1.1: Embeddings for `Data Schema A`

Instances in Motivating Example KG		Embedding
Property	diagnosisStage	[0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]
Entity	II	[0 1 1 0 1 0 0 1 1 0 2 1 1 2 1 2]
Entity	III	[1 0 1 1 0 0 1 0 0 1 1 2 2 1 1 2]
Entity	Patient3	[0 1 1 0 0 1 0 1 0 1 0 0 0 0 1 1]

Table 1.2: Embeddings for `Data Schema B`

In `Data Schema A`, the prediction indicates stage III, while in `Data Schema B`, it suggests stage II, which `Data Schema B` predicate correctly. It is worth noting the discrepancy in the results obtained from the various data schema decisions. Merging leads to greater plausibility scores between `Patient2` and `Patient3` in `Data Schema`

B, highlighting the influence of data schema choices. In general, using the right data schemas, using the same instances, and using the right categorization can have a significant impact on the outcome of the prediction.

Triple	Data Schema A: Score	Data Schema B: Score
(Patient3, diagnosisStage, II)	-3.60	**-2.64**
(Patient3, diagnosisStage, III)	**-3.00**	-3.00

Table 1.3: Computed scores for triples (Patient3, diagnosisStage, II) and (Patient3, diagnosisStage, III)

In contrast to knowledge graph embedding models, which rely on latent features, symbolic models use directly observable features. For example, a symbolic model can derive a rule, such as (?patient, hasFamilyHistory_ cancerType, Lung) ⇒ (?patient, diagnosisStage, III) by noting that most lung cancer patients in a KG are diagnosed at stage III if they have a family member with lung cancer. This leads to the problem in Data Schema A, where stage III is assigned to the Patient3 without taking into account the specific type of relationship of the family member with lung cancer disease.
However, by changing the data Schema to Data Schema B, which combines attributes such as hasFamilyHistory_cancerType and hasFamilyHistory_familyType, the number of available patients in this combined relation (hasFamilyHistory_ cancerType) is reduced. As a result, the next rules, such as ?Patient, hasFamilyHistory_ cancerType, Father_ Liver) ⇒ (?Patient, diagnosisStage, II), contain more shares in KG than the rule (?Patient, hasFamilyHistory_ cancerType, Uncle_ Lung) ⇒ (?Patient, diagnosisStage, III). As a consequence, we predicate stage II for Patient3 in Data Schema B.
Moreover, categorizing values, such as age, makes the rule more general, enabling informed decisions based on categories rather than specific values. Additionally, using consistent values enhances the effectiveness of the rule. As illustrated in Figure 1.1, the inclusion of a moderate level of smoking for certain patients and a medium level for others results in a more intricate rule. This is because the rule is divided into these two categories, even though they both represent the same concept.

1.3 Our Proposed Approach

This work tackles the challenge of KG completion by hybrid AI approach. These approaches are divided into two groups. One group is based on latent feature mod-

els, also known as KG embedding models or numerical models, including TransE [6], ConvE [10], and many other methods explained in the book [14].
The other group is symbolic models based on observed feature models that exploit observable properties of a KG; examples of such methods include rule mining systems [15] and path ranking algorithms [18]. In contrast to symbolic models, numerical models have been extensively studied for link prediction. Therefore, in this research, we have also proposed an approach (AMIE-LC) to evaluate the link prediction task over lung cancer KG based on the rules extracted from the symbolic model. In this research, we use AMIE [15] for rule extraction.
We compare our results using benchmarks created from a subset of a lung cancerKG. In this study, we use two sections of the lung cancer KG. One part is extended to incorporate additional knowledge to assess the impact of data enrichment. Furthermore, our experiment involves the use of both baseline and preprocessed versions of lung cancer KG to analyze the effect of data schema in a KG completion.

1.4 Contribution

Our research is structured into four main components. First, we explore the comparison between embedding and symbolic models. Second, we investigate the impact of enhancing the KG by integrating biomarker data with relapse data. Thirdly, we evaluate the impact of data schema on the KG. Lastly, we evaluate the effectiveness of data schema and methods across various types of relations.

- **Comparison Between Embedding and Symbolic Models**: In the first part of our research, we find that symbolic models deliver superior performance at all KG benchmarks. When comparing between KG embedding models, it becomes evident that RotatE fails to make accurate predictions within KGs, whereas other translational models (TransE, TransH, TransR, and TransD) outperform tensor decomposition models (Distmult, ComplEx, and TuckER) and neural models (ConvE) (as illustrated in Figures 5.5, 5.6, 5.7, 5.8).

- **Impact of Enhancing KG**: In the second part of our research, we discover several effects of increasing KG by integrating the Biomarker KG with the Relapse KG. The implementation of enhanced knowledge leads to an improvement in prediction performance when we change the data Schema within the KG by preprocessing the data (as shown in Figure 5.10). However, when we

use the original benchmarks, we observe a decrease in prediction performance after data enrichment (as illustrated in Figure 5.9).

- **Impact of Data Schema**: In the third part of our research, we aim to explore the impact of data schema on our results. Figures 5.12 and 5.13 illustrate how data schema can affect the performance of predictions. This underscores the importance of investing in effective data schema strategy before performing any AI task on KGs.

- **Evaluation on Different Types of Relations**: In the last part of our research, the results in Figures 5.15 and 5.16 show that the performance differs according to the type of relation. With the implementation of new data schema, we observe an enhancement in the performance of `n-to-1` relations, while the performance in `n-to-n` relation types decreases with a new data schema. However, due to the exclusion of `1-to-1` relation types in the new data schema, the overall performance is enhanced.

1.5 Structure of the Document

This document consists of six chapters.

- Chapter 1 provides an overview of the whole document. We provide the reader with a motivating example, a list of the major contributions, and a description of the problem that is being addressed in this work.

- Chapter 2 presents the key concepts required for the understanding of the problems tackled in this work. It covers the background, terminology, and the fundamental principles required to understand the subsequent chapters. Chapter 2 is followed by an exploration of inductive knowledge and its various classifications. Furthermore, we investigate various KGs embedding models and symbolic learning. Chapter 2 comes to a close by delving into the subject of link prediction and the evaluation metrics that are applied to assess the studied models within this context.

- Chapter 3 summarizes related works and provides an overview of the state of the art while referring to relevant publications, and position the thesis concerning them.

- The approach and implementation of the approach are defined in Chapter 4. First, we discuss the approach, the formal problem definition, and the proposed

solution for AMIE-LC. We evaluate our approach, providing information on the development methodology used to construct our framework and detailing its design. We then discuss the experimental study of this thesis, which is dedicated to evaluating the performance of a hybrid AI system in the context of link prediction tasks using lung cancer KGs. We outline the experimental setup, explain the data collection and preprocessing steps, and introduce the evaluation metrics.

- Chapter 5 reports on the results of the evaluation of inductive models in various lung cancer KGs.
- Lastly, Chapter 6 summarizes our work and outlines recommendations for future work.

1.6 Summary of the Chapter

This chapter introduces a hybrid evaluation AI for the prediction over KGs. For a better understanding of the problem, the chapter also contains a motivating example. Furthermore, this chapter describes the contribution of the present work and concludes with an overview of the structure of the work.

Chapter 2

Background

This chapter introduces the essential concepts for understanding the problem of the thesis. It defines KGs, inductive knowledge, and different models used for KG embedding, and explores the foundations of symbolic learning. The chapter concludes by discussing the problem of link prediction and the evaluation metrics used to evaluate the models used in this context.

2.1 Preliminaries of Knowledge Graphs

A KG is a data structure that captures the relationships between different entities and concepts. Using a graph-based data model, KGs model entities (nodes) and their relationships (edges), and both entities and relationships can have attributes or properties associated with them. KGs are used to represent and organize knowledge in a way that is more accessible and understandable to machines and humans alike. KGs are:

- **Declarative** (meaningful).
- **Annotated** (enriched with contextual information).
- **Large** (large number of entities and their relations with other entities)

The following sections describes some key features and use cases of KGs from book [14].

2.1.1 RDF

Resource Description Framework (RDF) is the W3C standard data model for representing and exchanging structured information on the Web. RDF is a core component of the Semantic Web, a vision of the World Wide Web in which data is not only presented in a human-readable format, but also structured in a way that computers can understand and process.
RDF data can be serialized in various formats, including RDF/XML, Turtle, JSON for Linking Data (JSON-LD), and N-Triples, among others. This flexibility in serialization allows RDF graphs to be easily exchanged between different systems and applications to make explicit additional details and meta-data.
RDF is used in a wide range of applications, including the Semantic Web, KGs, data integration, data interchange, search engines, recommendation systems, and more. It provides a foundation for structuring and linking data on the web, making information more accessible and meaningful for both humans and machines.

2.1.2 OWL

Web Ontology Language (OWL) [25] is a Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things. OWL is a computational logic-based language such that knowledge expressed in OWL can be exploited by computer programs, e.g., to verify the consistency of that knowledge or to make implicit knowledge explicit. OWL documents, known as ontologies, can be published in the World Wide Web and may refer to or be referred from other OWL ontologies. OWL is part of the W3C's Semantic Web technology stack, which includes RDF, SPARQL, etc.

2.1.3 Data Modeling

Data modeling in the context of KGs refers to the process of defining and structuring the data, including entities, relationships, and attributes, in a way that accurately represents a specific domain or knowledge area. It involves creating a formal and organized representation of the data in a graph-based format that enables efficient storage, retrieval, and analysis of information.
Here are some key aspects of data modeling in KGs:

- **Classes or Entity Types**: a presentation type of entities.
- **Attributes**: a property representing a characteristic of an entity.

- **Relationships**: set of associations among the same sets of entities.

Data modeling in a KG is typically an iterative process, as the structure of the graph and the data it contains may evolve over time to better capture and represent the knowledge within a specific domain. As an illustration, the age range within a KG in one case may simply range from $['Young','Old']$ while in another KG it may cover a broader range, including $['Infants','Children','Adolescents','Adults','Seniors']$. The specific configuration depends entirely on the unique requirements and use case of the respective KG. This is a critical step in creating a robust and effective KG for various applications, including semantic search, data integration, and knowledge discovery.

2.1.4 SPARQL

SPARQL is the standard query language for querying RDF data. It plays a crucial role in the Semantic Web and linked data environments. It provides a standardized and powerful way to retrieve and manipulate RDF data, enabling complex queries to be expressed and executed over the Semantic Web and linked data resources.

2.1.5 Data Integration

KGs are used to integrate data from disparate sources, which can include structured databases, unstructured text, and web data. This integration is valuable for tasks like data integration, data enrichment, and data analytic.
Data integration is a computational task where data collected from multiple autonomous data providers are merged into a unified data in order to offer uniform access to a set of autonomous and heterogeneous data sources. A data integration system Data Integration System (DIS) is a triple $DIS =< O, S, M >$ [11]:

- O is an ontology which provides a uniform view to the data sources in S.
- S is a set of the signatures of the data sources that compose a DIS.
- M is a set of mappings between signatures in S and concepts in O.

Challenges of Data Integration [4]

- **Query**: processing queries over disparate data sources and offering a uniform view.

- **Number of sources**: the complexity of data integration depends on the number of data sources. However, data integration can be challenging even for two data sources.
- **Heterogeneity**: data sources are developed independently. Data sources may suffer from various heterogeneity conflicts, e.g., different schemata, meaning of the attributes, and format.
- **Autonomy**: data sources may change their data formats and access patterns at any time, without having to notify any central administrative entity.

2.1.6 Mapping Languages

A mapping language is a declarative programming language to express how the data from the data sources in S is used to populate the concepts (i.e., classes and properties) in O. Mapping languages are used to define the identifiers of the entities of the classes in O based on the attributes of sources in S. Also they are used to define the properties (data type and object properties) in O in terms of attributes of the data sources in S.
RDB to RDF Mapping Language (R2RML) [1] [9] and RDF Mapping Language (RML)[2] [12] are two different mapping languages used in the context of transforming data from relational databases into RDF data, which is often used in the Semantic Web and linked data environments. These languages enable the mapping of structured, tabular data in relational databases to the graph-based format of RDF.

2.1.7 Closed and Open World Assumptions

The Closed World Assumption (Closed World Assumption (CWA)) and the Open World Assumption (Open World Assumption (OWA)) are two different principles used in KGs. They describe how we handle information when making statements or inferences about the world.

- **CWA**: Under the Closed World Assumption, any statement that is not explicitly known to be true is assumed to be false. In other words, if a piece of information is not in the knowledge base or database, it is considered negated or denied. CWA operates on the premise that if something is true, it should be documented or explicitly stated, and any unmentioned information is treated

[1]Source: https://www.w3.org/TR/r2rml/
[2]Source: https://rml.io/specs/rml/

as false.
CWA is helpful in situations where the available information is complete and well-defined, but it may not be suitable for domains where new information is continually emerging or where it's impossible to document all relevant facts.

- **OWA**: Under the Open World Assumption, the absence of information does not imply that something is false. In other words, in an open world, if a fact or statement is not in the knowledge base, it is neither affirmed nor denied; it is simply unknown. The world is considered to be open, with the possibility of unrecorded or undiscovered information.
  OWA is commonly used in knowledge representation systems, semantic web technologies, and many AI and reasoning systems. It allows for a more flexible and realistic representation of incomplete or evolving knowledge. OWA acknowledges that our knowledge is often incomplete, and there may be new, unanticipated information that can change our understanding of the world. Inference in an open world often involves reasoning with uncertainty and handling incomplete or uncertain information.

2.2 A Lung Cancer Knowledge Graph

Cancer Long Survivor AI Follow-up is a study supported by the European Union Horizon 2020 Research and Innovation Program under grant agreement nº 875160 CLARIFY project. Big data and AI for monitoring health status and quality of life after the cancer treatment [30].
Technische Informationsbibliothek (TIB) in Germany have merged data from Electronic Health Records (EHR) and database from Hospital Universitario Puerta de Hierro (HUPHM) and the Thoracic Tumors Registry (TTR) from the Spanish Lung Cancer Group (SLCG). The methods reported in this section are computed based on the data extracted during the execution of some queries over TTR KG.
Data collected from the Thoracic Tumors Registry database shared by the Spanish Lung Cancer Group (SLCG) is represented into the KG; it is accessible only to the authorized partners of the CLARIFY[3] consortium via Web APIs (Application Programming Interfaces). KGs describe entities (objects) of interest and their connections. They represent the convergence of knowledge and data as factual statements. A KG is a knowledge base that uses a graph structured data model to integrate data. They provide a compact formal specification of the meaning of entities, model

[3]Source: https://www.clarify2020.eu/

taxonomies of entities, relations, and classes, and develop a common understanding of a domain.

2.3 Inductive Knowledge

Inductively acquiring knowledge involves generalizing patterns from a given set of input observations, which can then be used to generate novel, but potentially imprecise predictions. For example, from a large data graph with lung cancer patients, we may observe the pattern that almost all person who their diagnosed stage is `IV`, their cancer type of `Small_cell_lung_cancer` and hence predict that if patient1 in this KG has stage of diagnose `IV`, he or she likely has cancer type of `Small_cell_lung_cancer`; however, the predictions drawn from this pattern do not hold for certain, where there are also Patients in KG who their diagnosed stage is `IV`, he or she has cancer type of `Non_Small_cell_lung_cancer`. Hence, predictions will often be associated with a level of confidence; e.g., we may say that a lung cancer patient with diagnosed stage `IV` has cancer type of `Small_cell_lung_cancer` in $\frac{1257}{1262}$ of cases, offering a confidence of 0.99 for predictions made with that pattern.
We then refer to knowledge acquired inductively as inductive knowledge, which includes both the models used to encode patterns, and the predictions made by those models. Though fallible, inductive knowledge can be highly valuable. Figure 2.1 presents an overview of inductive techniques for KGs.
In the case of unsupervised methods, there is a rich body of work on graph analytics, which uses well-known functions/algorithms to detect communities or clusters, find central nodes and edges, etc., in a graph. Alternatively, KG embedding models can use self-supervision to learn a low-dimensional numeric model of a KG that typically maps input edges to an output plausibility score indicating the likelihood of the edge being true.
The structure of graphs can also be directly leveraged for supervised learning, as explored in the context of graph neural networks. Finally, while the aforementioned techniques learn numerical models, symbolic learning can learn symbolic models – i.e., logical formulae in the form of rules or axioms – from a graph in a self-supervised manner. We now discuss each of the aforementioned techniques in turn.

2.3.1 Terminology

The following is an overview of the terminology from Book [14] used in this paper:

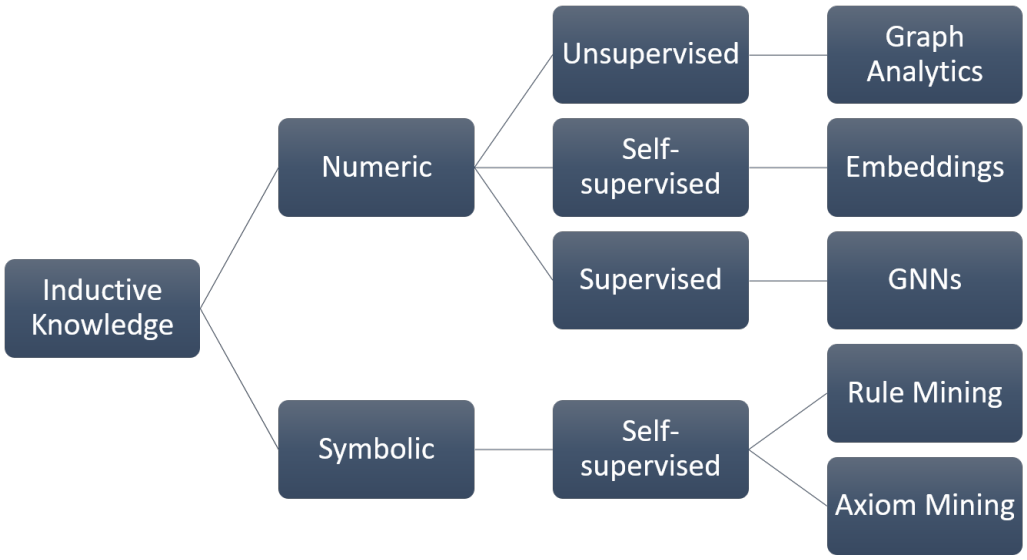


Figure 2.1: Conceptual overview of popular inductive techniques for KGs in terms of type of representation generated (Numeric/Symbolic) and type of paradigm used (Unsupervised/Self-supervised/Supervised)

Directed Edge-Labelled Graphs: A directed edge-labelled graph (sometimes known as a multi-relational graph [2] [6] [23] is defined as a set of nodes and a set of directed labelled edges between those nodes. In the case of KGs, nodes are used to represent entities and edges are used to represent (binary) relations between those entities. In the following, a directed edge-labelled graph is formally defined, where Con is countably infinite set of constants.
A directed edge-labelled graph is a tuple $G = (V\ E\ L)$, where $V \subset Con$ is a set of nodes, $L \subset Con$ is a set of edge labels, and $E \subset V \times L \times V$ is a set of edges.

RDF: the W3C standard data model to represent KGs. RDF graphs corresponds to directed edge-labelled graphs [8].

Numerical Inductive Knowledge: Inductive knowledge is learned following a numerical or embedding model.

Symbolic Inductive Knowledge: Inductive knowledge described based on a symbolic model, i.e., a set of rules or axioms.

Unsupervised Learning: They are models that learn on its own using the data,

which is received as input. Unsupervised algorithms create clusters or communities of similar types of data.

Supervised Learning: Previously collected and labelled data is utilized to train leaning models capable of predicting results that are more accurate.

Self-Supervised Learning: These models require supervisory data which are not labelled by humans. Models resort to embedded metadata as supervisory data (labels generated automatically) which are used by some supervised learning task.

Graph Analytic: Process of discovering, interpreting, and communicating meaningful patterns existing in graph data. Algorithms derive conclusions based on how the nodes are connected, i.e., based on the data graph topology.

KGs Embedding Models: Resort to self-supervised learning to generate a low-dimensional numeric model of KG. Graph Neural Networks: transformation on all the components of a graph (i.e., nodes and edges) that preserves graph symmetries and connectivity. The definition of KG embedding is defined in the following:
Given a directed edge-labelled graph $G = (V, E, L)$, a KG embedding of G is a pair of mappings (ϵ, ρ) such that $\epsilon : V \to \mathbb{T}$ and $\rho : L \to \mathbb{T}$.

Community Detection: Techniques that partition a graph into subgraphs composed of densely connected and similar nodes.

Rule Mining: techniques to discovering meaningful patterns in the form of rules from large collections of background knowledge.

Axiom Mining: Techniques to discovering meaningful patterns that correspond to axioms expressed in a logical language, e.g., Description Logic.

Vector, Matrix, Tensor, Order, Mode: For any positive integer a, a vector of dimension a is a family of real numbers indexed by integers in $\{1, ..., a\}$. For a and b positive integers, an (a, b) - matrix is a family of real numbers indexed by pairs of integers in $\{1, ..., a\} \times ... \times \{1, ..., b\}$. A tensor is a family of real numbers indexed by a finite sequence of integers such that there exist positive numbers $a_1, ..., a_n$ such that the indices are all the tuples of numbers in $\{1, ..., a_1\} \times ... \times \{1, ..., a_n\}$.
For example, if $a_1 = 2$ and $a_2 = 3$, the indices can be tuples from the set $1, 2 * 1, 2, 3$, the result is in pairs like $(1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3)$. Each tuple corresponds

to a specific element in the tensor, allowing to organize and access the real numbers within the tensor structure.
The number n is called the order of the tensor, the sub indices $i \in \{1, ..., n\}$ indicate the mode of a tensor, and each a_i defines the dimensions of the i^{th} mode. A 1-order tensor is a vector and a 2-order tensor is a matrix.$\mathbb{T}$ is the set of all tensors.

Plausibility Scores: A plausibility scoring function is a partial function :. Given a directed edge-labelled graph$G = (V, E, L)$, an edge $(s, p, o) \in V \times L \times V$, and a KG embedding (ϵ, ρ) of G, the plausibility of (s, p, o) is given as $\phi(\epsilon(s), \rho(p), \epsilon(o))$.
The objective of a plausibility score function is to assign high values of ϕ the triples that are likely to be true and low scores to triples that are likely to be false.

Matrix Multiplication: The multiplication of matrices $X \in \mathbb{R}^{a,b}$ and $Y \in \mathbb{R}^{b,c}$ is a matrix $XY \in \mathbb{R}^{a,c}$ such that $(XV)_{ij} = \sum_{k=1}^{b}(X)_{ik}(Y)_{kj}$.
The matrix multiplication of two tensors $\mathcal{X} \in \mathbb{R}^{a_1,...,a_m,c}$ and $\mathcal{Y} \in \mathbb{R}^{c,b_1,...,b_n}$ is a tensor $\mathcal{XY} \in \mathbb{R}^{a_1,...,a_m,,b_1,...,b_n}$ such that $(\mathcal{XY})_{i_1...i_m i_{m+1}...i_{m+n}} = \sum_{k=1}^{c}(\mathcal{X})_{i_1...i_m k}(\mathcal{Y})_{k i_{m+1} i_{m+n}}$.

L^p-Norm, $L^{p,q}$-Norm: For $p \in \mathbb{R}$, the L^p-norm of a vector $x \in \mathbb{R}^a$ is the scalar $||x||_p := (|(x)_1|^p + ... + |(x)_a|^p)^{\frac{1}{p}}$, where $|(x)_i|$ denotes the absolute value of the i^{th} element of x. For $p, q \in \mathbb{R}$, the $L^{p,q}$-norm of a matrix $X \in \mathbb{R}^{a,b}$ is the scalar $||X||_{p,q} := (\sum_{j=1}^{b}(\sum_{i=1}^{a}|(X)_{ij}|^p)^{\frac{q}{p}})^{\frac{1}{q}}$.

Hadamard Product: Given two tensors $\mathcal{X} \in \mathbb{R}^{a_1,...,a_n}$ and $\mathcal{Y} \in \mathbb{R}^{a_1,...,a_n}$, the Hadamard product $\mathcal{X} \odot \mathcal{Y}$ is defined as a tensor in $\mathbb{R}^{a_1,...,a_n}$, with each element computed as $(\mathcal{X} \odot \mathcal{Y})_{i_1...i_n} := (\mathcal{X})_{i_1...i_n}(\mathcal{Y})_{i_1...i_n}$.

Tensor Product: Given two tensors $\mathcal{X} \in \mathbb{R}^{a_1,...,a_m}$ and $\mathcal{Y} \in \mathbb{R}^{b_1,...,b_n}$,the tensor product $\mathcal{X} \otimes \mathcal{Y}$ is defined as a tensor in $\mathbb{R}^{a_1,...,a_m,b_1,...,b_n}$, with each element computed as $(\mathcal{X} \otimes \mathcal{Y})_{i_1...i_m j_1...i_n} := (\mathcal{X})_{i_1...i_m}(\mathcal{Y})_{j_1...j_n}$.

n-Mode Product: For a positive integer n, a tensor $\mathcal{X} \in \mathbb{R}^{a_1,...,a_{n-1},a_n,a_{n+1},...,a_m}$ and matrix $\mathcal{Y} \in \mathbb{R}^{b,a_n}$, the n-mode product $\mathcal{X}$ on and $\mathcal{Y}$ is the tensor $\mathcal{X} \otimes_n \mathcal{Y} \in \mathbb{R}^{a_1,...,a_{n-1},a_n,a_{n+1},...,a_m}$ such that $(\mathcal{X} \otimes_n \mathcal{Y})_{i_1...i_{n-1}j,i_{n+1},...,i_m} := \sum_{k=1}^{a_n}(\mathcal{X})_{i_1...i_{n-1}j,i_{n+1},...,i_m}(\mathcal{Y})_{jk}$

Convolution: Given two matrices $\mathbf{X} \in \mathbb{R}^{a,b}$ and $\mathbf{Y} \in \mathbb{R}^{e,f}$, the convolution of $\mathbf{X}$ and $\mathbf{Y}$ is the matrix $\mathbf{X} * \mathbf{Y} \in \mathbb{R}^{(a+e-1),(b+f-1)}$ such that
$(\mathbf{X} * \mathbf{Y})_{ij} = \sum_{k=1}^{a}\sum_{l=1}^{b}(\mathbf{X})_{kl}(\mathbf{Y})_{(i+k-a)(j+l-b)}$.
The convolution is defined as an operator per the widely-used convention for con-

volutional neural networks. Strictly speaking, the operator should be called cross-correlation, where traditional convolution requires the matrix X to be initially "rotated" by 180°. Since in our settings the matrix X is learned, rather than given, the rotation is redundant, and hence the distinction is not important.

Reverse Relations: The relation r^{-1} is the inverse of the relation r if for every triple (h, r, t) in a KG, the triple (h, r^{-1}, t) also exists in the KG.

Symmetric Relations: r_1 and r_2 are symmetric relations if for every triple (h, r_1, t) in a KG, the triple (t, r_2, h) also exists in the KG.

Duplicate Relations:r_1 and r_2 are duplicate relations if for every triple (h, r_1, t) in a KG, the triple (h, r_2, t) also exists in the KG.

CVT (Compound Value Type): It is a mediator node that represent multiply relationships [1]. Figure 2.2 shows an example of CVT node in KG.

Cartesian Product Relation: A Cartesian product relation has a set of subjects and a set of objects, and the relation exists from every subject in the first set to every object in the second set [1].

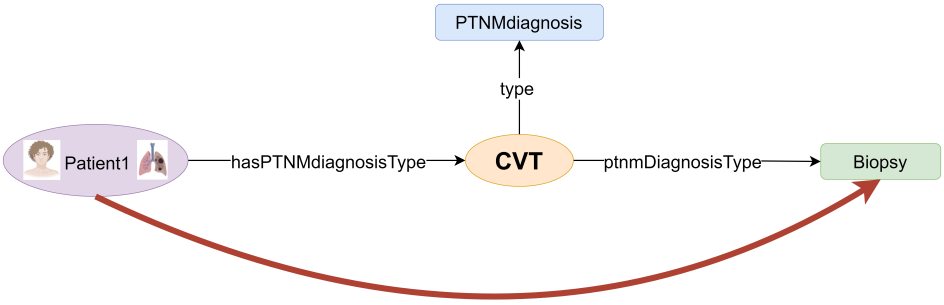


Figure 2.2: Example of mediator (CVT) nodes in lung cancer KG

2.4 Models for Knowledge Graph Embeddings

KG embedding models are techniques used in the field of knowledge representation and AI to transform structured data from KGs into continuous vector representa-

tions [14]. The primary goal of KG embedding models is to capture the relationships and properties of entities and concepts in a way that can be easily used by machine learning models for various tasks such as link prediction, entity classification, and recommendation systems. These models enable the integration of symbolic knowledge from KGs with numerical learning methods, but they do not capture semantics. In recent years, machine learning methods have gained significant attention, particularly in the context of KGs. Machine learning is used to refine KGs directly or for downstream tasks, including recommendation, information extraction, question answering, query relaxation, and query approximation. However, a challenge arises as traditional machine learning techniques typically rely on dense numeric input vectors, which differ from how graphs are typically represented. This raises the question of how to encode graphs, nodes, and edges as numeric vectors.

The initial approach to represent a graph using vectors is through one-hot encoding, which creates a vector for each node with a length of $|L|.|V|$. Here, $|V|$represents the number of nodes in the input graph, and $|L|$ is the number of edge labels. In this encoding, one is placed at the appropriate index to signify the presence of a specific edge in the graph, while other positions are filled with zero. However, this method typically results in large and sparse vectors, which can be problematic for many machine learning models.

KG embedding techniques aim to generate a compact representation of a graph in a low-dimensional vector space, enabling their use in machine learning tasks. These embeddings typically have a fixed and low dimensionality (denoted as d). So, graph embeddings consist of entity embeddings for nodes (vectors denoted as e) and relation embeddings for edge labels (vectors denoted as r). These embeddings aim to capture and preserve underlying graph structures. Various techniques can create embeddings, commonly involving a scoring function. This function takes entity embeddings and relation embeddings as inputs and calculates the plausibility of edges, indicating their likelihood to be true.

In practice, the goal is to learn vectors of dimension d that maximize the plausibility of positive edges and minimize the plausibility of negative examples based on the scoring function. These vectors can be seen as self-supervised models encoding latent graph features. They are used for multiple tasks, such as assessing edge confidence, completing edges with missing information, and measuring similarity between nodes and edge labels. The similarity measures derived from embeddings can be applied to various tasks, including identifying duplicate nodes that represent the same entity and providing recommendations.

A wide range of KG embedding techniques have been proposed [31], but in the following three categories of embedding models are explored: Translational models, which

take a geometric perspective and use relation vectors to translate subject entities to object entities. Tensor decomposition models, which extract latent factors to approximate the structure of the graph. Neural models, which employ neural networks to train vectors that yield precise plausibility scores. In the following, We discuss tensor-based approaches of KG embedding models.

2.4.1 Translational Models

Translational models interpret edge labels as transformations from subject nodes (aka the source or head) to object nodes (aka the target or tail), e.g., in the edge $Patient1 - smokerType \rightarrow HeavySmoker$ (Figure 1.1), the edge label $smokerType$ is seen as transforming $Patient1$ to $HeavySmoker$.

TransE

The most elementary approach in translational models is TransE [6]. For each triple (s, p, o) that is true in a KG, TransE learn that the embedding of the tail (o) should be close to the embedding of the head (s) plus the vector of the property (p). Conversely, if the triple (s, p, o) is false in a KG, TransE attempts to learn a representation that keeps the embedding of the tail (o) away from the embedding of the head (s) plus the vector of the property (p). TransE relies on a reduced set of parameters, as it learns only one low-dimensional vector for each entity and each relationship.
To model translational models of intuition, TransE uses a scoring function to measure the plausibility of a triple (s, p, o). The scoring function typically uses the $L1$ or $L2$ distance (e.g., Euclidean distance) between $s + p$ and o. The lower the distance, the more plausible the triple is considered. The scoring function of TransE model is defined as follows:

$$\phi(\varepsilon(s), \rho(p), \varepsilon(o)) = -||e_s + r_p - e_o||_q \tag{1}$$

with condition:

$$e_x \in \mathbb{R}^d, r_y \in \mathbb{R}^d, q \in \{1, 2\}, ||e_x||_2 = 1 \tag{2}$$

The high values of ϕ (more plausible) to triples that are likely to be true and low scores to triples that are likely to be false. TransE has some limitations, such as struggling to model asymmetric relations and handling more complex patterns in KGs.

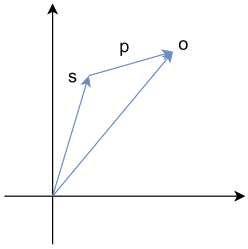


Figure 2.3: Simple illustration of TransE that connect s (subject) and o (object) with relation (p)

TransH

TransH [32] aims to address TransE's limitations by not using the same embedding of an entity in different relations. Translation on hyperplanes (TransH) interprets a relation as a translating operation on a hyperplane. In TransH, each relation is characterized by two vectors, the norm vector (w_p) of the hyperplane, and the translation vector (r_p) on the hyperplane. For a golden triplet (s, p, o), that it is correct in terms of worldly facts, the projections of s and o on the hyperplane are expected to be connected by the translation vector r_p with low error.
This simple method overcomes the flaws of TransE in dealing with its limitations while keeping the model complexity almost the same as that of TransE. The scoring function of TransH model is defined as follows:

$$\phi(\varepsilon(s), \rho(p), \varepsilon(o)) = -||e_s - ((e_s^T w_p)w_p) + r_p - (e_o - (e_o^T w_p)w_p)||_2^2 \tag{3}$$

with condition:

$$e_x \in \mathbb{R}^d, r_y \in \mathbb{R}^d, w_y \in \mathbb{R}^d, ||w_y||_2 = 1, \frac{w_y^T r_y}{||r_y||_2}, ||e_x||_2 \leq 1 \tag{4}$$

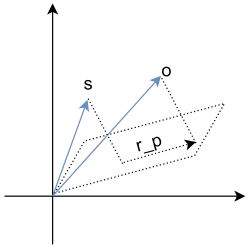


Figure 2.4: Simple illustration of TransH that connect a subject (s) and object (o with relation (r_p)

TransR

TransR [19] is a model that learns vectors for entities and relations in two separate vector spaces, denoted as $\mathbb{R}^{d_d}$ and $\mathbb{R}^{d_k}$, respectively. Unlike other models like TransE and TransH, which use a single semantic space for both entities and relations, TransR argues that this approach is inadequate because entities and relations are fundamentally distinct. To address this, TransR introduces a projection matrix, which is used to map entity embeddings into the appropriate vector space for each relation.The scoring function of TransR model is defined as follows:

$$\phi(\varepsilon(s), \rho(p), \varepsilon(o)) = -||W_p e_s + r_p - W_p e_o||_2^2 \tag{5}$$

with condition:

$$e_x \in \mathbb{R}^{d_e}, r_y \in \mathbb{R}^{d_r}, W_y \in \mathbb{R}^{d_r, d_e}, ||e_x||_2 \leq 1, ||r_y||_2 \leq 1, ||W_y e_x||_2 \leq 1 \tag{6}$$

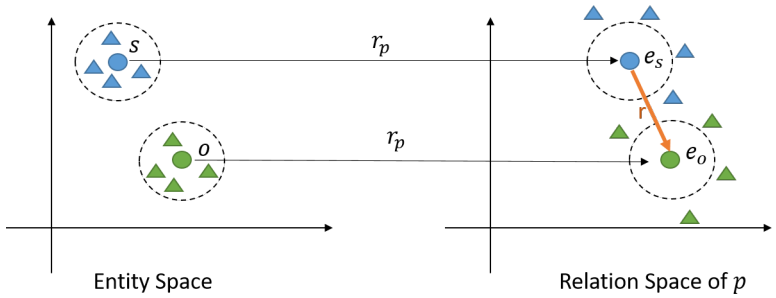


Figure 2.5: Simple illustration of TransR that connect s (subject) and o (object) with relation (r)

TransD

TransD [17] is an advancement over TransR, where the projection matrix is split into two vectors. Unlike TransR, TransD employs a distinct projection matrix for each combination of entities and relations because different types of entities have different attributes and functions, it is insufficient to let them share the same transform parameters of a relation. And for a given relation, similar entities should have similar mapping matrices and otherwise for dissimilar entities.
Furthermore, the mapping process is a transaction between entities and relations that both have various types. TransD considers different types of both entities and relations, to encode KGs into embedding vectors via dynamic mapping matrices

produced by projection vectors.The scoring function of TransD model is defined as follows:

$$\phi(\varepsilon(s), \rho(p), \varepsilon(o)) = -||(w_p \otimes w_s + I)e_s + r_p - (w_p \otimes w_o + I)e_o||_2^2 \tag{7}$$

with condition:

$$e_x \in \mathbb{R}^{d_e}, r_y \in \mathbb{R}^{d_r}, w_x \in \mathbb{R}^{d_e}, w_y \in \mathbb{R}^{d_r}, ||e_x||_2 \leq 1, ||r_y||_2 \leq 1, ||(w_y \otimes w_x + I)e_x||_2 \leq 1 \tag{8}$$

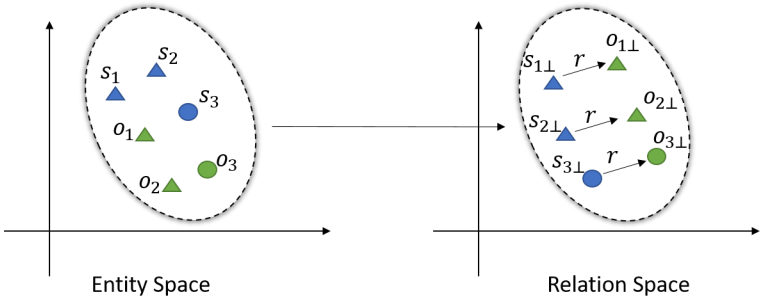


Figure 2.6: Simple illustration of TransD that connect s (subject) and o (object) with relation(r)

RotatE

RotatE [28] extends the idea of TransE by modeling relations as rotations in a complex vector space. It uses rotations in the complex vector space to calculate the score function. The scoring function is defined as follows:

$$\phi(\varepsilon(s), \rho(p), \varepsilon(o)) = -||e_s \odot r_p - e_o||_2 \tag{9}$$

with condition:

$$e_x \in \mathbb{C}^d, r_y \in \mathbb{C}^d, ||r_y||_2 = 1 \tag{10}$$

2.4.2 Tensor Decomposition Models

The second approach to create graph embedding models involves using tensor decomposition methods. Tensors are multidimensional numeric structures that generalize scalars (0-order tensors), vectors (1-order tensors), and matrices (2-order tensors) to higher dimensions. Tensors are widely used in machine learning. Tensor decomposition entails breaking down a tensor into more basic tensors with lower orders. These

elemental tensors are used to reconstruct or approximate the original tensor through a fixed sequence of basic operations. These elemental tensors capture latent factors that underlie the information in the original tensor. Various tensor decomposition methods exist, with rank decompositions being one of the key concepts [26].
In the context of graphs, similar principles can be applied to break down a graph into vectors, which results in embedding models. To do this, a graph can be represented as a one-hot 3-order tensor denoted as G, with a size of $|V| \times |L| \times |V|$ elements. In this tensor, an element $(\mathcal{G})_{ijk}$ is set to one if the i^{th} node is connected to the k^{th} node with an edge labeled as the j^{th} label, and it's set to zero otherwise.
It's important to note that such a tensor is often very large and sparse, making it suitable for rank decompositions. A Canonical Polyadic (CP) decomposition [16] would compute a sequence of vectors $(x_1, y_1, z_1, ..., x_d, y_d, z_d)$ such that $x_1 \otimes y_1 \otimes z_1 + ... + x_d \otimes y_d \otimes z_d \approx \mathcal{G}$.

DistMult

DistMult [33] is a seminal method for computing KG embeddings based on rank decompositions and a simplified and efficient model that models the interactions between entities and relations in a KG by applying a simple bilinear dot product scoring function. DistMult is particularly well-suited for modeling symmetric relations in KGs. The core idea of DistMult is to model the compatibility between an entity, a relation, and another entity using a bilinear dot product. The scoring function for a triple (s, p, o) is defined as follows:

$$\phi(\varepsilon(s), \rho(p), \varepsilon(o)) = e_s^T r_p^D e_o \tag{11}$$

with condition:

$$e_x \in \mathbb{R}^d, r_y \in \mathbb{R}^d, ||e_x||_2 = 1, ||r_y||_2 \leq 1 \tag{12}$$

A weakness of this approach is that per the scoring function, the plausibility of triple (s, p, o) will always be equal to that of triple (o, p, s); in other words, DistMult does not consider edge direction.

ComplEx

ComplEx [29] extends DistMult to the complex domain. Each entity and relation is represented as a complex-valued vector. By using complex numbers, the ComplEx model can capture more expressive and fine-grained semantics of relationships in KGs, making it well suited to modeling complex relationships.

The scoring function for a triple (s, p, o) for ComplEx model is defined as follows ($Re()$ denotes the real part of the result):

$$\phi(\varepsilon(s), \rho(p), \varepsilon(o)) = Re(e_s^T r_p^D \bar{e}_o) \tag{13}$$

with condition:

$$e_x \in \mathbb{C}^d, r_y \in \mathbb{C}^d, ||e_x||_2 \leq 1, ||r_y||_2 \leq 1 \tag{14}$$

While ComplEx model excels at capturing complex relationships, it may not be the best choice for simpler, more structured KGs because ComplEx involves a trilinear tensor decomposition, which significantly increases the model's computational complexity.

TuckER

TuckER [3] is a KG embedding model that uses a Tucker Decomposition to represent entities and relations in a KG. It decomposes the KG into a core tensor T and three matrices A, B, and C, such that $\mathcal{G} = T \otimes A \otimes B \otimes C$. The model represents entity embeddings using matrices A and C, while relation embeddings are captured by matrix B. This decomposition approach allows TuckER to capture complex semantic relationships in the KG and efficiently model entity-relation interactions.
TuckER decomposition is a way to break down a multidimensional tensor into a set of smaller tensors that capture essential information. The decomposition involves a Core Tensor ($\mathcal{W}$) that represents the shared interactions or relationships between entities, relationships, or attributes in the original tensor.
It's a compact representation of the most significant patterns in the data.The scoring function for a triple (s, p, o) for TuckER model is defined as follows:

$$\phi(\varepsilon(s), \rho(p), \varepsilon(o)) = \mathcal{W} \otimes_1 e_s^T \otimes_2 r_p^D \otimes_3 e_o^T) \tag{15}$$

with condition:

$$e_x \in \mathbb{R}^{d_e}, r_y \in \mathbb{R}^{d_r}, \mathcal{W} \in \mathbb{R}^{d_e, d_r, d_e} \tag{16}$$

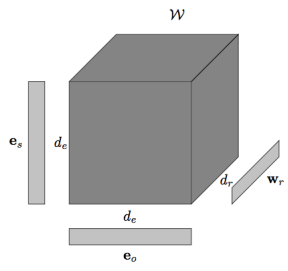


Figure 2.7: Visualization of the TuckER architecture.s. It is taken from [3]

In practice, these dimensions ($d_r = |L|$ and $d_e = |V|$) can become impractical due to computational limitations. This property demonstrates that TuckER does not have inherent restrictions in representing graphs, but may not be feasible with very high dimensions.

2.4.3 Neural Models

The mentioned approaches have a limitation in that they rely on either linear or bilinear operations for calculating plausibility scores based on embeddings. However, some alternative methods employ neural networks to learn embeddings and use non-linear scoring functions for assessing plausibility.

ConvE

ConvE [10] is a neural network model that employs 2D convolutional layers on embeddings to model interactions between entities and relations. It generates a matrix by combining vectors associated with entities (e_s) and relations (r_p), and this matrix is used as input for 2D convolutional layers. The resulting feature map tensor is transformed into a vector in a parameterized linear way, projected into a dimension denoted as d. The plausibility score for a triple(s, p, o) in the ConvE model is determined by the dot product of this vector and the embedding vector for the object entity (e_o). The scoring function for a triple (s, p, o) for ConvE model is defined as follows:

$$\phi(\varepsilon(s), \rho(p), \varepsilon(o)) = \psi \left(\begin{array}{l} vec \left(\psi \left(\mathcal{W} * \left[\begin{array}{l} e_s^{[a,b]} \\ r_p^{[a,b]} \end{array} \right] \right) \right) \mathbf{W} \\ \end{array} \right) \tag{17}$$

with condition:

$$e_x \in \mathbb{R}^d, r_y \in \mathbb{R}^d, d = ab, \mathbf{W} \in \mathbb{R}^{w_1(w_2+2a-1)(w_3+b-1),d}, \mathcal{W} \in \mathbb{R}^{w_1,w_2,w_3} \tag{18}$$

A disadvantage of ConvE is that by wrapping vectors into matrices, it imposes an artificial two-dimensional structure on the embeddings.

2.5 Symbolic Learning

The supervised techniques discussed thus far–namely KG embeddings–learn numerical models over graphs. However, such models are often difficult to explain or understand. KG embeddings will not provide an interpretable model to help understand why this is the case: the reason for the result may lie in a matrix of parameters acquired to fit a plausibility score on training data. Such approaches also suffer from the out-of-vocabulary problem, where they are unable to provide results for edges involving previously unseen nodes or edges.
An alternative (sometimes complementary) approach is to adopt symbolic learning in order to learn hypotheses in a symbolic (logical) language that "explain" a given set of positive and negative edges. These edges are typically generated from the KG automatically (similar to the case of KG embeddings). The hypotheses then serve as interpretable models that can be used for further deductive reasoning. In this section, we discuss one form of symbolic learning: rule mining, which learns rules.

2.5.1 Rule Mining

Rule mining in KGs involves the discovery of meaningful patterns in the form of rules from extensive background knowledge. In this context, positive edges are typically observed or inferred from a KG, while negative edges are defined based on an assumption of completeness. The goal of rule mining is to identify new rules that entail a high ratio of positive edges from other positive edges, but entail a low ratio of negative edges from positive edges.
Rule mining applies to a set of facts in the KG that infer another fact. Fact or atom in a KG is typically expressed as triple. A-Box and T-Box are two type of facts in KG. A-Box are instance data and T-Box define classes, domains, ranges for predicates and the class hierarchy.

Horn Rules

A Horn rules are defined as **Body:- Head**, where **Body** is a conjunction of predicate facts and **Head** is a predicate fact. There are three types of Horn rules:

- **Safe**: Horn Rules are safe if every variable in **Head** is also in **Body**.

- **Connected**: Horn rules are connected if every fact is connected transitively

to every other fact of the rule.

- **Close**: Horn rules are close if every variable in the rule appears at least twice.

Mining Model

Let us consider a given Horn rule $\vec{B} \Rightarrow r(x, y)$. Let us look at all facts with relation r (Figure 2.8). We distinguish four types of facts: True facts that are known to the Knowledge Base (KBtrue), true facts that are unknown to the Knowledge Base (KB) (NEWtrue), facts that are known to be false to the KB (KBfalse), and facts that are false, but unknown to the KB (NEWfalse). The rule will make certain predictions (blue circle). These predictions can be known to be true (A), known to be false (C), or unknown (B and D). When they are unknown to the KB, they can still be true (B) or false (D) with respect to the real world.

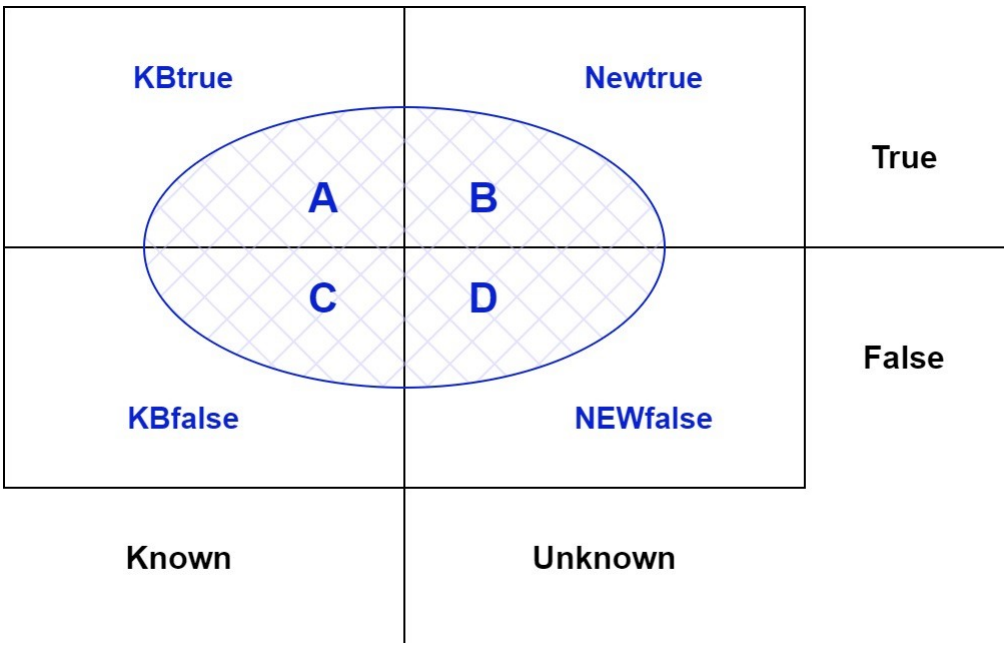


Figure 2.8: Prediction under incompleteness [15]

Our goal is to find rules that make true predictions that go beyond the current KB. In the figure, we wish to maximize the area B, and to minimize the area D.

There are two obvious challenges in our context: First, the areas NEWtrue and NEWfalse are unknown. So if we wish to maximize B at the expense of D, we are operating in an area outside our KB. We would want to use the areas KBtrue and KBfalse to estimate the unknown area. This, however, leads to the second challenge: Semantic knowledge bases do not contain negative evidence. Thus, the area KB-false is empty. We will now present different measures that address these challenges in representative of the observed feature models is the rule mining system AMIE [15].

- **Support**:the support of a rule as the number of distinct pairs of subjects and objects in the head of all instances found in the knowledge base. The support is calculated as follows:

$$supp(\vec{B} \Rightarrow r(x,y)) := \#(x,y) : \exists z_1, ..., z_m : \vec{B} \wedge r(x,y) \tag{19}$$

  where $z_1, ..., z_m$ are the variables of the rule apart from x and y.

- **Head Coverage**: support is an absolute measure, which makes it dependent on the knowledge base's size for meaningful assessment. To address this, a proportional version of support is introduced, calculated by dividing the absolute support by the size of KB. However, this approach may exclude relations with few facts. To account for this, head coverage is introduced, which measures the proportion of subject-object pairs from the head relation that the rule's predictions cover. This helps in learning rules even for relations with limited facts. The head coverage is calculated as follows:

$$hc(\vec{B} \Rightarrow r(x,y)) := \frac{supp(\vec{B} \Rightarrow r(x,y))}{\#(x',y') : r(x',y')} \tag{20}$$

- **Negative Examples**: the primary challenge in this setting is to generate counter-examples for rule mining, which help estimate areas where rules can be both NEWtrue and NEWfalse. Various methods exist to address this challenge, including standard confidence measures, positive-only learning evaluation scores from Inductive Logic Programming (Inductive Logic Programming (ILP)), and a new approach based on the partial completeness assumption.

- **Standard Confidence**: the standard confidence measure evaluates a rule by considering all facts not in the knowledge base (NEWtrue and NEWfalse) as negative evidence. It calculates the ratio of the rule's predictions that match facts in the knowledge base, which quantifies the share of true predictions

among all predictions made by the rule.
The standard confidence metric treats both 'false' and 'unknown' as the same, essentially assuming a closed world setting. It focuses on describing the known data and penalizes rules that make many predictions in the unknown region. In contrast, the approach being discussed aims to maximize the number of true predictions beyond existing knowledge. It's not about describing data, but predicting new data. The standard confidence is calculated as follows:

$$conf(\vec{B} \Rightarrow r(x,y)) := \frac{supp(\vec{B} \Rightarrow r(x,y))}{\#(x,y) : \exists z_1, ..., z_m : \vec{B}} \tag{21}$$

- **Positive-Only Learning**: in cases where the knowledge base lacks negative examples, Muggleton introduced a positive-only learning evaluation score for Inductive Logic Programming (ILP) [20] [21]. This approach uses randomly generated facts as negative evidence. The idea is that a good rule should cover many positive examples while minimizing the coverage of randomly generated examples. This approach promotes rules that are not overly general and encourages rule compression by using as few atoms as possible. The score is calculated as follows:

$$Score = log(P) - log\frac{R+1}{Rsize+2} - \frac{L}{P} \tag{22}$$

  Here, P is the number of known true facts covered (A in the figure), R is the number of randoms covered, $Rsize$ is the total number of randoms and L is the number of atoms in the hypothesis.

- **PCA Confidence** score: in the partial completeness assumption (PCA), the confidence score is normalized not by all facts, but by the set of facts known to be true, and the facts assumed to be false. This approach considers a more specific subset of facts for normalization. The PCA confidence score is calculated as follows:

$$pcaconf(\vec{B} \Rightarrow r(x,y)) := \frac{supp(\vec{B} \Rightarrow r(x,y))}{\#(x,y) : \exists z_1, ..., z_m, y' : \vec{B} \wedge r(x,y')} \tag{23}$$

2.6 Link Prediction

Link prediction in KGs is a task that aims to predict missing or potential relations (links) between entities based on the existing information in the KG. KGs represent

knowledge in a structured form using entities, relations (or predicates), and triples, such as RDF triples in the Semantic Web. Link prediction is a fundamental problem in KG completion and is used in various applications, including recommendation systems, semantic search, and KG construction.
The goal of link prediction is to predict the missing head (subject) or tail (object) for a triplet (s, p, o). Link prediction, in the general case, is often addressed with inductive techniques as discussed in this chapter, and in particular, KG embeddings and rule mining.
In summary, link prediction in KGs is a valuable task that addresses the incompleteness of structured data by predicting missing relationships between entities. It finds applications in various domains, and its success relies on the choice of appropriate scoring methods and evaluation metrics.

2.6.1 Evaluation Metrics for Link Prediction

Evaluation metrics for link prediction in KGs assess the quality and performance of link prediction algorithms by comparing their predicted links to the ground truth (actual links or missing links) in the KG. These metrics should help to understand how well different models are doing and make informed decisions. MR (Mean Rank) and MRR (Mean Reciprocal Rank) [1] are two common evaluation metrics used in link prediction tasks, especially in information retrieval and recommendation systems. These metrics assess the ability of a link prediction algorithm to rank predicted links.
MR measures the average rank of the correctly predicted links or items in a ranked list of predictions. For each entity (in our evaluation object of triples in test data sets) in the KG, the algorithm predicts a list of potential recommendations, and these predictions are ranked. MR is calculated by taking the average of the ranks of the correctly predicted links (equation 24). Lower values of MR indicate better performance.

$$MR = \frac{(Rank_1 + Rank_2 + ... + Rank_n)}{n} \tag{24}$$

MRR (Mean Reciprocal Rank) is similar to MR, but takes into account the reciprocal of the rank of the first correctly predicted link for each entity. MRR measures the average of the reciprocal ranks, giving more weight to higher-ranked correct predictions (equation 25). Higher values of MRR indicate better performance.

$$MRR = \frac{(1/Rank_1 + 1/Rank_2 + ... + 1/Rank_n)}{n} \tag{25}$$

In addition to MR and MRR, Hit@1, Hit@3, and Hit@10 metrics can be used to evaluate link prediction. Hit@k is the percentage of top k results that are correct. In other words, it assesses whether the ground truth is in the top k recommendation. Higher values of Hit@k indicate better performance, as they indicate that a larger proportion of test cases have their correct links within the top-k predictions.

$$Hit@k = \frac{(Number\ of\ Cases\ where\ the\ true\ missing\ link\ is\ in\ th\ top\ k\ predictions)}{n} \tag{26}$$

2.7 Summary of the Chapter

This chapter provides an overview of the basic concepts necessary to understand the problem addressed in the thesis. It starts with an introduction and terminology of KG concepts and the notion of inductive knowledge, and categorizes different types of inductive knowledge. The chapter then discusses different models for embedding KGs and goes on to discuss symbolic learning. Finally, it discusses the task of link prediction and the evaluation metrics used to evaluate the models used in link prediction.

Chapter 3

Related Work

In this chapter, we provide an overview of the relevant literature related to our thesis "Evaluating Hybrid AI for Prediction over lung cancer KGs". The research presented in this thesis involves a fusion of AI techniques for predictive analysis of complex KGs, with a particular focus on lung cancer KG. We investigate state-of-the-art methods for KG completion, drawing on insights from recent advances in the field.

3.1 Knowledge Graph Completion Methods

KGs are increasingly recognized as a valuable tool for structuring and organizing information in diverse domains, including healthcare. Particularly when dealing with disease-specific data, such as lung cancer KG, it is essential that these KGs are completed quickly and thoroughly.
The task of KG completion involves predicting missing relations or edges in the graph, contributing to a comprehensive and accurate representation of the underlying domain.
Experimental Study: Realistic Re-evaluation of KG Completion Methods [1] has focused on this evaluation task.This pivotal paper presents a thorough examination of KG completion methods, emphasizing the need for a more realistic evaluation setup. The paper explores the complexities of link prediction by leveraging data redundancy stemming from semantic duplication, correlation, data incompleteness, and Cartesian product relations.
It highlights the limitations and challenges of current completion techniques and presents innovative approaches to improve prediction performance. Although the research focuses primarily on general KGs, its contributions have significant implications for our thesis. We leverage the methodologies and evaluation criteria proposed

in this study to assess the performance of hybrid AI models in predicting missing links within lung cancer KG.
In this Master thesis, different AI models of self-supervised numeric inductive knowledge (embeddings models) and self-supervised symbolic inductive knowledge (rule mining models) are evaluated based on the metrics and methods that are used in paper [1] over lung cancer KGs.
A lung cancer KG stores facts in the form of triples denoted as $(subject, property, object)$ or $(tail, relation, head)$, represented as (s, p, o) or (h, r, t), e.g., $(patient1, sex, male)$. Although KGs are extensive, they remain incomplete in most instances, which could limit their practical use. Several approaches have been suggested to address this challenges, and one of them is automatic completion of a lung cancer KG. Previous research has classified existing methods in this prominent area into two main categories [22]: one relies on latent feature models, or embedding models, such as TransE [6], ConvE [10], and a variety of other approaches [7]. The other is dependent on observed feature models that take advantage of the observable characteristics of a KG. For instance, rule mining systems [15] and ranking methods based on mined rules are examples of this category. Authors in [1] evaluate the impact of reverse relations, data redundancy, and Cartesian product relations on KGs in FB15k [5], WN18 [6], and YAGO3-10 [27].
In the paper [1], it was unclear how data schema can affect the evaluation of the link prediction task. Therefore, this study also evaluates different data schemas on the lung cancer KGs. To conclude, this paper [1] makes the following contributions:

3.1.1 Impact of Data Redundancy

The analysis in paper [1] shows that the reverse, duplicate and symmetric relations led to a substantial over estimation of the model's accuracy.

3.1.2 Impact of Cartesian Relations

The research in paper [1] identifies the existence of Cartesian product relations problem, makes previous performance measures of models unrealistic.

3.2 Summary of the Chapter

This chapter reviewed some work and KGs related to this thesis. Additionally, it reviews state-of-the-art methods in KG completion.

Chapter 4

Our Approach and Proposed Solution

This chapter provides an in-depth exploration of our practical application of the approach, the developmental methodology employed for creating our framework, and its design.

4.1 Problem Statement

The proposed master thesis outlines a comprehensive approach for evaluating the performance of various embedding and symbolic models in link predicting over both base and preprocessed KGs. Several challenges and potential issues arise in the implementation of the described approach. The primary problems are identified as follows:

- **Model Diversity**: Incorporating a variety of embedding models and symbolic model introduces the challenge of ensuring compatibility and comparability among diverse approaches. The implementation needs to carefully manage the integration and comparison of these models within the evaluation framework.
- **Evaluation Metric Design**: Defining appropriate metrics for assessing the performance of the models in link prediction is critical. The challenge is to select or design metrics that capture the nuances of predictive accuracy, taking into account the characteristics of the KGs.
- **Systematic Procedure**: The proposed approach emphasizes systematic evaluation. The challenge lies in designing and implementing a step-by-step proce-

dure that guarantees consistency and reliability in the assessment of different models over both base and preprocessed KGs.

- **Scalability**: The approach mentions evaluation over both base and preprocessed KGs. The preprocessing steps need to be explicitly addressed to ensure the reliability of the evaluation results.

By addressing these challenges, the implemented approach aims to contribute valuable insights into the comparative performance of embedding and symbolic models in link prediction, thereby advancing the field of the KG analysis.

4.2 Proposed Approach

In this master thesis, we present an approach to evaluate the performance of different embedding models and symbolic model for predicting over base and preprocessed KG graph. To facilitate this evaluation, we propose a black box evaluation framework (Figure 4.1) that takes a KG as input and provides an evaluation of the different models as output over base and preprocessed KG.
In our approach, we evaluate link prediction that predicts the object part (right side) of triples (s, p, o), because in our KG, only the object part of triples had predictive information. The approach involves several key steps to ensure a systematic evaluation. Figure 4.2 represents the movement or transfer of data from one part of a process or system to another.In the following are key elements used to represent data flow in flowcharts:

- **Input**: The approach takes as an input two subsets of a lung cancer KG.
- **Arrows**: Arrows in the flowchart indicate data flow direction, moving from input to processing and then to output symbols. The use of two different arrow colors represents the comparison and evaluation of predictions between base and preprocessed data. Orange arrows signify the evaluation of base data, while blue arrows represent the evaluation of preprocessed data.
- **Processes or Functions**: These represent the operations and framework that are performed on the data.
- **Output**: The approach generates evaluation metrics for comparing methods and for both the base and preprocessed data.

Figure 4.1: The illustration of our approach that defines as a Black Box to evaluate hybrid AI for prediction over lung cancer KG

All codes, experiment scripts, datasets, and results are in a public repository https://github.com/SDM-TIB/SaharSafaei_Thesis. It will help ensure the reproducibility of the results reported in this Master thesis.

4.2.1 Methodology

The following sections describe in detail the processes and steps of the proposed methodology.

1.1: Data Preprocessing

The data preprocessing in Figure 4.3 includes the following step to apply data redundancy and remove CVT nodes:

1.1.1 Delete triples that describe a type of entities.

1.1.2 Delete triples that describe data or the name (or label) of entities.

1.1.3 Find duplicate entities and use a unique identifier for them.

1.1.4 Find Mediator (CVT) nodes and combine their relations together.

1.1.5 Identify the object that can be categorized (classified) and replace its value with its category class.

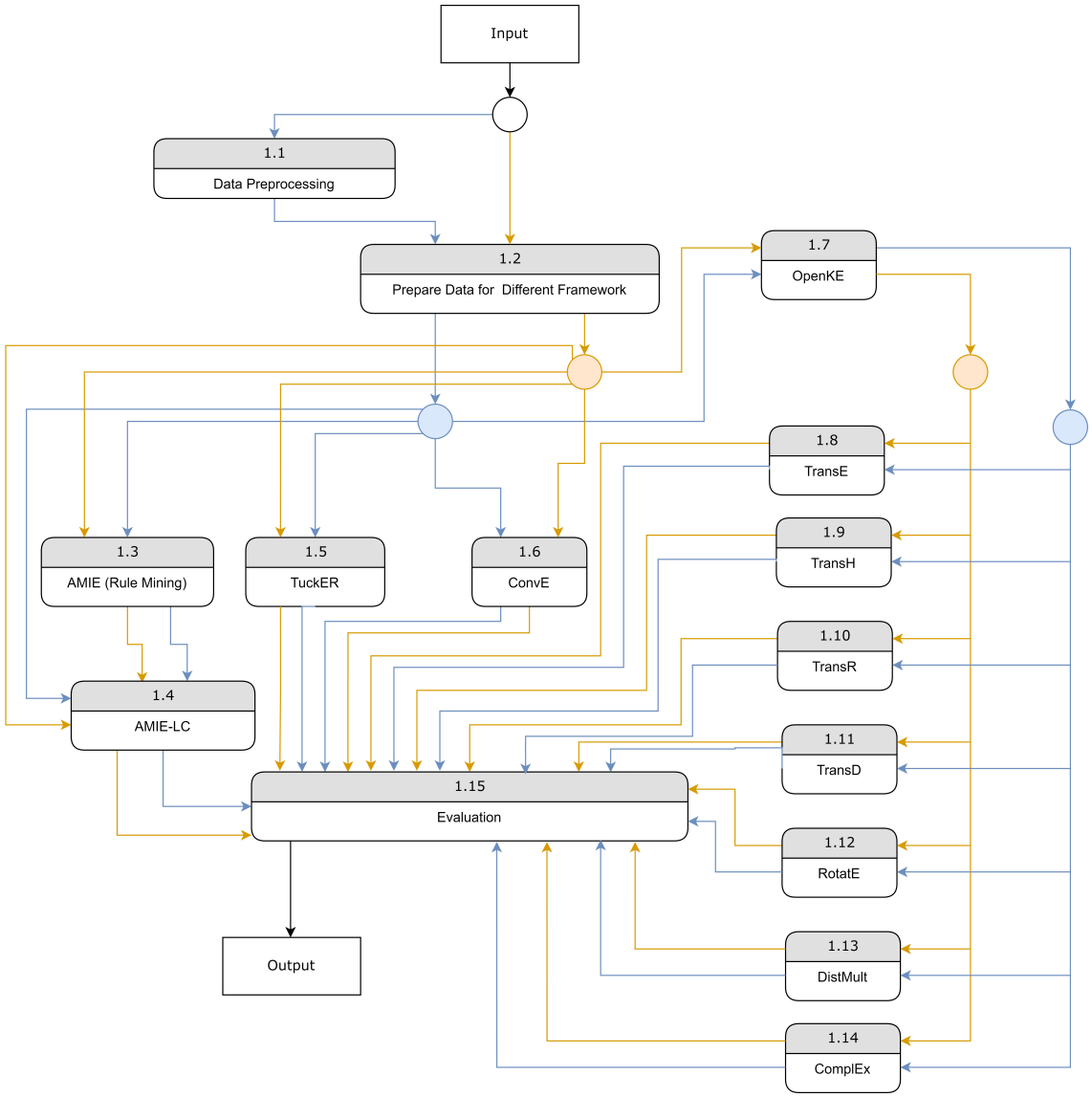


Figure 4.2: The workflow visualization of the approach. The workflow uses two different arrow colors. Orange arrows signify the workflow of base data, while blue arrows represent the workflow of preprocessed data with new data schema.

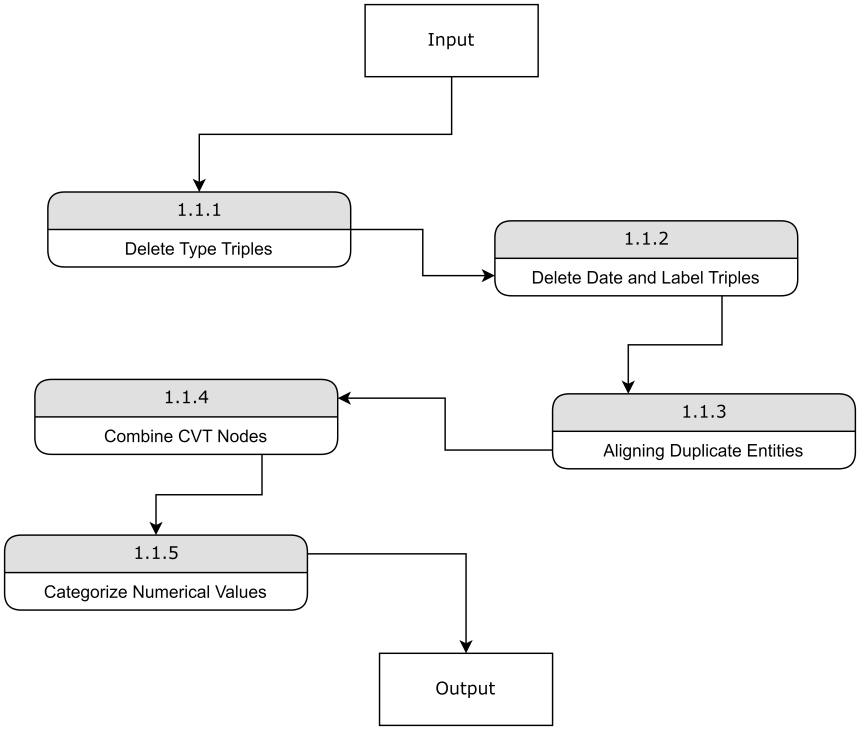


Figure 4.3: The workflow visualization of data preprocessing

1.2: Prepare Data for Different Frameworks

Different frameworks need different formats of data. In this step, we prepare data for each framework that we use in next steps. We also generate various test data triples for evaluation based on relation types (1-to-1, 1-to-n, n-to-1 and n-to-n). Figure 4.4 illustrates this step.
In this procedure, we initially divide our data into training, validation, and test sets to ensure uniform data distribution across models for a consistent evaluation. Subsequently, we create distinct test datasets categorized by their relation types. Afterward, we tailor the data according to the specific requirements of each AI model under evaluation.

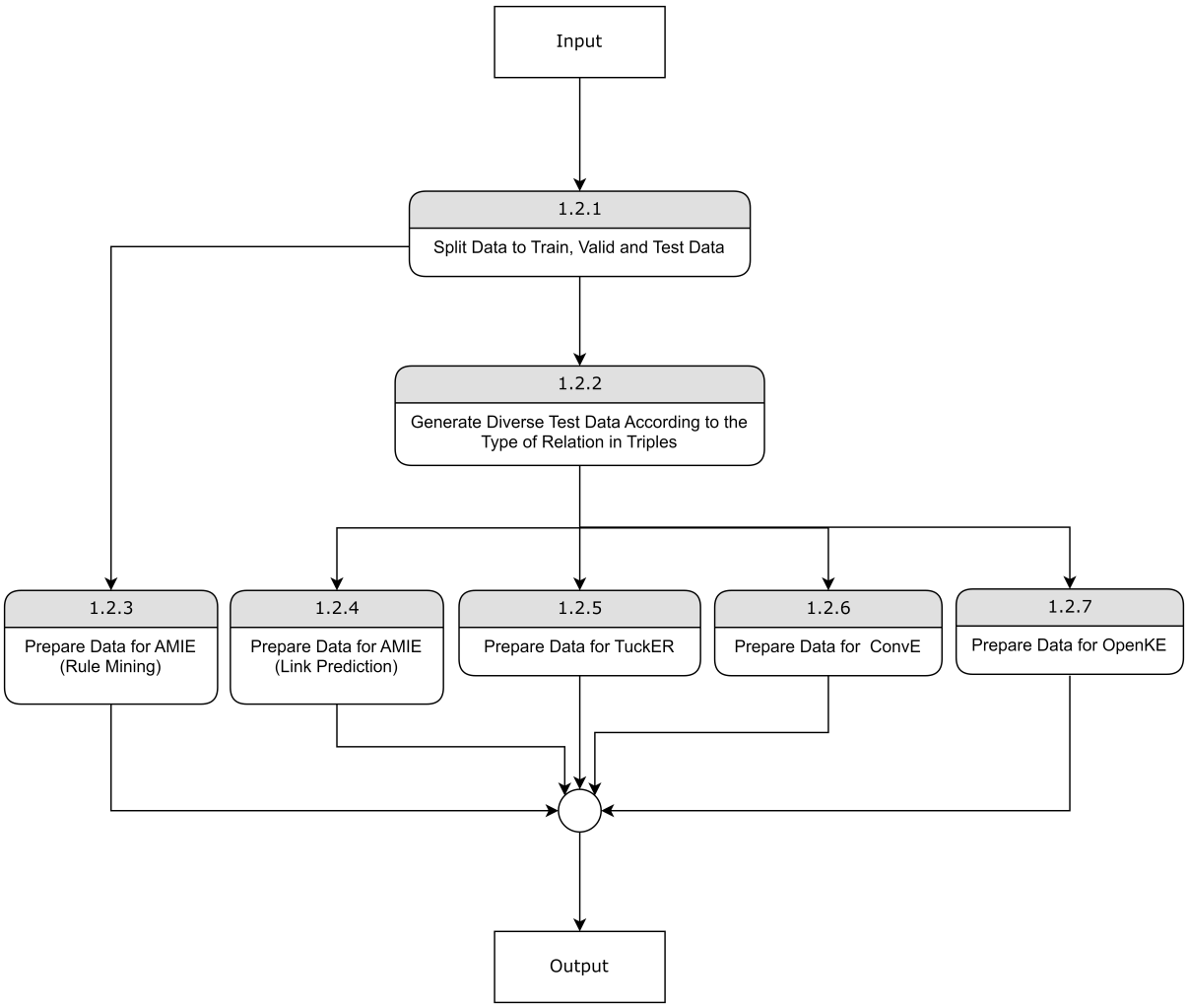


Figure 4.4: The workflow visualization of prepare data for different framework

1.3: AMIE (Rule Mining)

This process extracts the rules by running AMIE[4] on the training data. Subsequently, in this process the format of extracted rules are prepared for link prediction based on AMIE rules as shown in Figure 4.5.

[4]Source: https://github.com/dig-team/amie

1.4: AMIE-LC

Figure 4.6 shows how we use AMIE to evaluate right link prediction.

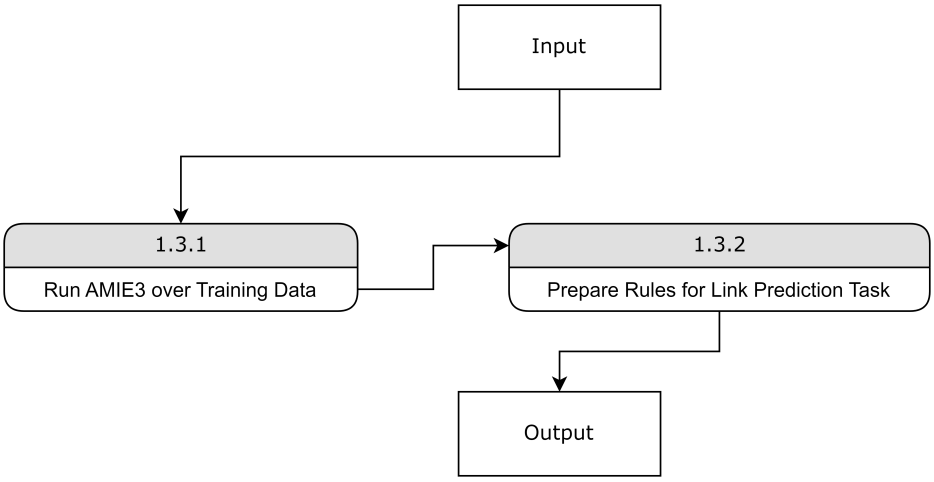


Figure 4.5: The workflow visualization of AMIE (Rule Mining) process

To incorporate symbolic learning for link prediction in a KG, we develop a link prediction function (**AMIE-LC**) that predict the object of a triple based on rules mined from AMIE and their associated PCA confidence metrics. The performance of the function is evaluated using test data from the KG. Code 4.1 shows the pseudocode for this function.
In enhancing our algorithm, our initial assumption involves considering objects of relations that appear at least once in the object part of relations within the KG. Additionally, we narrow down our focus to utilize rules from the mined set where the body of the rule shares the same relation as the relation in the test triple.

Listing 4.1: Pseudocode of AMIE-LC

```
Start
Read the list "train_data" from the file
Read the list "test_data" from the file
Read the list "rules" from the file
Initialize an empty dictionary named "dic_rel2objects"
Initialize a variable named "mr", "mrr", "hit1", "hit3", "hit10" with 0
Initialize a variable named "n" with length("test_data")
```

```
for each element "train" in the list "train_data":
..Split "train" into "subject", "relation", "object"
..if "relation" is not in the keys of "dic_rel2objects":
....Add a new key-value pair to "dic_rel2objects" where "relation" is the
    ↪ key and list("object") is the value
..Else:
....Add a new value "object" to the list "dic_rel2objects" where "relation"
    ↪  is the key
..End if
End for
for each element "test" in the list "test_data":
..Split "test" into "subject", "relation", "object"
..Filter "rules" into the list "filterd_rules" where the relation of Body is
    ↪  same as "relation"
.."filterd_rules"="filterd_rules".orderBy(PCA_Confidence).desc()
..Initialize an empty list named "predictioned_tails"
..Initialize a variable named "isFound" with false
..Initialize a variable named "rank" with 0
..for each element "rule" in the list "filterd_rules":
....for each element "tail" in the list "dic_rel2objects[relation]":
......if Body part of rule cover the "subject" and "tail":
........"rank"="rank"+1
........Append "tail" to list "predictioned_tails"
........if "tail"=="object":
..........."isFound"=true
..........Break
........End if
......End if
....End for
....if "isFound":
......Break
....End if
..End for
..if not("isFound"):
...."rank"=length("dic_rel2objects[relation]")
..End if
.."mr"="mr"+"rank"
.."mrr"="mrr"+1/"rank"
..if "rank"=1:
...."hit1"="hit1"+1
..End if
```

```
..if "rank"<=3:
....."hit3"="hit3"+1
..End if
..if "rank"<=10:
....."hit10"="hit10"+1
..End if
End for
Display ("mr"/"n", "mrr"/"n", "hit1"/"n", "hit3"/"n", "hit10"/"n")
Stop
```

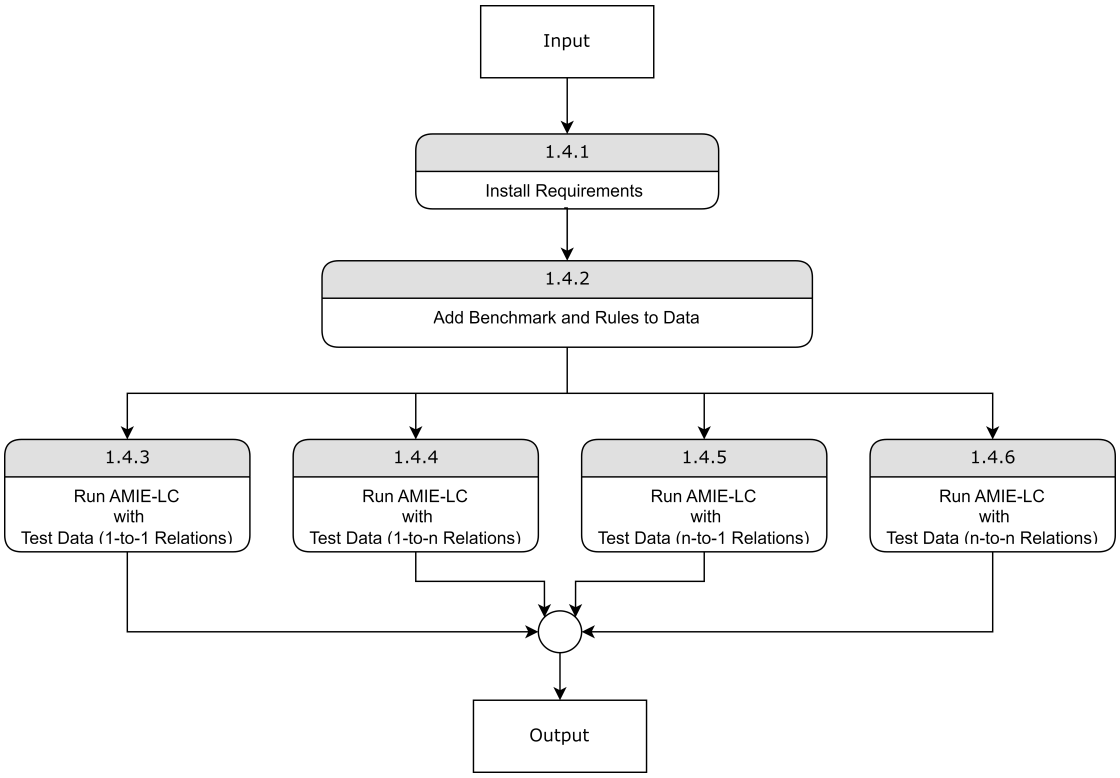


Figure 4.6: The workflow visualization of AMIE-LC process

1.5: TuckER

Figure 4.7 shows how we run the TuckER framework to evaluate right link prediction. First, we install the torch package in Python and then clone its repository [5].The next

[5]Source: https://github.com/ibalazevic/TuckER

step is to add our benchmarks to its data. Then we need to modify the 'main.py' file to customize the code for right link prediction. Finally, we can run TuckER separately on different types of relationships within the test data.

Figure 4.7: The workflow visualization of TuckER process showing how the TuckER model is executed through our benchmarks

1.6: ConvE

Figure 4.8 shows how we run the ConvE framework to evaluate right link prediction. First, we clone its repository [6], and then install the torch, scipy, spodernet [7], and bashmagic [8] packages in Python. The next step is to add our benchmarks to the data and then run 'preprocess.sh' to prepare the benchmarks that were entered into the file. Then we run ConvE over the benchmarks and save the model. Finally, we can separately run the model on different types of relationships within the test data.

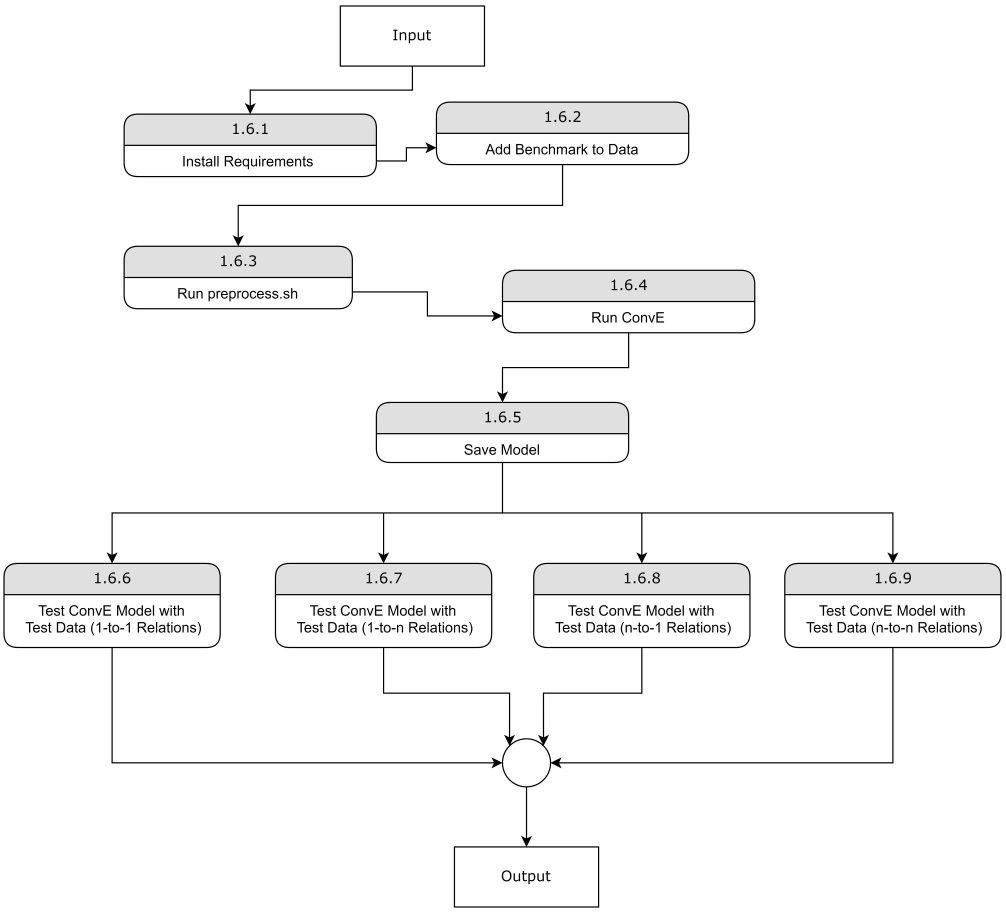


Figure 4.8: The workflow visualization of ConvE process showing how the ConvE model is executed through our benchmarks

[6]Source: https://github.com/TimDettmers/ConvE
[7]Source: https://github.com/TimDettmers/spoderne
[8]Source: https://github.com/TimDettmers/bashmagic

1.7: OpenKE

Figure 4.9 shows how we prepare OpenKE framework to prepare it for other models to evaluate the right link prediction. First, we install the torch package in Python and then clone its repository [9].Next is to add our benchmarks to the data and run "n-n.py" for each benchmark to prepare the data. Then we need to modify the "test.h" and "Tester.py" files to customize the code for the right link prediction. Then we run the "make.sh" file to connect Python to a shell script.

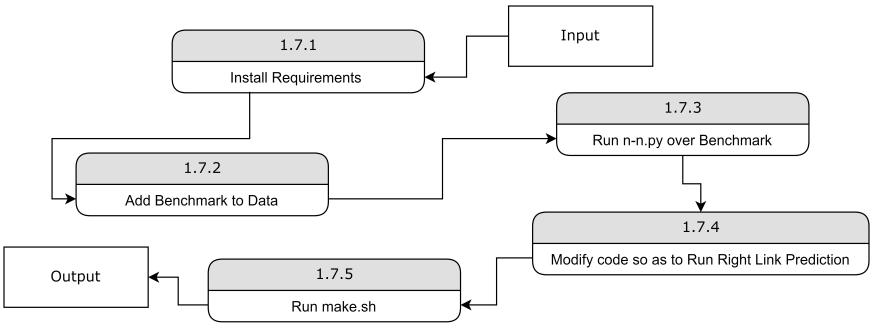


Figure 4.9: The workflow visualization of OpenKE process showing how we prepare OpenKE framework to run other models through our benchmarks

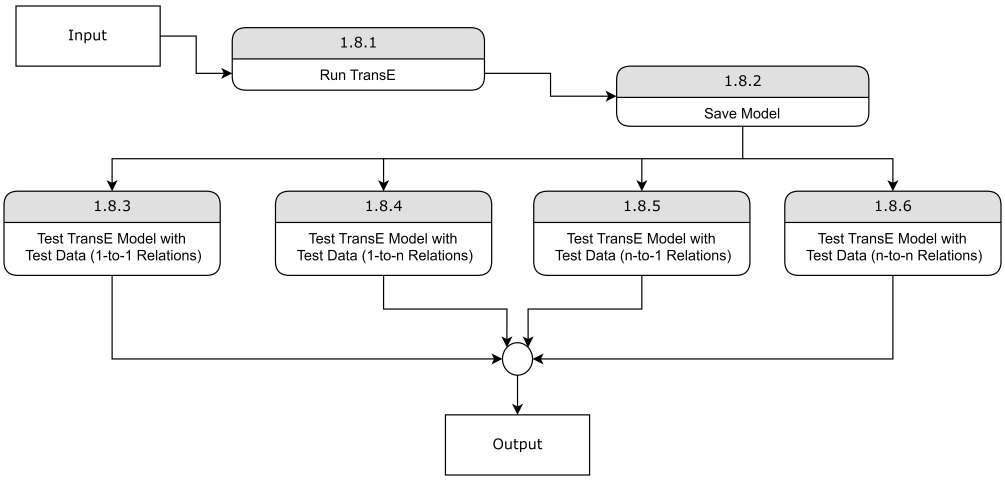


Figure 4.10: The workflow visualization of TransE process showing how the TransE model is executed through our benchmarks

[9]Source: https://github.com/thunlp/OpenKE)

1.8: TransE

Figure 4.10 shows how we run TransE to evaluate right link prediction over different type of relations. First, we train the TransE model. Then we save the model and evaluate it using different types of relations for each test data in our benchmarks.

1.9: TransH

Figure 4.11 shows how we run TransH to evaluate right link prediction over different type of relations. First, we train the TransH model. Then we save the model and evaluate it using different types of relations for each test data in our benchmarks.

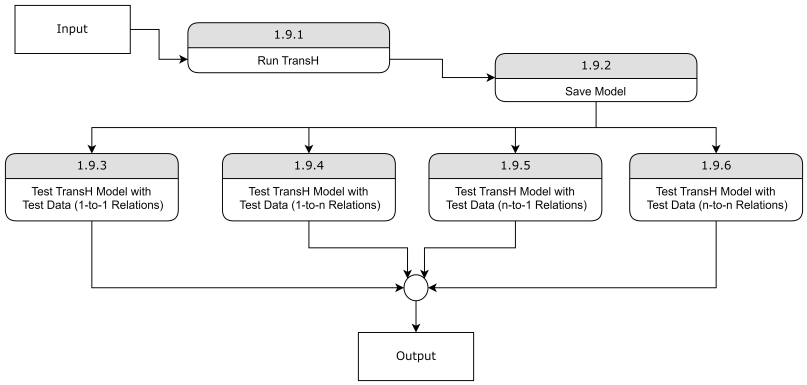


Figure 4.11: The workflow visualization of TransH process showing how the TransH model is executed through our benchmarks

1.10: TransR

Figure 4.12 shows how we run TransR to evaluate right link prediction over different type of relations. First, we train the TransR model. Then we save the model and evaluate it using different types of relations for each test data in our benchmarks.

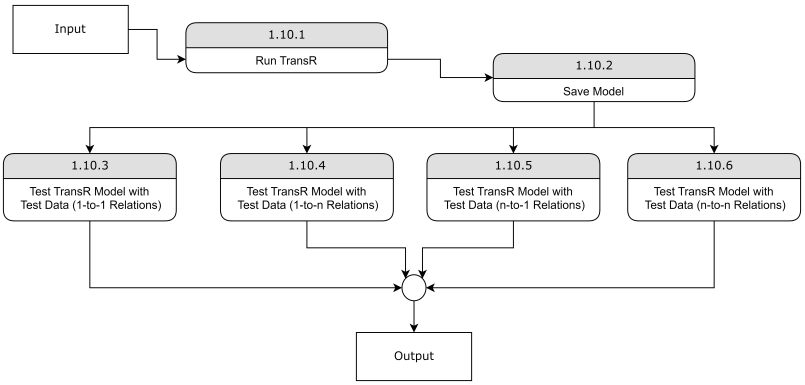


Figure 4.12: The workflow visualization of TransR process showing how the TransR model is executed through our benchmarks

1.11: TransD

Figure 4.13 shows how we run TransD to evaluate right link prediction over different type of relations. First, we train the TransD model. Then we save the model and evaluate it using different types of relations for each test data in our benchmarks.

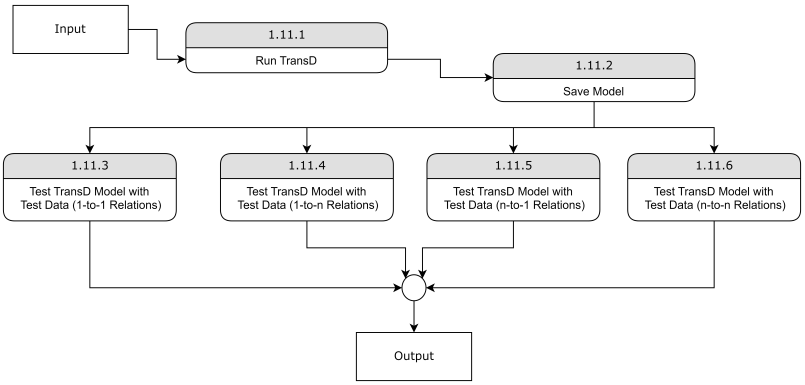


Figure 4.13: The workflow visualization of TransD process showing how the TransD model is executed through our benchmarks

1.12: RotatE

Figure 4.14 shows how we run RotatE to evaluate right link prediction over different type of relations. First, we train the RotatE model. Then we save the model and evaluate it using different types of relations for each test data in our benchmarks.

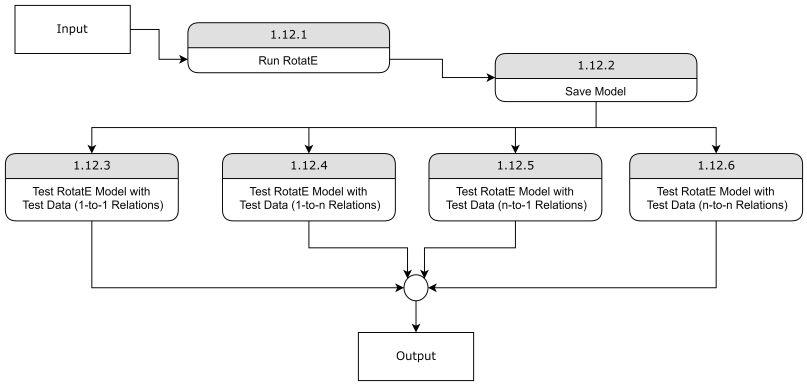


Figure 4.14: The workflow visualization of RotatE process showing how the RotatE model is executed through our benchmarks

1.13: DistMult

Figure 4.15 shows how we run DistMult to evaluate right link prediction over different type of relations. First, we train the DistMult model. Then we save the model and evaluate it using different types of relations for each test data in our benchmarks.

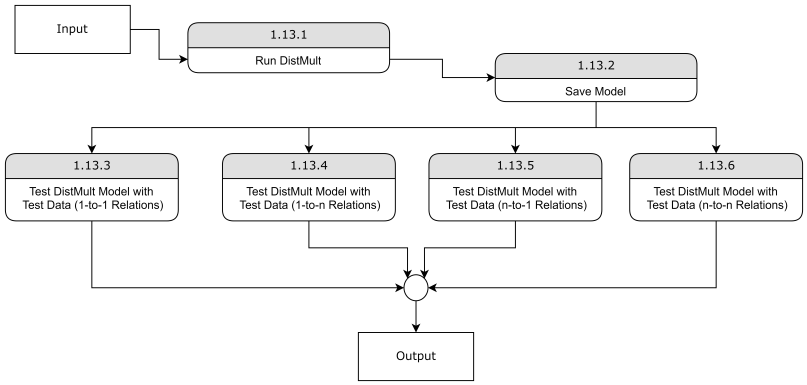


Figure 4.15: The workflow visualization of DistMult process showing how the DistMult model is executed through our benchmarks

1.14: ComplEx

Figure 4.16 shows how we run ComplEx to evaluate right link prediction over different type of relations. First, we train the ComplEx model. Then we save the model and evaluate it using different types of relations for each test data in our benchmarks.

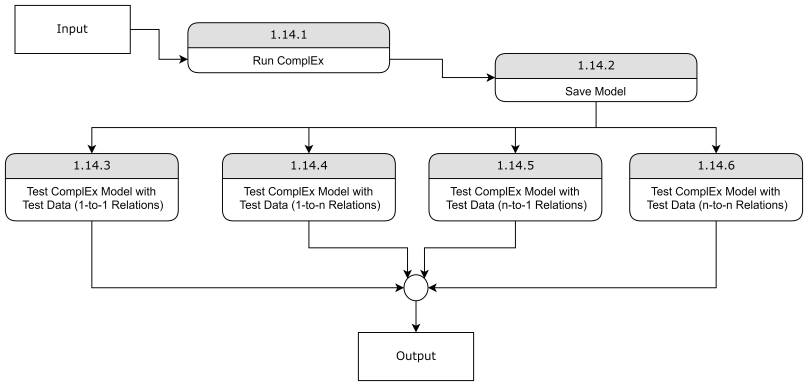


Figure 4.16: The workflow visualization of ComplEx process showing how the ComplEx model is executed through our benchmarks

1.15: Evaluation

This process receives as input the evaluation metrics (MR, MRR, Hit@1, Hit@3, Hit@10) of the models over the base and preprocessed KGs that were run in the previous steps. In this step, we can now compare the evaluation of the models over different types of relations after and before preprocessing.

4.3 Summary of the Chapter

The chapter begins with a definition of the problem and then a visual representation of the approach in the form of a black box. A class diagram is then created to provide a structural overview. The step-by-step process of implementing the approach is then explained to illustrate how we use different frameworks to evaluate different models against base and preprocessed benchmarks. It also illustrates how we evaluate models using different relation types.

Chapter 5

Experimental Evaluation

The experimental section of this master thesis focuses on evaluating the performance of a hybrid AI system for link prediction tasks over lung cancer KG. In this section, we present the experimental setup, data collection and preprocessing steps, evaluation metrics, and the result of the experimental procedure. The experiment was designed to address the following research questions:
Q1) How efficient is our proposed AMIE-LC approach compared to numerical models in the KG?
Q2) What is the impact of the extension of the KG on the task of link prediction, in particular in our scenario of the integration of a Biomarker KG with Relapse KG?
Q3) How can the performance of the link prediction task be enhanced by changing the data schema within KG?
Q4) Do the models developed for the link prediction task demonstrate varied evaluations based on different types of relations within KG?

5.1 Experimental Setup

Our experimental divide to two category of embeddings and symbolic models.
The embedding model experiments were carried out in a cloud-based computing environment to harness the computational resources required to train complex AI models and preprocess large-scale KG data. Google Colab, a cloud-based Jupyter notebook platform provided by Google, was utilized for its ease of access and flexibility in handling both Python-based coding and data analysis tasks. Google Colab offers several advantages, including access to GPU and TPU resources, collaborative features, and seamless integration with popular AI libraries such as TensorFlow and

PyTorch. The use of Google Colab ensured that our experiments were reproducible, scalable, and efficient. We used Google Colab Pro with GPU runtime and high RAM for running the models and evaluation.
The experiments used source code from a variety of places, including the OpenKE (https://github.com/thunlp/OpenKE) repository that contains implementations of OpenKE which contains implementations of TransE [6], TransH [32], TransR [19], TransD [17], RotatE [28], ComplEx [29], and DistMult [33], as well as source code from the releases of ConvE [10] (https://github.com/TimDettmers/ConvE), TuckER [3] (https://github.com/ibalazevic/TuckER).
The different models used in our experiments have different hyperparameters. We used the same hyperparameter settings for the Biomarker and Relapse benchmarks. The specific hyperparameter values used for each experiment are provided in Tables 5.1, 5.2, 5.3, and 5.4.

Framework	train_times	alpha
OpenKE	500	1.0

Table 5.1: Hyperparameters for OpenKE

Model	batch_size	sampling	dim	dim_e & dim_r	margin	epsilon	loss	opt_method
TransE	100	normal	20	-	-	-	MarginLoss	-
TransH	100	normal	20	-	-	-	MarginLoss	-
TransR	100	normal	-	20	-	-	MarginLoss	-
TransD	100	normal	-	20	-	-	MarginLoss	-
RotatE	2000	cross	20	-	6.0	2.0	SigmoidLoss	adam
DistMult	100	normal	20	-	-	-	SoftplusLoss	adagrad
ComplEx	100	normal	20	-	-	-	SoftplusLoss	adagrad

Table 5.2: Hyperparameters for models used OpenKE

Model	epochs	other parameter
ConvE	6	default

Table 5.3: Hyperparameters for ConvE

Another category of our experiments are symbolic models that are rule-based systems. We use AMIE to extract the rules. AMIE rules were generated by applying

Model	num_ iterations	batch_ size	lr	dr	edim & rdim	input_ dropout	hidden_ dropout1	hidden_ dropout2	label_ smoothing
TuckER	100	100	0.0005	1.0	20	0.3	0.4	0.5	0.1

Table 5.4: Hyperparameters for TuckER

the AMIE+ (https://bit.ly/2Vq2OIB) code released by the authors of [15].
We run AMIE on an Intel-based machine with an Intel Corei7 v Pro 9th Gen processor running at 2.60GHz, 32GB of RAM to extract the rules on KG.
Afterwards, we use the AMIE output (rules) as input for the link prediction process as described in Chapter 4.2.1. We develop this approach in Python 3 and run it on an Intel Corei7 v Pro 9th Gen processor running at 2.60GHz, 32GB RAM. To speed up the approach, we use Python's multiprocessing library with 6 parallel processes.
It is appropriate to mention that before developing this approach we tried to reproduce the AMIE part of paper[1] over cancer KG, but as the result was not satisfying, we reviewed the code and understood that the code was developed based on other benchmarks (FB15k, WN18, FB15k-237, WN18RR, and YAGO3-10-DR) and was not compatible with our benchmark. So, we developed a new approach to apply rules from AMIE that was compatible with our problem.

5.2 Benchmarks

There are two different benchmarks used in our experimental evaluation. Both benchmarks are a subset of lung cancer KG [30], which is about lung cancer patients, as explained in Section 2.2. The lung cancer KGs are created by SDM-RDFizer [10]. SDM-RDFizer is an RML that is used to map relational data to the KG, as we explain in section 2.1.6.

Dataset	#entities	#relations	#train	#valid	#test
BaseBiomarker	78,486	50	709,426	39,413	39,413
Biomarker	17,635	33	274,076	15,226	15,227
BaseRelpase	102,013	74	908,607	50,478	50,479
Relapse	17,682	55	360,774	20,043	20,043

Table 5.5: Statistics of evaluation datasets

[10]Source: https://github.com/SDM-TIB/SDM-RDFizer

Biomarker : It is a subset of lung cancer KG [30] which contains 33 different relations, 17,635 nodes and 304,529 edges. This KG describes the family history, treatment, smoking habit, gender, carcinogen, stage, age, region, occupation, and cancer history of the patient with lung cancer.
In our experiment, we use BaseBiomarker and Biomarker datasets. By BaseBiomarker we mean the KG without any modification in the data schema, and by Biomarker we mean the Biomarker benchmark with modified data schema. For our evaluation, these triples were randomly divided into training, validation, and test sets, and Table 5.5 shows the statistics.

Relapse : It includes the Biomarker benchmark and the other subset of lung cancer KG [30] on disease recurrence in a patient who has previously been treated for lung cancer and has achieved remission or a period of no detectable cancer.This KG contains 55 different relations, 17,682 nodes and 400,860 edges.
In our experiment, we use BaseRelapse and Relapse datasets. By BaseRelapse we mean KG without any modification in the data schema, and by Relapse we mean the Relapse benchmark with modified data schema.
For our evaluation, these triples were randomly divided into training, validation, and test sets, and Table 5.5 shows the statistics.

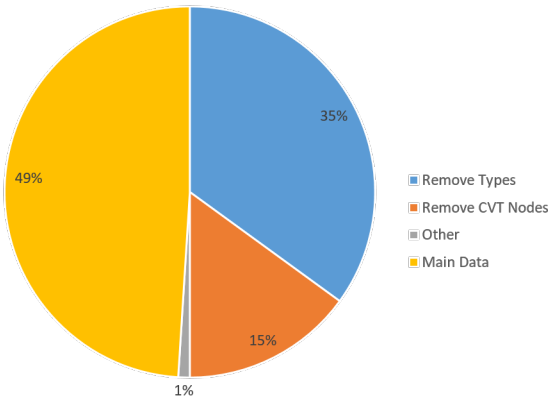


Figure 5.1: Preprocessing portion to change data schema in KG

5.3 PreProcessing Steps

In the following steps, the data schema within the KG are modified to assess the impact of data schema on the evaluation of hybrid AI for link prediction.
Step 1) Since the type of entities and the type of relations do not provide any relevant information, all triples with type relations are deleted in order to reduce complexity.
Step 2) As the names or labels of entities and relations do not provide useful information, all triples with names of relations are also removed to reduce complexity.
Step 3) There are also entities in the KG that are identified as the same entities with the `owl:sameAs` in property of triples. Therefore, we replace all entities with only one of these values in the triple so that we have more qualitative data, and by this step we also remove duplicates.

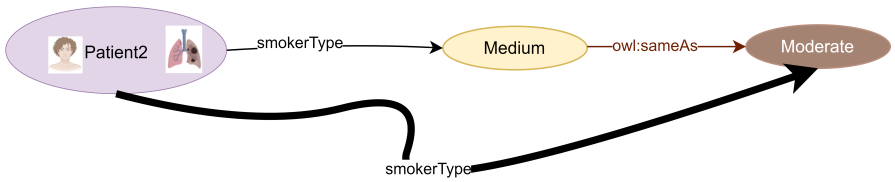


Figure 5.2: Example of duplicates

Step 4) Mediator nodes, also called CVT nodes, are used in both benchmarks to represent a multivalued composite attribute. For example, in Figure 5.3 `Patient1_4` is a CVT node that describes the family history of the patient. There may also be another patient in KG who also has the same family history, but with a different CVT node. Due to the same feature of these mediator nodes, we remove these CVT nodes and connect directly to the patient with the feature of the CVT nodes. In Figure 5.4 we also concatenate `Sister` and `Breast` entities together as an entity so it can describe `FamilyType` and `CancerType` together. In Table 5.6 the relations that are replaced with CVT nodes and its relations are illustrated.

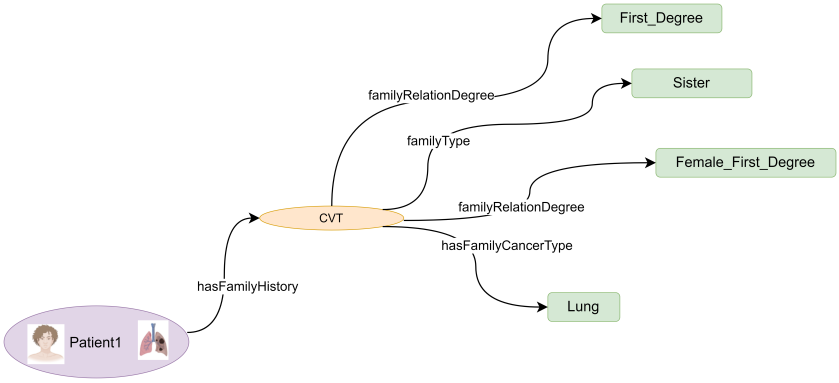


Figure 5.3: Example of mediator (CVT) node in lung cancer KG

Step 5) In the KG, objects are continuous or numerical values. This type of object does not provide meaningful data because some of these values only occur for one patient. Nevertheless, we classify these values and define a range for each category of classes to obtain more informative data. For example, we have classified age into two categories: old (older than 50) and young (younger than or equal to 50).

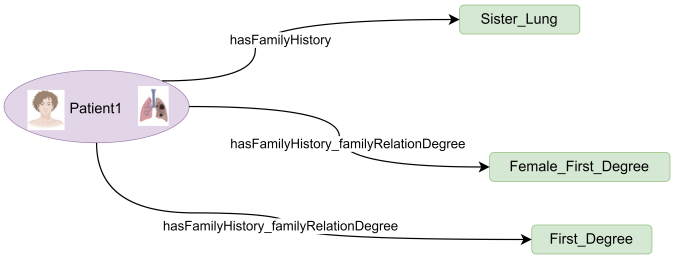


Figure 5.4: Illustration of the removal of the CVT node in the lung cancer KG and the establishment of connections between two entities, namely familyType and has familyCancerType, aimed at introducing more informative relations.

Old Relations	New Relation
hasFamilyHistory & (familyType , hasFamilyCancerType)	hasFamilyHistory
hasFamilyHistory & familyRelationDegree	hasFamilyHistory_familyRelationDegree
hasPTNMdiagnosisType & ptnmDiagnosisType	hasPTNMdiagnosisType_ptnmDiagnosisType
hasPTNMdiagnosisType & hasCarcinogenType	hasPTNMdiagnosisType_hasCarcinogenType
hasPTNMdiagnosisType & hasCarcinogenYears	hasPTNMdiagnosisType_hasCarcinogenYears
hasCarcinogen & ptnmDiagnosisType	hasCarcinogen_ptnmDiagnosisType
hasCarcinogen & hasCarcinogenType	hasCarcinogen_hasCarcinogenType
hasCarcinogen & hasCarcinogenYears	hasCarcinogen_hasCarcinogenYears
hasCarcinogen & hasCarcinogenCityContact	hasCarcinogen_hasCarcinogenCityContact
hasDiagnosis & hasDiagnosisStage	hasDiagnosis_hasDiagnosisStage
hasDiagnosis & has_SynchronousTumor_OrNot	hasDiagnosis_has_SynchronousTumor_OrNot

Table 5.6: Establishment of relations used to create new data schema in lung cancer KG

5.4 Evaluation Metrics

As we mentioned in Implementation part, we use MR, MRR, Hit@1, Hit@3, and Hit@10 (they are explained in section 2.6.1)to compare the performance of hybrid AI models over our KG.

5.5 Results

In this section, we present the results of our evaluation of numerical and symbolic models for the link prediction task over the lung cancer KG. We use 2 benchmarks of the lung cancer KG with base and modified data schema to compare these models. Tables 5.7 and 5.8 show the results of the right_link prediction on BaseBiomarker, Biomarker, BaseRelapse, and Relapse benchmarks after and before modifying the data schema for all compared models using the evaluation metrics explained in Section 5.4. In this KG, we only predicate the object part of the triples, because the subject part in the KG has just defined the patient and it is just an identifier for each patient, and it does not provide any valuable information.

Metrics	MR(↓)		MRR(↑)		Hits@1(↑)		Hits@3(↑)		Hits@10(↑)	
Model	BB	B	BB	B	BB	B	BB	B	BB	B
TransE	**704.58**	7.68	0.49	0.61	0.33	0.41	0.60	0.78	0.76	0.93
TransH	1,390.45	**6.56**	0.52	0.68	0.36	0.51	0.64	0.82	**0.79**	**0.94**
TransR	2,461.35	10.55	0.50	0.59	0.34	0.39	0.61	0.76	0.78	0.93
TransD	902.36	9.61	0.46	0.63	0.30	0.44	0.58	0.80	0.78	0.93
RotatE	6,888.29	1,125.23	0.01	0.01	0	0	0	0	0.02	0.02
DistMult	3,055.05	24.44	0.15	0.25	0.01	0.11	0.17	0.28	0.50	0.56
ComplEx	4,275.48	16.29	0.26	0.26	0.12	0.11	0.33	0.30	0.57	0.57
ConvE	3,207.29	43.08	0.47	0.59	0.25	0.42	0.67	0.73	0.78	0.88
TuckER	4,392.82	24.12	0.41	0.36	0.34	0.27	0.42	0.39	0.57	0.62
AMIE-LC	1,163.00	10.57	**0.64**	**0.80**	**0.57**	**0.71**	**0.68**	**0.89**	0.78	**0.94**

Table 5.7: Right_Link prediction result on BaseBiomarker (BB) and Biomarker (B). Better performance is indicated by a lower MR value, along with higher values for MRR and Hits@k.

Metrics	MR(↓)		MRR(↑)		Hits@1(↑)		Hits@3(↑)		Hits@10(↑)	
Model	BR	R	BR	R	BR	R	BR	R	BR	R
TransE	**967.76**	**6.35**	0.50	0.64	0.35	0.46	0.61	0.78	0.77	0.95
TransH	1,691.49	9.95	0.48	0.68	0.32	0.51	0.60	0.83	0.78	**0.96**
TransR	3,527.89	10.62	0.47	0.60	0.30	0.39	0.61	0.77	**0.79**	0.94
TransD	1,048.93	10.62	0.48	0.60	0.30	0.39	0.62	0.82	0.78	0.95
RotatE	9,616.76	1,138.62	0	0.01	0	0	0	0	0.01	0.02
DistMult	5,553.05	25.53	0.10	0.20	0.01	0.09	0.05	0.18	0.37	0.52
ComplEx	5,474.78	18.14	0.33	0.22	0.23	0.09	0.36	0.21	0.51	0.53
ConvE	4,161.42	24.89	0.39	**0.79**	0.26	**0.68**	0.396	**0.90**	0.74	**0.96**
TuckER	6,864.79	24.14	0.12	0.30	0.03	0.18	0.09	0.31	0.44	0.58
AMIE-LC	1,396.12	8.22	**0.66**	**0.79**	**0.59**	**0.68**	**0.71**	**0.90**	0.76	0.94

Table 5.8: Right_Link prediction result on BaseRelapse (BR) and Relapse (R). Better performance is indicated by a lower MR value, along with higher values for MRR and Hits@k.

The Tables 5.7 and 5.8 show that the RotatE method fails to make accurate predictions within the KG because it works with symmetrical relations and there are no symmetrical relations in our KG.

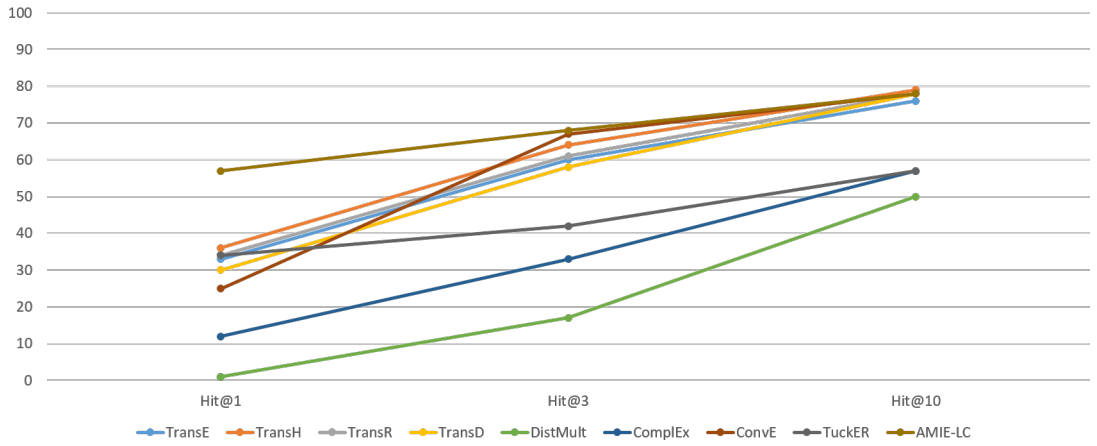


Figure 5.5: Evaluation models based on Hit@k(↑) metrics over BaseBiomarker. Better performance is indicated by a higher Hit@k value.

5.5.1 Analyze Different Models

Comparing the Hit@1 metric across the Figures 5.5, 5.7, 5.6 and 5.8, symbolic models such as AMIE outperform numerical models. In the Hit@3 metric, the neural model ConvE and the translational models (TransH, TransR, TransE and TransD) subsequently converge to a similar level of performance as AMIE, demonstrating comparable effectiveness.

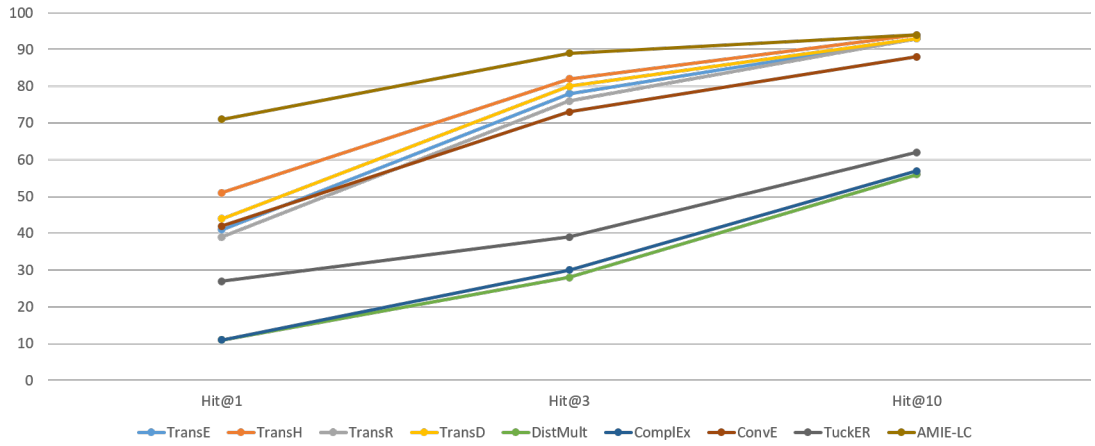


Figure 5.6: Evaluation models based on Hit@k(↑) metrics over Biomarker. Better performance is indicated by a higher Hit@k value.

Their performance becomes almost identical in the Hit@10 metric. All figures consistently show that tensor decomposition models, namely Distmult, ComplEx and TuckER, do not perform well lung cancer KG for the link prediction task.

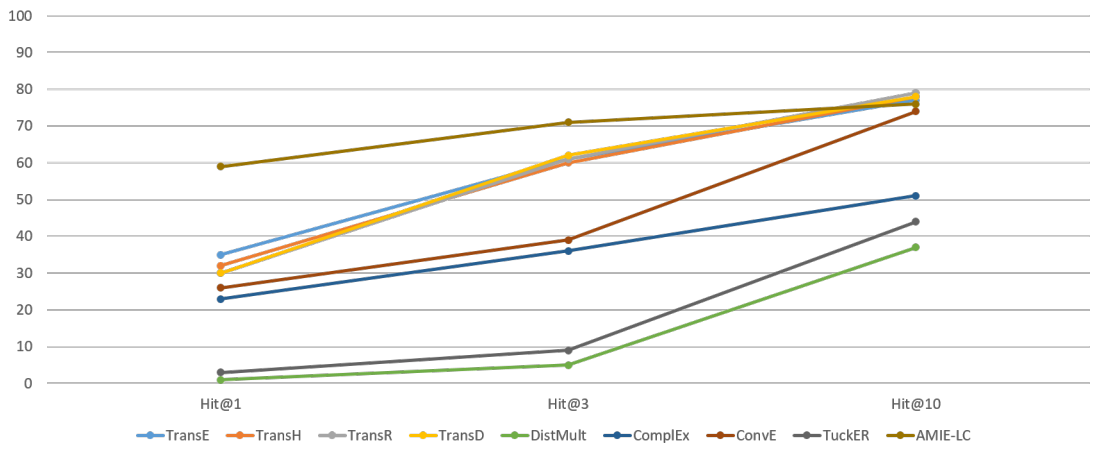


Figure 5.7: Evaluation models based on Hit@k(↑) metrics over BaseRelapse. Better performance is indicated by a higher Hit@k value.

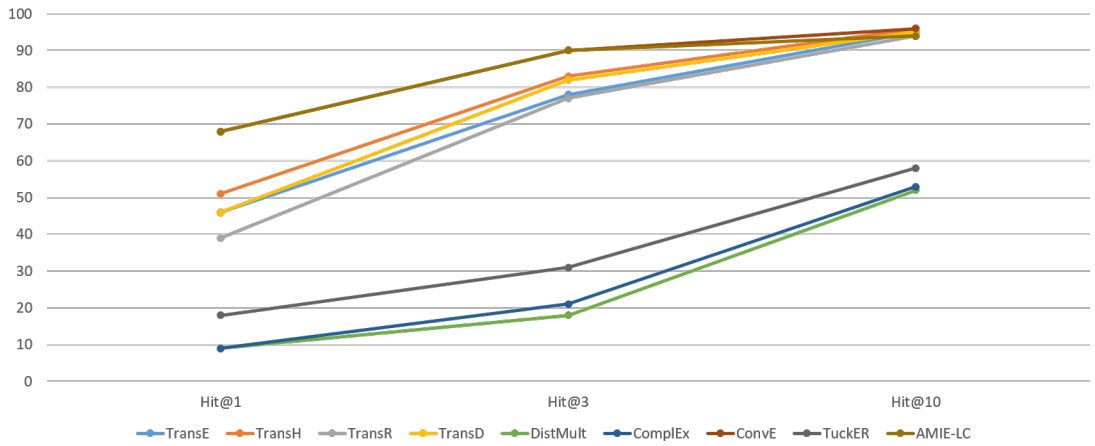


Figure 5.8: Evaluation models based on Hit@k(↑) metrics over Relapse. Better performance is indicated by a higher Hit@k value.

5.5.2 Analyze Performance of Different Models by Enhancing Data

Figures 5.9 and 5.10 show the different effects of integrating relapse data into the Biomarker Benchmark KG on both the base KG and KG with the modified data schema. In Figure 5.9, all models show superior performance when no relapse data

are integrated, indicating a lower performance when the data is expanded. However, Figure 5.10 shows that certain models show improved predictive ability with the integration of relapse data.
In particular, translational models (TransE and TransD), the tensor decomposition model (TuckER) and the symbolic model (AMIE) show improved performance when more data is integrated.

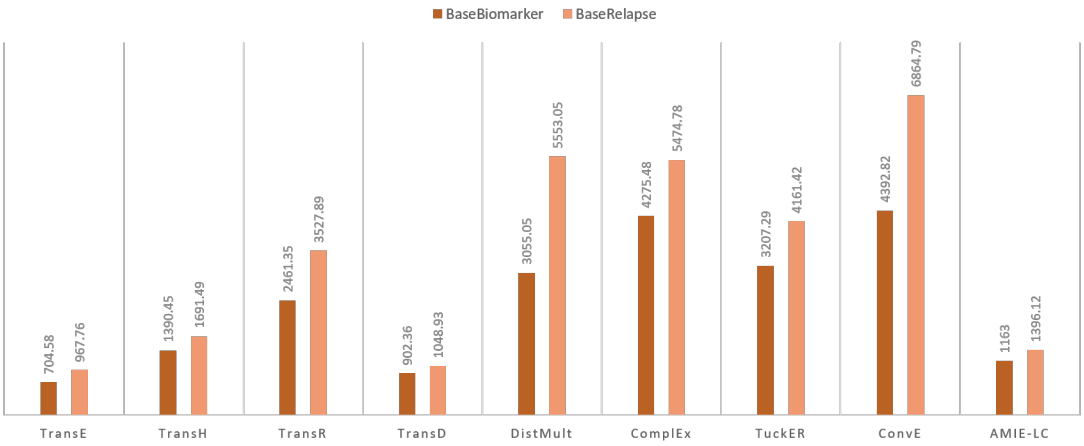


Figure 5.9: Compare MR(↓) between BaseBiomarker and BaseRelapse. Better performance is indicated by a lower MR value.

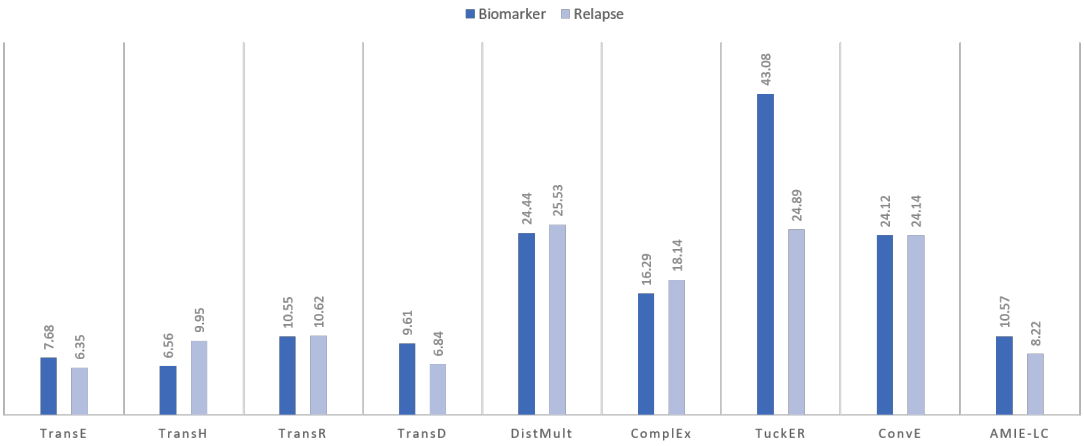


Figure 5.10: Compare MR(↓) between Biomarker and Relapse. Better performance is indicated by a lower MR value.

5.5.3 Analyze Performance with Different Data Schemas

As described in sections 4.2.1 and 5.2, we use both base benchmarks and benchmarks with modified data schema to assess the impact of models on the evaluation of various numerical and symbolic models. Figures 5.12 and 5.13 show that the application of new data schema improves the performance of both symbolic and numerical models. Figure 5.11 demonstrates that performance improvement is achieved through enhanced data with new data schema compared to the baseline benchmark. At baseline, we utilize data schema without any preprocessing.

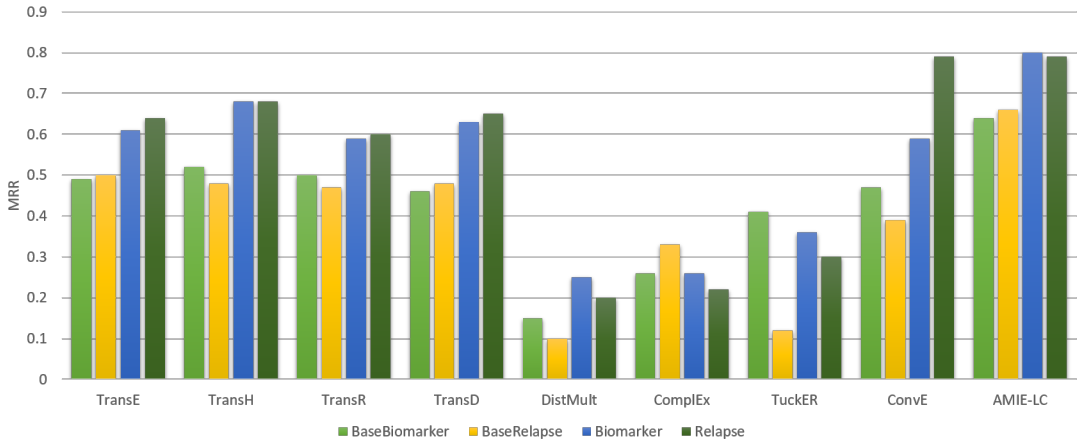


Figure 5.11: Compare MRR(↑) by enhancing data in BaseBiomarker and Biomarker. Better performance is indicated by a higher MRR value.

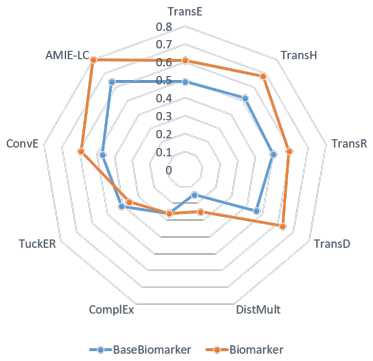


Figure 5.12: Evaluation models based on MRR(↑) metrics over Biomarker by modifying data schema. Better performance is indicated by a higher MRR value.

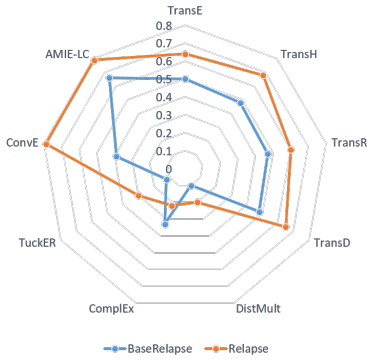
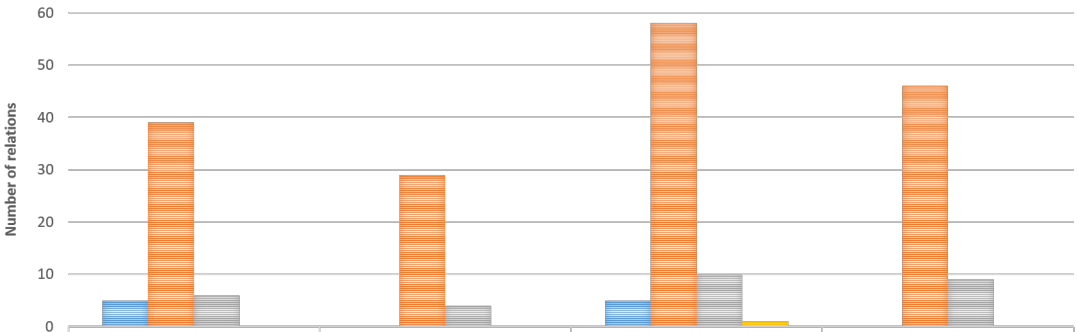


Figure 5.13: Evaluation models based on MRR(↑) metrics over Relapse by modifying data schema. Better performance is indicated by a higher MRR value.

5.5.4 Analyze Different Type of Relations

The Figure 5.14 shows the statistics of the types of relations used to evaluate the link prediction on the BaseBiomarker, Biomarker, BaseRelapse, and Relapse benchmarks.



	BaseBiomarker	Biomarker	BaseRelapse	Relapse
1-to-1	5	0	5	0
n-to-1	39	29	58	46
n-to-n	6	4	10	9
1-to-n	0	0	1	0

Figure 5.14: Number of relation types in different Benchmarks

The results shown in Figures 5.15 and 5.16 show the differences in performance depending on the type of relations. The new data schema in both the Biomarker and Relapse benchmarks improves the performance of the `n-to-1` relations, but there is a decrease in performance for the `n-to-n` relations types.
In particular, the removal of the relation types `1-to-1` and `1-to-n` due to the removal of CVT nodes in the new data schema contributes to the overall performance

improvement. The reason for this is that almost no prediction can be made for the relation types `1-to-1` and `1-to-n` due to the lack of reverse, symmetric and duplicate relations in the lung cancer KG.

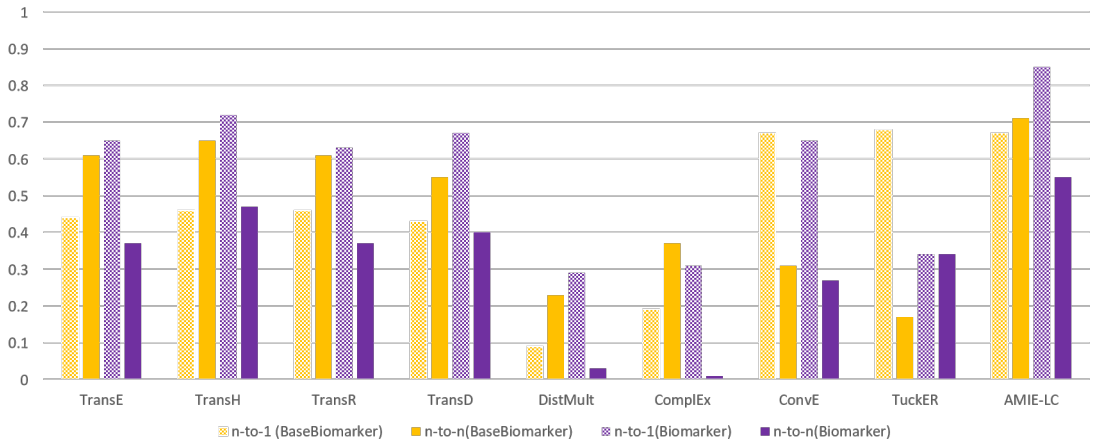


Figure 5.15: Evaluation MRR(↑) over Biomarker with different relation types. Better performance is indicated by a higher MRR value.

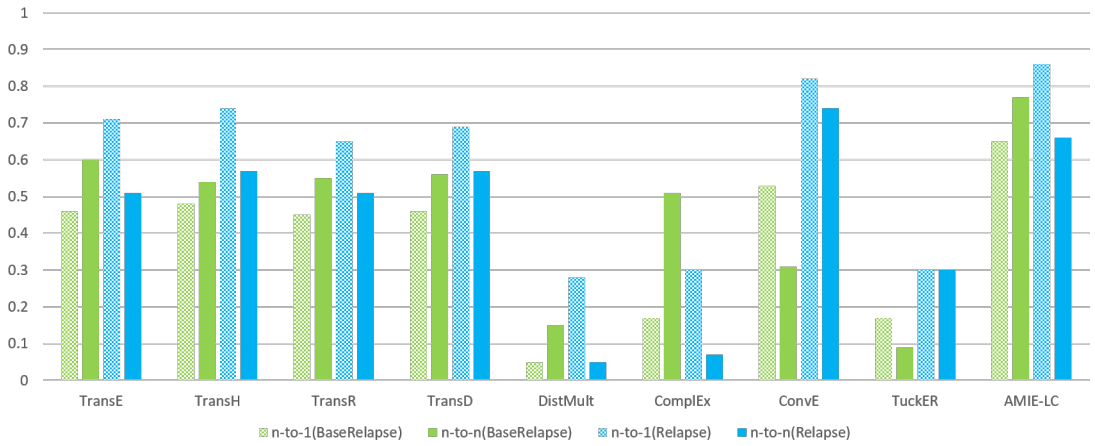


Figure 5.16: Evaluation MRR(↑) over Relapse with different relation types. Better performance is indicated by a higher MRR value.

Relation Types	n-to-1	n-to-n	1-to-1	n-to-1	n-to-n	1-to-1	n-to-1	n-to-n	1-to-1	n-to-1	n-to-n	1-to-1	n-to-1	n-to-n	1-to-1
Model	MR(↓)			MRR(↑)			Hits@1(↑)			Hits@3(↑)			Hits@10(↑)		
TransE	384	14	**8,213**	0.44	0.61	0	0.3	0.4	0	0.51	0.77	0	0.67	**0.96**	0
TransH	850	16	15,579	0.46	0.65	0	0.3	0.47	0	0.59	**0.78**	0	0.73	**0.96**	0
TransR	1,761	13	25,676	0.46	0.61	0	0.3	0.43	0	0.58	0.72	0	0.71	**0.96**	0
TransD	712	**11**	8,843	0.43	0.55	0	0.28	0.36	0	0.57	0.67	0	0.71	**0.96**	0
RotatE	5,876	2067	49,746	0.01	0.01	0	0	0	0	0	0	0	0.01	0.03	0
DistMult	986	113	4,0568	0.09	0.23	0	0	0.03	0	0.1	0.26	0	0.24	0.84	0
ComplEx	1,108	294	57,911	0.19	0.37	0	0.08	0.17	0	0.22	0.49	0	0.42	0.81	0
ConvE	1,139	202	41,165	0.67	0.31	0	0.43	0.08	0	1.17	0.22	0	0.74	0.93	0
TuckER	**330**	68	67,519	**0.68**	0.17	0	**0.69**	0	0	**0.86**	0	0	0.52	0.7	0
AMIE-LC	507	14	14,618	0.67	**0.71**	0	0.61	**0.62**	0	0.72	0.75	0	**0.75**	0.93	0

Table 5.9: Right_Link prediction result on BaseBiomarker over different relations types. Better performance is indicated by a lower MR value, along with higher values for MRR and Hits@k.

Tables 5.9 and 5.11 show that AMIE-LC and TuckER outperform the other models in the `n-to-n` and `n-to-1` relation in the BaseBiomarker and BaseRelapse benchmarks. Tables 5.9 and 5.11 show that we cannot predict objects in `1-to-1` and `1-to-n` relations, which is due to CVT nodes. In lung cancer KG, the object of the triples with `1-to-1` and `1-to-n` relations are all CVT nodes. As we remove CVT nodes from the biomarker and the relapse benchmark, performance improves. Tables 5.10 and 5.12 show that AMIE-LC and ConvE outperform the other models in the `n-to-n` and `n-to-1` relation in the Biomarker benchmark.

Relation Types	n-to-1	n-to-n	n-to-1	n-to-n	n-to-1	n-to-n	n-to-1	n-to-n	n-to-1	n-to-n
Model	MR(↓)		MRR(↑)		Hits@1(↑)		Hits@3(↑)		Hits@10(↑)	
TransE	6	**13**	0.65	0.37	0.46	0.16	0.83	0.5	0.96	0.79
TransH	4	20	0.72	0.47	0.55	0.27	0.86	0.63	**0.97**	0.78
TransR	8	22	0.63	0.37	0.43	0.17	0.81	0.49	0.96	0.77
TransD	8	15	0.67	0.4	0.49	0.19	0.85	0.55	0.96	0.77
RotatE	1,330	18	0.01	0	0	0	0	0	0.02	0.01
DistMult	19	52	0.29	0.03	0.13	0	0.33	0	0.66	0
ComplEx	12	37	0.31	0.01	0.13	0	0.36	0	0.67	0.03
ConvE	31	104	0.65	0.27	0.48	0.09	0.8	0.36	0.92	0.64
TuckER	45	42	0.34	0.34	0.28	0.28	0.31	0.32	0.37	0.41
AMIE-LC	**2**	55	**0.85**	**0.55**	**0.77**	**0.39**	**0.93**	**0.73**	0.96	**0.81**

Table 5.10: Right_Link prediction result on Biomarker over different relations types. Better performance is indicated by a lower MR value, along with higher values for MRR and Hits@k.

Relation Types	n-to-1	n-to-n	1-to-1	1-to-n	n-to-1	n-to-n	1-to-1	1-to-n	n-to-1	n-to-n	1-to-1	1-to-n	n-to-1	n-to-n	1-to-1	n-to-1	1-to-n	n-to-n	1-to-1	1-to-n
Model	MR(↓)				MRR(↑)				Hits@1(↑)				Hits@3(↑)				Hits@10(↑)			
TransE	713	20	8,422	11,391	0.46	0.6	0	0	0.33	0.41	0	0	0.54	0.75	0	0	0.67	0.96	0	0
TransH	1,554	25	16,076	10,165	0.48	0.54	0	0	0.34	0.34	0	0	0.58	0.7	0	0	0.7	0.96	0	0
TransR	3,982	17	25,965	24,378	0.45	0.55	0	0	0.3	0.34	0	0	0.56	0.74	0	0	**0.71**	**0.97**	0	0
TransD	1,032	**6**	8,246	**9,227**	0.46	0.56	0	0	0.31	0.33	0	0	0.57	0.75	0	0	0.7	0.96	0	0
RotatE	9,028	2,932	66,554	46,302	0	0	0	0	0	0	0	0	0	0	0	0	0	0.02	0	0
DistMult	1,759	429	76,858	38,982	0.05	0.15	0	0	0	0.02	0	0	0	0.1	0	0	0.18	0.59	0	0
ComplEx	2,789	50	62,035	57,967	0.17	0.51	0	0	0.07	0.4	0	0	0.16	0.58	0	0	0.38	0.69	0	0
ConvE	863	954	61,050	15,342	0.53	0.31	0	0	0.27	0.28	0	0	0.57	0.28	0	0	0.66	0.91	0	0
TuckER	2,881	7,838	**7,675**	67,695	0.17	0.09	0	0	0.08	0	0	0	0.21	0	0	0	0.34	0.59	0	0
AMIE-LC	**675**	14	14,544	18,639	**0.65**	**0.77**	0	0	**0.59**	**0.69**	0	0	**0.7**	**0.83**	0	0	**0.71**	0.92	0	0

Table 5.11: Right_Link prediction result on BaseRelapse over different relations types. Better performance is indicated by a lower MR value, along with higher values for MRR and Hits@k.

Relation Types	n-to-1	n-to-n	n-to-1	n-to-n	n-to-1	n-to-n	n-to-1	n-to-n	n-to-1	n-to-n
Model	MR(↓)		MRR(↑)		Hits@1(↑)		Hits@3(↑)		Hits@10(↑)	
TransE	4	**9**	0.71	0.51	0.55	0.29	0.84	0.66	0.97	0.92
TransH	8	13	0.74	0.57	0.59	0.36	0.86	0.77	0.98	0.92
TransR	9	12	0.65	0.51	0.44	0.3	0.83	0.65	0.95	0.92
TransD	3	13	0.69	0.57	0.52	0.35	0.85	0.76	0.97	0.92
RotatE	1045	1321	0.01	0.01	0	0	0	0	0.02	0.02
DistMult	19	37	0.28	0.05	0.14	0	0.27	0	0.65	0.27
ComplEx	13	27	0.3	0.07	0.14	0	0.32	0	0.66	0.28
ConvE	17	38	0.82	**0.74**	0.71	**0.62**	**0.94**	0.83	**0.97**	**0.94**
TuckER	19	33	0.3	0.3	0.16	0.22	0.32	0.29	0.65	0.44
AMIE-LC	**2**	20	**0.86**	0.66	**0.79**	0.48	0.93	**0.84**	0.96	0.92

Table 5.12: Right_Link prediction result on Relapse over different relations types. Better performance is indicated by a lower MR value, along with higher values for MRR and Hits@k.

5.6 Summary of the Chapter

In this section, we evaluated different numerical and symbolic models and our composed approach for link prediction task on AMIE over lung cancer KG. We go into detail about experimental setup and preprocessig steps. We calculated MR, MRR, Hits@1, Hits@3, and Hits@10 to evaluate the models. To conclude, this section proves the impact of the data scheme on our results and shows how our preprocessing step improves the outcome in all numerical models and AMIE-LC. It shows that AMIE-LC outperforms the other models in all benchmarks.

Chapter 6

Conclusions and Future Work

In this chapter, we summarize the most important results of our work and the methods used to solve the problem and achieve our goal. We also highlight the main limitations of our work and explore the perspectives for future research within the previously identified constraints.

6.1 Conclusions

This master thesis focuses on the evaluation of hybrid AI for the prediction over lung cancer KG to address the following research questions:

Q1) How efficient is our proposed AMIE-LC approach compared to numerical models in the KG?
Our proposed AMIE-LC approach outperform all numerical models.

Q2) What is the impact of the extension of the KG on the task of link prediction, in particular in our scenario of the integration of Biomarker KG with Relapse KG?
Most approaches perform less well with the integration of relapse data without any preprocessing. However, by using preprocessed data and new data schema, the integration of relapse data improves performance.

Q3) How can the performance of the link prediction task be enhanced by changing the data schema within KG?
The removal of CVT nodes, elimination of data redundancy, and properly categorizing data through preprocessing methods lead to remarkable improvements in

prediction.

Q4) Do models developed for the link prediction task demonstrate varied evaluations based on different types of relations within a KG?
The removal of the relation types `1-to-1` and `1-to-n` due to the removal of CVT nodes in the new data schema contributes to the overall performance improvement. Our evaluation also shows that the relation type `n-to-1` performs better compared to the other relation types.

In conclusion, this thesis presents an investigation of data schema normalization in lung cancer KG and its impact on the performance of link prediction models. The experiments show that the AMIE-LC approach outperforms all numerical models, and the `n-to-1` relation type outperforms the other relation types. Furthermore, the study highlights the importance of data schema for the performance of link prediction tasks in knowledge graphs. The results show that data schema normalization improves the link prediction performance of all numerical models and AMIE-LC approach. The data schema normalization also improved the performance by integrating KGs.

6.2 Limitations

Like any other framework, AMIE-LC and numerical models have some limitations as well. In the following, we will talk about the limitations of our work:

- **Complexity of Data Modeling**: Integration of multiple KGs make data modeling more complex. This involves creating efficient data schema through find similar instances, data redundancy, removing mediator nodes, and categorizing values within triples.

- **Unnecessary Mining of Rules**: The processing of symbolic systems may result in the mining of rules and inference of triples that link prediction may not use, introducing potential inefficiencies.

- **Optimization Challenges**: All used approaches for link prediction, by their nature, require the optimization of hyperparameters related to different methods. Adjusting these hyperparameters for both symbolic and numerical methods is crucial for improving performance.

6.3 Future Work

We suggest researching the following topics for future work:

- **Advanced Preprocessing Techniques**: Exploring advanced preprocessing techniques, such as feature engineering or more sophisticated data cleansing methods, to normalize the data schema to benefit the predictive models, and a schema transformation algorithm.
- **Automation Methods for Hyperparameter Tuning**: Researching and implementing automation methods for tuning hyperparameters of approaches. This can contribute to optimize process and improve the adaptability of the models.
- **Develop a New Hybrid Model**: Development of an efficient hybrid model in which a balance between symbolic and numerical methods must be found in order to increase the effectiveness of the integrated model.

6.4 Summary of the Chapter

At the beginning of this chapter, we explain what we have learned in this Master's thesis. Then we talk about the limitations of our work and finally about future work.

Bibliography

[1] Farahnaz Akrami, Mohammed Samiul Saeef, Qingheng Zhang, Wei Hu, and Chengkai Li. "Realistic re-evaluation of knowledge graph completion methods: An experimental study". In: *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2020, pp. 1995–2010.

[2] Ivana Balažević, Carl Allen, and Timothy M Hospedales. "Hypernetwork knowledge graph embeddings". In: *Artificial Neural Networks and Machine Learning–ICANN 2019: Workshop and Special Sessions: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings 28*. Springer. 2019, pp. 553–565.

[3] Ivana Balažević, Carl Allen, and Timothy M Hospedales. "Tucker: Tensor factorization for knowledge graph completion". In: *arXiv preprint arXiv:1901.09590* (2019).

[4] M Bergman. "Sources and classification of semantic heterogeneities". In: *Web Blog: AI3-Adaptive Information, Adaptive Innovation, Adaptive Infrastructure* (2006).

[5] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. "Freebase: a collaboratively created graph database for structuring human knowledge". In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 2008, pp. 1247–1250.

[6] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. "Translating embeddings for modeling multi-relational data". In: *Advances in neural information processing systems* 26 (2013).

[7] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. "A comprehensive survey of graph embedding: Problems, techniques, and applications". In: *IEEE transactions on knowledge and data engineering* 30.9 (2018), pp. 1616–1637.

[8] World Wide Web Consortium et al. "RDF 1.1 concepts and abstract syntax". In: (2014).

[9] Christophe Debruyne and Declan O'Sullivan. "R2RML-F: Towards Sharing and Executing Domain Logic in R2RML Mappings." In: *LDOW@ WWW* 1593 (2016).

[10] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. "Convolutional 2d knowledge graph embeddings". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[11] Henrik Dibowski, Stefan Schmid, Yulia Svetashova, Cory Henson, and Tuan Tran. "Using Semantic Technologies to Manage a Data Lake: Data Catalog, Provenance and Access Control." In: *SSWS@ ISWC*. Athen. 2020, pp. 65–80.

[12] Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. "RML: A generic language for integrated RDF mappings of heterogeneous data." In: *Ldow* 1184 (2014).

[13] Xin Luna Dong. "Generations of Knowledge Graphs: The Crazy Ideas and the Business Impact". In: *arXiv preprint arXiv:2308.14217* (2023).

[14] Dieter Fensel, Umutcan Şimşek, Kevin Angele, Elwin Huaman, Elias Kärle, Oleksandra Panasiuk, Ioan Toma, Jürgen Umbrich, Alexander Wahler, Dieter Fensel, et al. "Introduction: what is a knowledge graph?" In: *Knowledge graphs: Methodology, tools and selected use cases* (2020), pp. 1–10.

[15] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. "AMIE: association rule mining under incomplete evidence in ontological knowledge bases". In: *Proceedings of the 22nd international conference on World Wide Web*. 2013, pp. 413–422.

[16] Frank L Hitchcock. "The expression of a tensor or a polyadic as a sum of products". In: *Journal of Mathematics and Physics* 6.1-4 (1927), pp. 164–189.

[17] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. "Knowledge graph embedding via dynamic mapping matrix". In: *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*. 2015, pp. 687–696.

[18] Ni Lao and William W Cohen. "Relational retrieval using a combination of path-constrained random walks". In: *Machine learning* 81 (2010), pp. 53–67.

[19] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. "Learning entity and relation embeddings for knowledge graph completion". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 29. 1. 2015.

[20] Thierry Mamer, Christopher H Bryant, and John McCall. "L-modified ilp evaluation functions for positive-only biological grammar learning". In: *Inductive Logic Programming: 18th International Conference, ILP 2008 Prague, Czech Republic, September 10-12, 2008 Proceedings 18*. Springer. 2008, pp. 176–191.

[21] Stephen Muggleton. "Learning from positive data". In: *International conference on inductive logic programming*. Springer. 1996, pp. 358–376.

[22] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. "A review of relational machine learning for knowledge graphs". In: *Proceedings of the IEEE* 104.1 (2015), pp. 11–33.

[23] Maximilian Nickel and Volker Tresp. "Tensor factorization for multi-relational learning". In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13*. Springer. 2013, pp. 617–621.

[24] Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. "Industry-scale Knowledge Graphs: Lessons and Challenges: Five diverse technology companies show how it's done". In: *Queue* 17.2 (2019), pp. 48–75.

[25] *OWL*. URL: `https://www.w3.org/OWL/`.

[26] Stephan Rabanser, Oleksandr Shchur, and Stephan Günnemann. "Introduction to tensor decompositions and their applications in machine learning". In: *arXiv preprint arXiv:1711.10781* (2017).

[27] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. "Yago: A large ontology from wikipedia and wordnet". In: *Journal of Web Semantics* 6.3 (2008), pp. 203–217.

[28] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. "Rotate: Knowledge graph embedding by relational rotation in complex space". In: *arXiv preprint arXiv:1902.10197* (2019).

[29] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. "Complex embeddings for simple link prediction". In: *International conference on machine learning*. PMLR. 2016, pp. 2071–2080.

[30] Maria-Esther Vidal, Emetis Niazmand, Philipp D Rohde, Enrique Iglesias, and Ahmad Sakor. "Challenges for Healthcare Data Analytics Over Knowledge Graphs". In: *Transactions on Large-Scale Data-and Knowledge-Centered Systems LIV: Special Issue on Data Management-Principles, Technologies, and Applications*. Springer, 2023, pp. 89–118.

[31] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. "Knowledge graph embedding: A survey of approaches and applications". In: *IEEE Transactions on Knowledge and Data Engineering* 29.12 (2017), pp. 2724–2743.

[32] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. "Knowledge graph embedding by translating on hyperplanes". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 28. 1. 2014.

[33] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. "Embedding entities and relations for learning and inference in knowledge bases". In: *arXiv preprint arXiv:1412.6575* (2014).