LEVERAGING LITERALS FOR KNOWLEDGE GRAPH EMBEDDINGS

Zur Erlangung des akademischen Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN

(Dr.-Ing.)

von der KIT-Fakultät für Wirtschaftswissenschaften

des Karlsruher Instituts für Technologie (KIT)

genehmigte

DISSERTATION

von

M.Sc. GENET ASEFA GESESE

Tag der mündlichen Prüfung: 12.07.2023

Referent: Prof. Dr. Harald Sack

Korreferentin: Assoc. Prof. Dr. Mehwish Alam

Karlsruhe (2023)

This thesis is dedicated to:

my grandmother, *Abiye*. Your love and sacrifices will never be forgotten.

and

to the loving memory of my father *Asefa Gesese*, who passed away during the final year of my Ph.D. I know you are still rooting for me from above, Thank you!!

ABSTRACT

Knowledge Graphs (KGs) are a structured representation of facts pertaining to a specific domain or multiple domains, composed of entities and their relationships. KGs have been used in various applications such as relation extraction, question answering, and recommender systems. However, to maximize efficiency, it is beneficial to transform KGs into a low-dimensional vector space. Moreover, the incompleteness of KGs caused by the open-world assumption hinders their applicability to real-world use cases. Hence, there is a need for embedding-based link prediction (LP) approaches to complete KGs. LP can be performed in two settings, transductive and inductive, where the former requires all test set entities to be present in the training set, while the latter allows for the possibility of test set entities that were not seen during training. This thesis investigates the use of literals in both transductive and inductive LP, as large KGs contain numerous numerical and textual literals that hold essential semantics. Additionally, high-quality benchmark datasets for the evaluation of LP methods are proposed.

Specifically, a novel KG embedding (KGE) method RAILD is proposed which leverages textual literals along with graph contextual information to learn embeddings of KGs with a LP objective. RAILD aims to address the gap with the state-of-the-art embedding models in learning embeddings for relations that are not observed during training. In order to do so, it proposes an architecture that combines language models (LMs) with network embeddings. It fine-tunes powerful pre-trained LMs such as BERT with a LP objective by utilizing the textual descriptions of entities and relations. It also introduces a new algorithm WeiDNeR to generate a network of relations which is then used to learn graph-based embeddings of relations using a network embedding model. The representations of relations obtained using the given LM as well as the network embedding model are combined when performing the LP task. Moreover, another novel embedding model LitKGE is proposed which utilizes numeric literals for transductive LP. It aims to generate numerical features for entities through graph traversal. To achieve this, it introduces a new algorithm WeiDNeR_Extended to generate a network of object properties and datatype properties. The property paths extracted from this network are used to obtain the numerical features of entities.

Furthermore, in this thesis, the benefits of utilizing a multilingual LM for encoding entity descriptions in various natural languages for the LP task are studied. For the evaluation of KGE models, the benchmark datasets LiterallyWikidata and Wikidata68K are created. The promising results obtained with the models proposed in this thesis open up interesting directions for future research in the area of KGEs and their downstream tasks.

ZUSAMMENFASSUNG

Wissensgraphen (Knowledge Graphs, KGs) repräsentieren strukturierte Fakten, die sich aus Entitäten und den zwischen diesen bestehenden Relationen zusammensetzen. Um die Effizienz von KG-Anwendungen zu maximieren, ist es von Vorteil, KGs in einen niedrigdimensionalen Vektorraum zu transformieren. KGs folgen dem Paradigma einer offenen Welt (Open World Assumption, OWA), d. h. fehlende Information wird als potenziell möglich angesehen, wodurch ihre Verwendung in realen Anwendungsszenarien oft eingeschränkt wird. Link-Vorhersage (Link Prediction, LP) zur Vervollständigung von KGs kommt daher eine hohe Bedeutung zu. LP kann in zwei unterschiedlichen Modi durchgeführt werden, transduktiv und induktiv, wobei die erste Möglichkeit voraussetzt, dass alle Entitäten der Testdaten in den Trainingsdaten vorhanden sind, während die zweite Möglichkeit auch zuvor nicht bekannte Entitäten in den Testdaten zulässt. Die vorliegende Arbeit untersucht die Verwendung von Literalen in der transduktiven und induktiven LP, da KGs zahlreiche numerische und textuelle Literale enthalten, die eine wesentliche Semantik aufweisen. Zur Evaluierung dieser LP Methoden werden spezielle Benchmark-Datensätze eingeführt.

Insbesondere wird eine neuartige KG Embedding (KGE) Methode, RAILD, vorgeschlagen, die Textliterale zusammen mit kontextuellen Graphinformationen für die LP nutzt. Das Ziel von RAILD ist es, die bestehende Forschungslücke beim Lernen von Embeddings für beim Training ungesehene Relationen zu schließen. Dafür wird eine Architektur vorgeschlagen, die Sprachmodelle (Language Models, LMs) mit Netzwerkembeddings kombiniert. Hierzu erfolgt ein Feintuning von leistungsstarken vortrainierten LMs wie BERT zum Zweck der LP, wobei textuelle Beschreibungen von Entitäten und Relationen genutzt werden. Darüber hinaus wird ein neuer Algorithmus, WeiDNeR, eingeführt, um ein Relationsnetzwerk zu generieren, das zum Erlernen graphbasierter Embeddings von Relationen unter Verwendung eines Netzwerkembeddingsmodells dient. Die Vektorrepräsentationen dieser Relationen werden für die LP kombiniert. Zudem wird ein weiteres neuartiges Embeddingmodell, LitKGE, vorgestellt, das numerische Literale für die transduktive LP verwendet. Es zielt darauf ab, numerische Merkmale für Entitäten durch Graphtraversierung zu erzeugen. Hierfür wird ein weiterer Algorithmus, WeiDNeR_Extended, eingeführt, der ein Netzwerk aus Objekt- und Datentypproperties erzeugt. Aus den aus diesem Netzwerk extrahierten Propertypfaden werden dann numerische Merkmale von Entitäten generiert.

Des Weiteren wird der Einsatz eines mehrsprachigen LM zur Kodierung von Entitätenbeschreibungen in verschiedenen natürlichen Sprachen zum Zweck der LP untersucht. Für die Evaluierung der KGE-Modelle wurden die Benchmark-Datensätze LiterallyWikidata und Wikidata68K erstellt. Die vielversprechenden Ergebnisse, die mit den vorgestellten Modellen erzielt wurden, eröffnen interessante Fragestellungen für die zukünftige Forschung auf dem Gebiet der KGEs und ihrer Folgeanwendungen.

LIST OF PUBLICATIONS

This thesis is based on the following publications:

[1] Genet Asefa Gesese, Russa Biswas, Mehwish Alam, and Harald Sack. "A Survey on Knowledge Graph Embeddings with Literals: Which Model Links Better Literal-Ly?" In: *Semantic Web* 12.4 (2021), 617–647. ISSN: 1570-0844. DOI: 10.3233/SW-200404. URL: https://doi.org/10.3233/SW-200404.

[2] Genet Asefa Gesese, Mehwish Alam, and Harald Sack. "LiterallyWikidata - A Benchmark for Knowledge Graph Completion Using Literals." In: *The Semantic Web – ISWC 2021*. Cham: Springer International Publishing, 2021, pp. 511–527. ISBN: 978-3-030-88361-4.

[3] Genet Asefa Gesese, Mehwish Alam, and Harald Sack. "RAILD: Towards Leveraging Relation Features for Inductive Link Prediction In Knowledge Graphs." In: *IJCKG*. 2022.

[4] Genet Asefa Gesese, Russa Biswas, and Harald Sack. "A Comprehensive Survey of Knowledge Graph Embeddings with Literals: Techniques and Applications." In: *Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG2019) Co-located with the 16th Extended Semantic Web Conference 2019 (ESWC 2019), Portoroz, Slovenia, June 2, 2019*. 2019, pp. 31–40. URL: http://ceur-ws.org/Vol-2377/paper_4.pdf.

[5] Genet Asefa Gesese, Mehwish Alam, and Harald Sack. "Semantic Entity Enrichment by Leveraging Multilingual Descriptions for Link Prediction." In: *DL4KG workshop co-located with ESWC*. 2020.

[6] Genet Asefa Gesese, Mehwish Alam, Fabian Hoppe, and Harald Sack. "Leveraging Multilingual Descriptions for Link Prediction: Initial Experiments." In: *ISWC 2020 Posters and Demos Track, co-located with ISWC*. 2020.

[7] Genet Asefa Gesese. "Leveraging Literals for Knowledge Graph Embeddings." In: *International semantic web conference, Doctoral Consortium*. 2021.

[8] Genet Asefa Gesese, Harald Sack, and Mehwish Alam. "LitKGE: Improving numeric LITerals based Knowledge Graph Embedding models." In: *[Under Review]*. 2023.

The author has further contributed to the following publications:

[1] Cristian Santini, Genet Asefa Gesese, Silvio Peroni, Aldo Gangemi, Harald Sack, and Mehwish Alam. “A Knowledge Graph Embeddings Based Approach for Author Name Disambiguation Using Literals.” In: *Scientometrics* 127.8 (2022), 4887–4912. ISSN: 0138-9130. DOI: 10.1007/s11192-022-04426-2. URL: https://doi.org/10.1007/s11192-022-04426-2.

[2] Mehwish Alam, Russa Biswas, Yiyi Chen, Danilo Dessì, Genet Asefa Gesese, Fabian Hoppe, and Harald Sack. “HierClasSArt: Knowledge-Aware Hierarchical Classification of Scholarly Articles.” In: *Companion Proceedings of the Web Conference 2021*. WWW ’21. Ljubljana, Slovenia: Association for Computing Machinery, 2021, 436–440. ISBN: 9781450383134. DOI: 10.1145/3442442.3451365. URL: https://doi.org/10.1145/3442442.3451365.

[3] Rick Petzold, Genet Asefa Gesese, Viktoria Bogdanova, Thorsten Zylowski, Harald Sack, and Mehwish Alam. “Challenges of Applying Knowledge Graph and their Embeddings to a Real-world Use-case.” In: *Workshop on Deep Learning for Knowledge Graphs (DL4KG 2021), co-located with the 20th International Semantic Web Conference (ISWC 2021), Virtual Conference, online, October 25, 2021*. Vol. 3034. 2021, p. 4.

[4] Mehwish Alam, Genet Asefa Gesese, Zahra Rezaie, and Harald Sack. “MigrAnalytics: Entity-based Analytics of Migration Tweets.” In: *ISWC 2020 Posters and Demos Track, co-located with ISWC*. 2020.

[5] Yiyi Chen, Genet Asefa Gesese, Harald Sack, and Mehwish Alam. “Temporal Evolution of the Migration-related Topics on Social Media.” In: *ISWC 2021 Posters and Demos Track, co-located with ISWC*. 2021.

ACKNOWLEDGMENTS

I would like to start by expressing my deepest gratitude to Prof. Dr. Harald Sack for giving me the opportunity to pursue my Ph.D. under his supervision. Throughout the years, he has provided me with unwavering support, encouragement, and thought-provoking discussions that have helped me to navigate the research challenges with precision and confidence. My gratitude also extends to Asoc. Prof. Dr. Mehwish Alam, who has been instrumental in my success. Her mentorship has been invaluable, as she recognized my research efforts and gave me the motivation and confidence I needed to succeed.

I also want to thank my team for all the unforgettable experiences that I have shared with you. I have learned a lot from working with you! A special mention goes to Russa with whom I have had successful collaborations and interesting professional and personal discussions, and to Vivien for all the support that you have given me. To Yiyi, heartfelt thanks for your unwavering kindness. I would like also to extend my thanks to Alex, Rima, and Angela, for their support and assistance in various personal matters over the past year. Their help was greatly appreciated.

I am eternally grateful to God for providing me with my chosen sister, Shibire Bekele, who has been a constant source of support and encouragement since the day we met - she made Germany home for me. Though not bound by blood, we share a heavenly mother and our bond is as strong as any sisterhood, and I am blessed to have her in my life. I would also like to express my heartfelt gratitude to Netsanet, who has been a supportive friend and made Karlsruhe a home away from home. Moreover, I am beyond grateful to the Ethiopian community and friends in Karlsruhe and in Germany in general for creating an environment that is full of support and cooperation.

Last but not least, my grandmother - Abiye, who has been my backbone since childhood, my father Asefa, and my family in general, thank you for all that you have done for me!!

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

KG	Knowledge Graph
KGC	Knowledge Graph Completion
KGE	Knowledge Graph Embeddings
LP	Link Prediction
NLP	Natural Language Processing
SoTA	State-of-the-art
ML	Machine Learning
RELU	Rectified Linear Unit
FNN	Feed-forward Neural Networks
MLP	Multi-Layer Perceptron
FCNN	Fully Connected Neural Network
CNN	Convolutional Neural Networks
GRU	Gated Recurrent Unit
LM	Language Model
NLM	Neural Language Model
SLM	Statistical Language Model
OOV	Out-of-Vocabulary
BERT	Bidirectional Encoder Representations from Transformers
MRR	Mean Reciprocal Rank

Part I

MOTIVATION AND FOUNDATIONS

1 INTRODUCTION

Knowledge Graphs (KGs) have been among the driving forces in the advancement of Artificial Intelligence (AI), mainly in the field of semantic understanding and Natural Language Processing (NLP), especially since the launch of Google's KG in 2012. They have become quite crucial to storing structured information and can be used to improve various machine learning applications. Popular general-purpose KGs such as DBpedia [1], Wikidata [2], Freebase [3], and YAGO [4], which consist of vast quantities of facts represented using millions of entities as nodes and relations as edges, are publicly available and facilitate a range of machine learning applications such as question answering [5], recommender systems [6], and relation extraction [7]. Even though knowledge graphs are useful for representing structured data, their symbolic nature often makes them challenging to handle. To address this problem, an area of research called Knowledge Graph Embeddings (KGE) has emerged and gained significant interest [8, 9]. Despite the fact that KGs contain information represented as literals (i.e., text, numerical values, and so on), in addition to entities and relations, most of the existing KGE approaches do not utilize the semantics present in those literals. Therefore, in this thesis, the use of literals for the representation learning of KGs is studied.

The rest of this introductory chapter is organized as follows. To begin with, Section 1.1 elaborates on the motivation behind the research works presented in this thesis. Subsequently, the hypotheses and the formulated research questions are given in Section 1.2. Finally, the contributions made and the thesis outline are provided in Section 1.3.

1.1 MOTIVATION

KG is a large network of facts that are organized in the form of triples [10]. A triple can be represented as a connection between either two entity nodes (`<head` h`, relation` r`, tail` t`>`) or an entity node and a literal node (`<entity` e`, attribute` a`, literal` l`>`), where the relation r and the attribute a are directed and labeled edges. Figure 1.1.1 illustrates a KG that contains information about the real-world entity OpenAI. The entities *dbr:OpenAI*, *dbr:GPT-2*, etc. are represented as nodes and the relations/attributes such as *dbo:product* and *dbo:releaseDate* are represented as directed edges. The KG encompasses multiple triples such as <*dbr:OpenAI, dbo:product, dbr:GPT-2*> which represents a relationship between the entities *dbr:OpenAI* and *dbr:GPT-2*. These connections between entities are commonly known as relational triples. In addition to such triples, the KG also contains auxiliary information about the entities in the form of triples which are often referred to as attributive triples. For example, <*dbr:GPT-2, dbo:releaseDate, "2019-02-14"*> represents a connection between

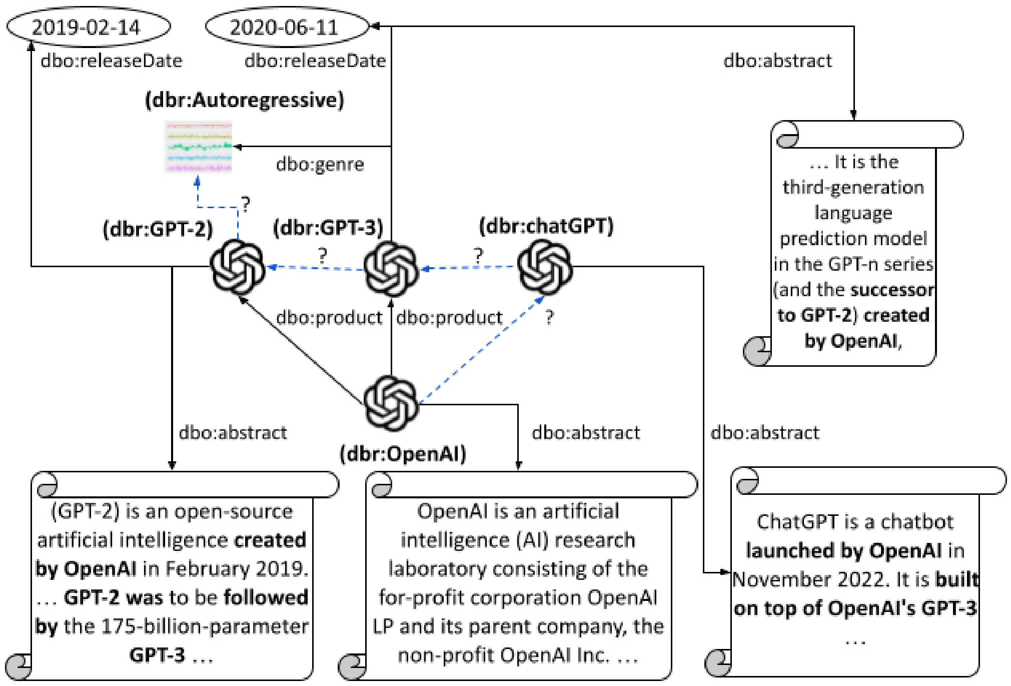

2019-02-14
2020-06-11
dbo:releaseDate
dbo:releaseDate
dbo:abstract
(dbr:Autoregressive)
dbo:genre
?
(dbr:GPT-2)
(dbr:GPT-3)
(dbr:chatGPT)
?
?
dbo:product
dbo:product
?
dbo:abstract
(dbr:OpenAI)
dbo:abstract
dbo:abstract
… It is the third-generation language prediction model in the GPT-n series (and the successor to GPT-2) created by OpenAI,
(GPT-2) is an open-source artificial intelligence created by OpenAI in February 2019. … GPT-2 was to be followed by the 175-billion-parameter GPT-3 …
OpenAI is an artificial intelligence (AI) research laboratory consisting of the for-profit corporation OpenAI LP and its parent company, the non-profit OpenAI Inc. …
ChatGPT is a chatbot launched by OpenAI in November 2022. It is built on top of OpenAI's GPT-3 …

Figure 1.1.1: An example KG representing real-world entities

the entity *dbr:GPT-2* and the literal node representing the date "2019-02-14", which encodes the date of release of GPT-2. Moreover, in this KG, textual descriptions about the entities are also provided in the form of attributive triples, e.g., *<dbr:OpenAI, dbo:abstract, "OpenAI is an artificial intelligence (AI) research laboratory consisting of the for-profit corporation OpenAI LP and its parent company, the non-profit OpenAI Inc. . . . ">.*

KGs have been demonstrated to be an effective means of representing structured data. However, there are several limitations that hinder their efficient manipulation. These limitations include: i) KGs are usually based on different rigorous symbolic frameworks, which makes it challenging to utilize their data in other applications [11] and ii) the fact that a significant number of important graph algorithms required for efficient manipulation and analysis of KGs have been proven to be NP-complete [12]. To address these limitations and increase the efficiency of working with KGs, several studies have been conducted which learn embeddings of KGs by mapping them to low dimensional vector spaces [8, 13, 14]. These studies aim to preserve the underlying semantics of the KG while reducing its dimensionality.

Another challenge in the effective utilization of KGs is the incompleteness of the information stored within them. Open KGs, such as Wikidata, DBpedia, and Freebase which span multiple domains are either curated by human editors, extracted through automated or semi-automated methods, or generated using heuristics. These KGs operate under open-world assumptions, which means that they are not a complete representation of all possible knowledge in a given domain. As a result, there are always missing facts or incomplete information within these KGs. The incompleteness of KGs has a significant impact on their usage for different real-world applications. For example, the KG depicted in Figure 1.1.1 is characterized by incompleteness, as shown by the links labeled with "?" representing missing triples. Consequently, it is plausible that a Question Answering system that employs this KG would be unable to provide either an accurate response or any response to the query "What is the genre of GPT-2?" despite the fact that both the entity "GPT-2" and the relation "genre" exist in the KG. This implies the necessity of predicting links between entities while leveraging the semantics which is present in the KG.

Literals, being one of the integral components of KGs, provide crucial semantics about entities and relations. For instance, if a KG contains entities that represent individuals and literal values indicating their ages, incorporating these literal values when learning the representation of the KG can result in entities with similar ages being placed closer together in the vector space, while entities with different ages are separated. Another example, in reference to the KG shown in Figure 1.1.1, the textual descriptions of the entities <*dbr:GPT-2*> and <*dbr:GPT-3*> provide details about the companies responsible for creating or launching them and also reveal the sequence in which these two products were introduced. Hence, incorporating literals can significantly enhance the performance of KG completion tasks, as well as the transformation of KGs into vector spaces for optimized manipulation. A majority of large-scale KGs, such as Wikidata, contain a substantial number of numerical attributive triples which connect entities to numerical literal nodes. In addition to numerical literals, KGs also contain a significant amount of both short and long textual literals, such as names, labels, and descriptions of entities and relations. These literals play a crucial role in providing vital semantics about the entities and should be leveraged to help generate new facts that are not included in the KG.

Link Prediction (LP) is a widely known KG Completion (KGC) task that could be used to address the above-mentioned problems, i.e., to transform KG elements into a low-dimensional vector space while generating missing facts in the KG. Specifically, LP is the task of estimating the likelihood of the existence of links between entities based on the current observed information in the KG. It can be divided into two major categories i) transductive LP and ii) inductive LP based on the nature of the prediction, i.e., whether it involves entities that are not observed during training. These categories are discussed in detail as follows:

- **Transductive LP:** All entities in the test and validation sets are required to be part of the training set. In other words, in this LP setting, it is possible to complement the KG only with those entities that are already observed during training. For example in Figure 1.1.1, despite the fact that both the entities *dbr:GPT-2* and *dbr:Autoregressive* and the relation *genre* exist in the KG, the fact <*dbr:GPT-2*, *genre*, *dbr:Autoregressive*> is missing. This fact could be predicted by leveraging the textual descriptions of the entities. Different LP models which leverage textual literals, numerical literals or both are introduced so far. However, these models do not leverage indirect associations between entities and literals, i.e., literals are considered only as features (sources of information) for the entities to which they are directly linked. For evaluation of those LP methods which utilize literals in transductive setting, the existing KGC benchmark datasets such as FB15K-237 [15], WN18RR [16], and YAGO3-10 [16] are used. As discussed in [17], these benchmarks do not give proper emphasis to attributive triples, i.e., attributes are treated as auxiliary information. Consequently, the attributive triples are either way unbalanced, less in number, or have few unique attributes.

- **Inductive LP:** A LP task performed in a setting where the validation and test sets may contain entities that are not seen during training is referred to as inductive LP. For example in Figure 1.1.1, it can be observed that the ownership of the product ChatGPT by OpenAI and the entity ChatGPT itself are not included in the KG. In order to compliment the KG with this information, the link <*dbr:OpenAI*, *dbo:product*, *dbr:chatGPT*> should be generated. This could be achieved by utilizing the information present in the textual descriptions of the entities *dbr:OpenAI* and *dbr:chatGPT* which indicate that 'ChatGPT is a chatbot launched by OpenAI'.

  The existing representation learning-based inductive LP approaches such as BLP [18] and KEPLER [19], do not pay attention to relations. Unlike entities for which there are textual descriptions that could be used as features for the entities, relations are usually just randomly initialized like in BLP. For example, in Figure 1.1.1 the description of the entity *dbr:chatGPT* contains the information that chatGPT is built on top of openAI's GPT-3. In order to accurately predict the missing triple or fact <*dbr:chatGPT*, *dbo:follows*, *dbr:GPT-3*>, where the relation *dbo:follows* is not observed while training, an LP approach is required which is capable of effectively utilizing this information.

  Moreover, due to the fact that there is a lack of an inductive LP approach that addresses unseen relations, there also exists no benchmark dataset which can be utilized to perform the evaluation of LP with unseen relations.

Motivated by the research gaps identified above, in this thesis, novel KGE approaches which utilize literals are proposed along with high-quality benchmark datasets for both transductive and inductive settings. A detailed discussion of the formulated research objectives and the significant contributions made are discussed in detail in the subsequent sections.

1.2 RESEARCH OBJECTIVES

In this thesis, based on the shortcomings of the existing KGE approaches in making use of literals as discussed in Section 1.1, the following hypotheses are defined.

- **Hypothesis 1** *Combining contextual information about relations and their corresponding textual descriptions is crucial to learn representation for unseen relations when performing inductive LP.* The benefits of combining text-based and graph-based features for relations are thoroughly investigated in the inductive LP approach with unseen relations.

- **Hypothesis 2** *High-quality benchmark datasets containing literals would facilitate the proper evaluation of Multimodal KGE models, specifically for the task of LP in both transductive and inductive settings.* Several benchmark datasets are created from Wikidata KG with extensive experiments for evaluations in a transductive setting. Moreover, a standard dataset to facilitate inductive LP with unseen relations is also generated.
- **Hypothesis 3** *Numerical features generated based on graph structure and literals provide relevant information to learn better representations for entities in the task of transductive LP.* Property paths leading to literal nodes are generated and incorporated into KGE models in a transductive setting.
- **Hypothesis 4** *Multilingual LMs can be leveraged to generate embeddings for entities with textual descriptions in multiple languages to perform LP.* Different Multilingual LMs are leveraged to learn entity embeddings and predict missing links in transductive LP task.

This dissertation aims to address two challenges of LP for KGE: i) Inductive LP in KGs using literals, and ii) Transductive LP in KGs using literals.

- **Challenge 1 (C_1): Inductive LP in KGs using literals:**
  - **C_1-RQ_1**: Can utilizing both graph-based features and description-based embeddings for relations improve State-of-the-art (SoTA) inductive LP models? Moreover, can this approach enable inductive LP with unseen relations?
  - **C_1-RQ_2**: Can the existing inductive LP benchmark datasets with textual literals be extended to perform inductive LP evaluation with unseen relations?
- **Challenge 2 (C_2): Transductive LP in KGs using literals:**
  - **C_2-RQ_1**: How well do the SoTA KGE approaches which use literals perform for the task of LP?
  - **C_2-RQ_2**: How to extract high-quality benchmark datasets from popular KGs such as Wikidata, focusing primarily on literals?
  - **C_2-RQ_3**: Does generating entity features based on property paths, and incorporating these features into existing KGE models result in improving LP tasks?
  - **C_2-RQ_4**: How beneficial is to leverage multilingual embeddings to incorporate multilingual entity descriptions into the task of LP in KGs?

1.3 THESIS OUTLINE AND CONTRIBUTIONS

The rest of this dissertation is comprised of fundamental concepts related to the topics addressed in this thesis, the SoTA works in the area of KGEs with literals in both inductive and transductive settings, the proposed contributions to the hypotheses defined above, and finally concluding remarks. Chapter 2 presents the several fundamental concepts required to understand the proposed methodologies as well as the SoTA models. A comprehensive literature review along with extensive experiments on the SoTA transductive LP models with literals is provided in Chapter 3 and also a detailed literature review on inductive LP is presented in Chapter 4. These literature reviews also discuss the shortcomings of the existing models. The two challenges discussed above and the research questions associated with them are addressed in different chapters of this thesis. Figure 1.3.1 presents an overview of the associations between the chapters and the research questions. Besides, the contributions of this dissertation are summarized as follows:

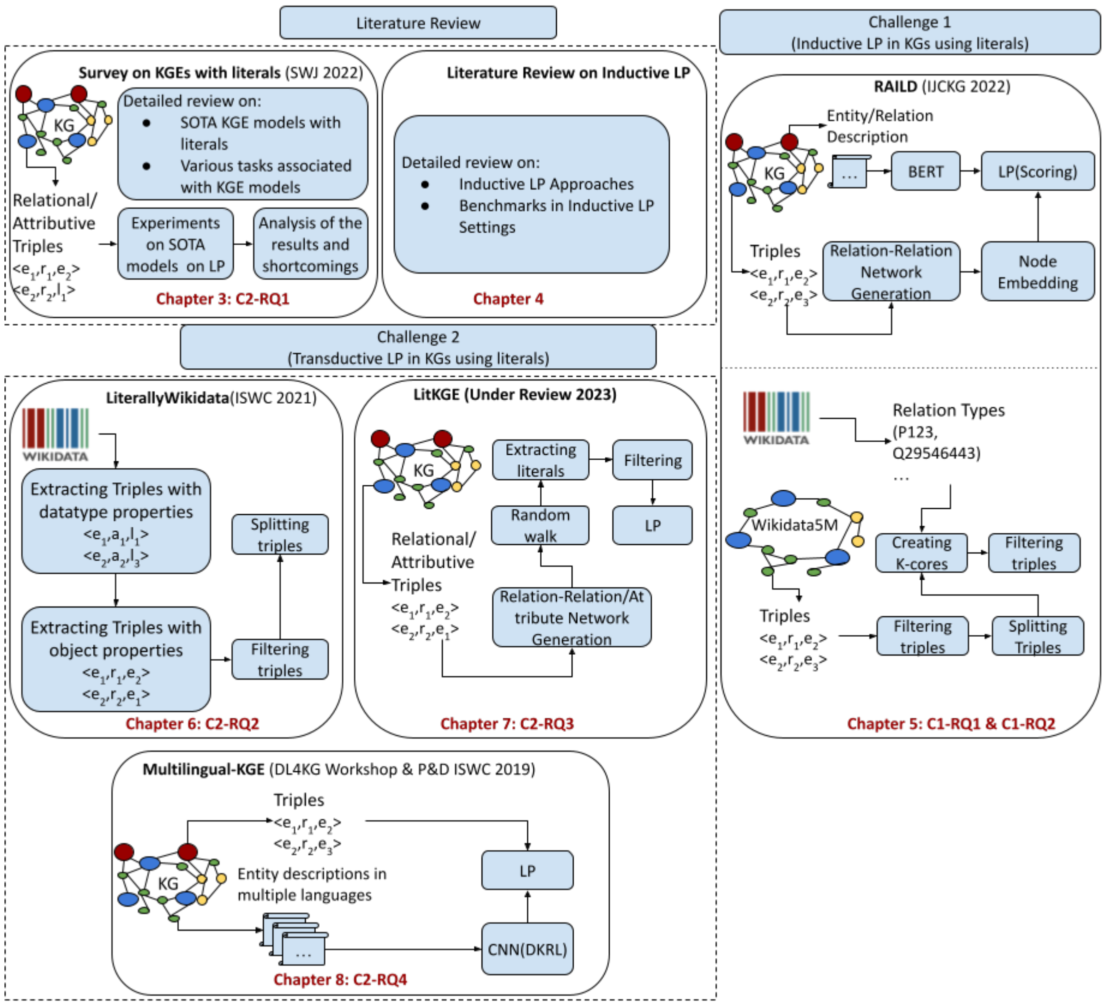

Literature Review
Challenge 1 (Inductive LP in KGs using literals)
Survey on KGEs with literals (SWJ 2022)
Detailed review on: SOTA KGE models with literals; Various tasks associated with KGE models
Relational/ Attributive Triples
Experiments on SOTA models on LP
Analysis of the results and shortcomings
Chapter 3: C2-RQ1
Literature Review on Inductive LP
Detailed review on: Inductive LP Approaches; Benchmarks in Inductive LP Settings
Chapter 4
RAILD (IJCKG 2022)
Entity/Relation Description
BERT
LP(Scoring)
Relation-Relation Network Generation
Node Embedding
Challenge 2 (Transductive LP in KGs using literals)
LiterallyWikidata(ISWC 2021)
Extracting Triples with datatype properties
Extracting Triples with object properties
Splitting triples
Filtering triples
Chapter 6: C2-RQ2
LitKGE (Under Review 2023)
Extracting literals
Filtering
Random walk
LP
Relation-Relation/Attribute Network Generation
Chapter 7: C2-RQ3
Relation Types (P123, Q29546443) ...
Wikidata5M
Creating K-cores
Filtering triples
Splitting Triples
Chapter 5: C1-RQ1 & C1-RQ2
Multilingual-KGE (DL4KG Workshop & P&D ISWC 2019)
Entity descriptions in multiple languages
CNN(DKRL)
Chapter 8: C2-RQ4

Figure 1.3.1: Outline of the thesis.

- **Challenge 1 (Inductive LP in KGs using literals)** :

  - **C1-RQ1** is addressed in Chapter 5 by introducing a novel inductive LP model named *RAILD* which applies a pre-trained *BERT* model to encode entities and relations using their corresponding textual descriptions, as well as a feature generator component that is based solely on graph structure to encode relations. Hence, two kinds of vectors are generated as features for relations, i.e., text-based and graph-based. The graph-based encoder employs a novel algorithm called *WeiDNeR* to create a relation-relation network, which serves as input for the *Node2Vec* [20] node embedding algorithm to generate latent features/embeddings. Given a triple with a relation *r*, the two feature vectors generated for *r* are concatenated into a single vector. Then, the resulting head, tail, and relation vectors are passed to an LP scoring function.
  - The contribution made to address **C1-RQ2** is provided in Chapter 5, which is a novel benchmarking pipeline that takes triples as input along with relation types from Wikidata, and generates a dataset with train, test, and validation splits with a mutually exclusive set of relations in these splits. In order to evaluate the *RAILD* approach introduced in this thesis with unseen relations, a new dataset named *Wikidata68K* is created using this pipeline, by taking Wikidata5M [19] dataset as input along with the relation types from Wikidata for the relations found in Wikidata5M.

- **Challenge 2 (Transductive LP in KGs using literals)** :
  - **C2-RQ1** is addressed in Chapter 3, with a comprehensive survey conducted on the existing KGE models which leverage literals. This survey also contains extensive experiment-based comparisons in order to better analyze and understand the capability of these models for the task of LP. Specifically, these experiments are conducted with multimodal models which use numerical literals, text literals, etc. Moreover, it discusses in detail the shortcomings of each of the existing models. This survey including the results obtained is reported in [1] as a journal.
  - **C2-RQ2** is answered in Chapter 6 by providing a set of high-quality KG completion benchmark datasets extracted from Wikidata and Wikipedia, named *LiterallyWikidata*. It is generated with a special focus on multimodal KG Embedding (KGE) models, specifically for models using numeric and/or text literals. It contains three novel datasets which vary both in size and structure. Besides, benchmarking experiments on the task of LP have been conducted on these datasets with extensively tuned unimodal/multimodal KGE models.
  - A novel approach named LitKGE created to tackle **C2-RQ3** is presented in Chapter 7. It utilizes a modified version of the WeiDNeR algorithm (i.e., the relation-relation network generator algorithm proposed in RAILD) called *WeiDNeR_Extended* which is enabled to handle numerical attributes. The approach starts by using WeiDNeR_Extended to create a relation-relation/attribute network from a given knowledge graph followed by generating property paths by using a

random-walk strategy. Then, it extracts literals associated with entities through these property paths and applies a filtering mechanism to obtain high-quality features for the entities. Finally, the numerical features are leveraged to perform the task of LP with literals.

– **C2-RQ4** is addressed in Chapter 8 by exploring the benefits of leveraging multilingual entity descriptions for LP task on KGs. Specifically, the performance of the existing model DKRL in leveraging multilingual descriptions using multilingual embeddings has been analyzed and the results of the experiments are discussed. The languages considered are English, German, and French whereas MUSE [21] is used to encode the descriptions.

Note that, as it can be observed in Section 1.1, Chapter 3, and Chapter 4, transductive LP is discussed first and then follows inductive LP. This is because the majority of the SOTA inductive LP methods are designed to address the limitations of previously introduced transductive LP methods in handling entities that are not present during training. Conversely, in the rest of the chapters inductive LP is addressed followed by transductive LP. It is organized in this manner due to the fact that the *WeiDNeR_extended* algorithm of the transductive LP model *LitKGE*, is developed by extending the *WeiDNeR* algorithm introduced as part of the *RAILD* model proposed for inductive LP.

2 FOUNDATIONS

In this chapter, a brief introduction to the fundamental concepts and notation that will be leveraged extensively in this thesis is provided. It covers Graphs and KGs in Section 2.1 and Section 2.2 respectively, Deep Neural Networks (DNNs) in Section 2.3, Neural Language Models (NLMs) in Section 2.4, Network Embeddings (NEs) in Section 2.5, KGC and KGE in Section 2.7 and Section 2.6 respectively. Finally, in Section 2.8, the various evaluation metrics that are used in this thesis are discussed.

2.1 GRAPHS

A graph is a mathematical representation used to model the relationship between objects. The origin of graph theory can be traced back to 1741 when Leonhard Euler first proposed it as a solution to the Seven Bridges of Königsberg Problem [22]. Over the years, graphs have been widely used in fields such as mathematics, computer science, and physics to comprehend complex phenomena and address issues involving organization, connectivity, optimization, and matching. With the rise of massive social networks, the significance of graph theory has become even more prominent as it facilitates the understanding and analysis of the relationships between different entities. A graph consists of objects, depicted as nodes or vertices, and the relationships between these objects, represented as edges connecting pairs of nodes.

Definition 1 (Graph)
A graph G is an ordered pair and is given by $G = (V, E)$*, where* V *is the set of nodes or vertices and* $E \subseteq \{(u, v) | u, v \in V\}$ *is the set of edges between the nodes.*

- *If* $(u, v) \in E$ *is an edge in G, u is referred to as being adjacent to v.*
- *Two edges* $(u, v), (x, y) \in E$ *are referred to as adjacent if* $u = x$ *or* $u = y$ *or* $v = x$ *or* $v = y$*, i.e., the two edges share a common vertex in* G.

In general, graphs vary based on the connectivity between the nodes and the nature of the edges. They can be categorized into 4 types, namely, *undirected simple graph*, *directed graph*, *undirected multigraph*, and *directed multigraph* as shown in Figure 2.1.1. These graph variants are defined as follows.

Definition 2 (Undirected Simple Graph)
An Undirected Simple Graph G *is an ordered pair defined as* $G = (V, E)$ *with no parallel edges and no self-loops (an edge connecting a vertex to itself) given by* $(u, v) \in E \leftrightarrow (v, u) \in E$.

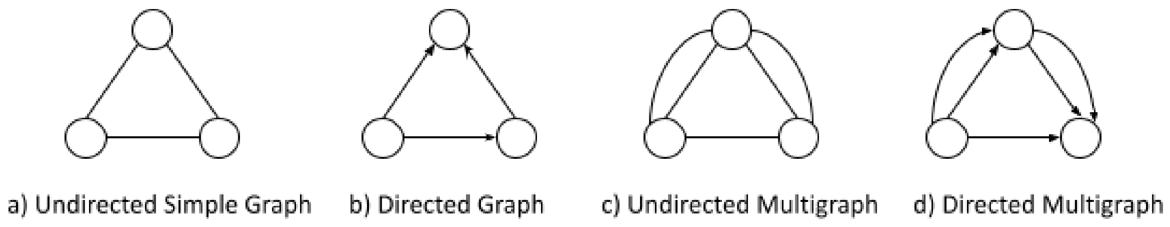


Figure 2.1.1: Different variants of graphs

Definition 3 (Directed Graph)
A Directed Graph G is an ordered pair $G = (V, E)$*, where V is the set of nodes and* $E = \{(u, v)|(u, v) \in V^2\}$ *is a finite set of directed edges, also called arcs, each of which is an ordered pair of vertices from V.*

Definition 4 (Undirected Multigraph)
An Undirected Multigraph G is an ordered triple $G = (V, E, R)$*, where V is the set of nodes and E is the set of edges.* $\phi : E \to \{(u, v)|u, v \in V\}$ *is a function that assigns to each edge an unordered pair of endpoint nodes.*

Definition 5 (Directed MultiGraph)
A Directed MultiGraph G *is an ordered pair* $G = (V, E)$*, where* V *and* E *denote the set of nodes and directed edges respectively. It consists of multiple edges or arcs with the same source node* $u \in V$ *and the same target node* $v \in V$*. It may also contain self-loops.*

Figure 2.1.1 part a) is an illustration of an undirected simple graph composed of three nodes ($|V| = 3$) and every node has a maximum of two degrees ($|V| - 1 = 2$) where $|V|$ refers to the total number of nodes in the graph. Part b) is a directed graph, characterized by the presence of arcs between source and target nodes. In contrast, part c) represents an undirected multigraph, which includes parallel edges. Finally, part d) is a simple example of a directed multigraph that contains directed edges.

2.2 KNOWLEDGE GRAPHS

In 1972, the term "Knowledge Graph" (KG) was initially mentioned in literature [23]. It was later reintroduced by Google with the launch of Google Knowledge Graph in 2012 and since then, KG has become a key factor in advancing the field of Artificial Intelligence. A KG, as defined in [24], is a graph-based representation of real-world entities and their interrelations, incorporating a schema that defines entity classes and relations, facilitating potential interrelations between arbitrary entities, and covering various topical domains. Some of the widely used open KGs include DBpedia [25], YAGO [26], Freebase [3], and Wikidata [2]. These KGs are either curated by human editors, extracted through automated or semi-automated methods, or generated using heuristics. Moreover, various enterprise KGs are also being used in different industries for the purpose of web search, commerce, finance, social networks and so on [10].

Definition 6 (Knowledge Graph)
A KG $\mathcal{G}$ is a directed labeled graph consisting of a set of triples $\mathcal{T}$, given by, $\mathcal{T} \subseteq (\mathcal{E} \cup \mathcal{C}) \times \mathcal{R} \times (\mathcal{E} \cup \mathcal{C} \cup \mathcal{L})$ where $\mathcal{E}$ is a set of resources referred to as entities, $\mathcal{C}$ is is the set of semantic types or classes of the entities, $\mathcal{L}$ a set of literals, and $\mathcal{R}$ a set of relations. An entity is identified by a URI which represents a real-world object or an abstract concept. A relation (or property) is a binary predicate and a literal can be a string, date, or number eventually followed by its data type.

Definition 7 (Triple)
A triple $< e_h, r, e_t > \in \mathcal{T}$ in a KG $\mathcal{G}$, is an ordered set, where $e_h \in \mathcal{E} \cup \mathcal{C}$ is the subject, $r \in \mathcal{R}$ is the relation, and $e_t \in \mathcal{E} \cup \mathcal{C}$ is referred to as tail entity. The subject and object are often referred to as *head* and *tail* entity respectively. The triples consisting of literals as objects, i.e., $e_t \in \mathcal{L}$ are known as *attributive triples*.

Relations (or Properties): Based on the nature of the objects, relations are classified into two main categories:

- **Object Relation** links an entity to another entity. E.g., in the triple `<dbr:Albert_Einstein, dbo:field, dbr:Physics>`, both `dbr:Albert_Einstein` and `dbr:Physics` are entities, the relation `dbo:field` is an *Object Relation*.
- **Datatype Relation** links an entity to its values, i.e., literals. For example, in `<dbr:Albert_Einstein, dbo:birthDate, "1879-03-14">`, where `"1879-03-14"` is a literal value, the relation `dbo:birthDate` is a *Datatype Relation*.

Literals: The literals in a KG encode additional information which in general can not be represented by the entities or relations. As described in [10], literals allow for representing strings (with or without language tags) and other datatype values (integers, dates, etc.). The different types of literals present in a KG are given as follows:

- **Text:** A wide variety of information can be stored in KGs in the form of free text such as names, labels, titles, descriptions, comments, etc. In most of the KG embedding models with literals, text information is further categorized into ***Short text*** and ***Long text***. The literals which are fairly short such as for relation like names, titles, labels, etc. are considered as *Short text*. On the other hand, for strings that are much longer such as descriptions of entities, comments, etc. are considered as *Long text* and are usually provided in natural language.
- **Numeric:** Information encoded as integers, float and so on such as height, date, population, etc. also provide useful information about an entity. It is worth considering

the numbers as distinct entities in the embedding models, as it has its own semantics to be covered which cannot be covered by string distance metrics. For instance, 777 is more similar to 788 than 77.

- **Units of Measurement:** Numeric literals often denote units of measurement to a definite magnitude. For example, Wikidata property `wdt:P2048` ("height") takes values in mm, cm, m, km, inch, foot and pixel. Hence, discarding the units and considering only the numeric values without normalization results in loss of semantics, especially if units are not comparable, e.g., units of length and units of weight.

- **Image:** Images also provide latent useful information for modeling the entities. For example, a person's details such as age, gender, etc. can be deduced via visual analysis of an image depicting the person.

- **Others:** Useful information encoded in the form of other literals such as external URIs which could lead to an image, text, audio or video files.

The datasets used for the evaluation of the proposed approaches in this thesis, as presented in Chapter 5, 7, and 8, are extracted from the open KGs, Wikidata, Freebase, YAGO, and WordNet. Additionally, Wikidata is used as a source to extract the KGC benchmark datasets introduced in this thesis, as discussed in Chapter 6. Details pertaining to these KGs are presented below.

WIKIDATA Wikidata is a free, open, and multilingual KG that was launched in 2012 by the Wikimedia Foundation [2]. It enables collaboration to gather and present structured data supporting various knowledge domains and projects. Wikidata contains information on various topics such as people, organizations, events, places, and concepts. It acts as a central data management platform for linking data across various language versions of Wikipedia, most of its sister projects, external databases, and resources. Wikidata is constantly evolving and growing, with contributions from a diverse community of users and developers around the world. As of the 1st of August 2023, Wikidata contains 105,599,208 items and 1,943,680,661 edits have been made since its launch[1].

YAGO YAGO is an open KG that has been constructed from Wikipedia (like Categories, Redirects, infoboxes), WordNet (e.g., synsets, hyponymy), and GeoNames in 2007. As of 2019, it encompasses over 10 million entities (including persons, organizations, cities, etc.) and contains over 120 million facts related to these entities [27]. The accuracy of YAGO has been manually evaluated, proving a confirmed accuracy of 95% [27, 28].

FREEBASE Freebase [3] Freebase was a large collaborative KG developed and launched by Metaweb in 2007. It consists of structured data (more than 125 million triples and more

1 https://www.wikidata.org/wiki/Wikidata:Statistics

than 7000 properties) composed mainly by its community members, i.e., it was harvested from many sources, including individual, user-submitted wiki contributions. Metaweb was later acquired by Google in 2010, and ultimately, the Freebase API was terminated in 2016.

WORDNET WordNet [29] is a large lexical database of English words in more than 200 languages, which organizes nouns, verbs, adjectives, and adverbs into groups of related words called synsets, which correspond to specific meanings or concepts. These synsets are connected to each other through both conceptual-semantic and lexical relationships, forming a network of interlinked words. Synonymy, antonymy, hyponymy, and meronymy are among the semantic relations used to determine word definitions. WordNet contains 155,327 words organized in 175,979 synsets for a total of 207,016 word-sense pairs. It was first released in the 1980s in English only in the Cognitive Science Laboratory of Princeton University.

2.3 NEURAL NETWORKS

Neural networks, which are a subset of machine learning and a fundamental component of deep learning algorithms, are also referred to as simulated neural networks (SNNs) or artificial neural networks (ANNs). Their structure and nomenclature are inspired by the human brain, replicating the way biological neurons communicate with each other. In 1943, the first computational model of neural networks was designed by McCulloch and Pitts [30]. Later in 1958, Frank Rosenblatt created the first perceptron for pattern recognition. It was a type of neural network that could learn and recognize patterns in input data and was used in a variety of applications, including image recognition and natural language processing.

An activation function is a mathematical function used in neural networks to determine whether a neuron should be activated or not. The activation function operates on the weighted sum of the inputs to a neuron (further with bias), called the activation, and produces an output that is then passed on to the next layer of the network. The most commonly used activation functions in neural networks are shown below and the corresponding illustrations are provided in Figure 2.3.1.

SIGMOID The Sigmoid function is a popular S-shape non-linear activation function that transforms its inputs into the interval of values between 0 and 1. It is effective for modeling binary classification problems and is defined as follows:

$$\mathrm{sigmoid}(x) = \frac{1}{1+e^{-x}} \tag{1}$$

RELU The ReLU (Rectified Linear Unit) function is a non-linear activation function (also known as a ramp function) that returns the input if it is positive and 0 otherwise and it is given by:

$$\mathrm{ReLU}(x) = \max(0, x) \tag{2}$$

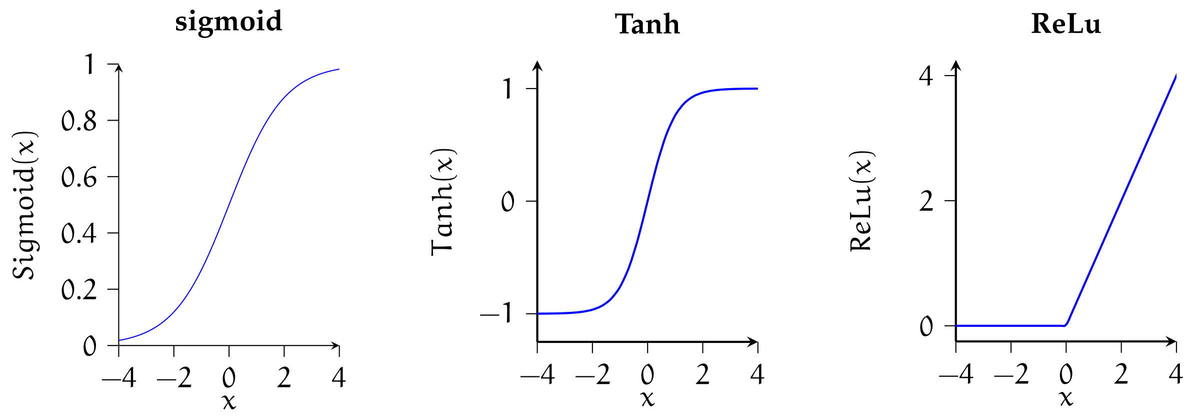


Figure 2.3.1: Activation functions

Due to its lower computational complexity, ReLU is preferred over sigmoid and tanh for complex deep neural networks with large datasets and numerous neurons.

TANH The Tanh (Hyperbolic Tangent) activation function is similar to the sigmoid function but produces output values between -1 and 1. Differently from sigmoid, the Tanh function is mostly used only in the hidden layers, as the output values of the Tanh function cannot be interpreted as probabilities. Note that Tanh is useful for modeling problems with symmetric data around zero.

$$\mathrm{Tanh}(x) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{3}$$

Over the years, more sophisticated types of ANNs are developed to handle more complex data. In the subsequent sections, the neural networks Feed-Forward Neural Network (FFNN), Convolutional Neural Network (CNN), and Gated Recurrent Unit (GRU) that are used in this thesis, as presented in Chapter 5, 7, and 8, are discussed.

2.3.1 *Feed-Forward Neural Network*

A Feed-Forward Neural Network (FNN) is a type of ANN where the connections between nodes do not form a cycle. The simplest form of FNN is referred to as a single-layer perceptron (SLP), where the inputs are fed directly to the outputs via a series of weights. The process involves multiplying each input with its corresponding weight, summing up the results, adding a bias term, and then passing the resulting value through an activation function. The SLP is an important model of FNN and is often used in classification tasks. However, SLPs are incapable of handling non-linearly separable data. Hence, Multi-layer perceptron (MLP) is proposed to deal with the limitations of SLPs.

The basic architecture of an MLP is composed of an input layer, one or more hidden layers, and an output layer. Each layer is made up of one or more neurons that perform

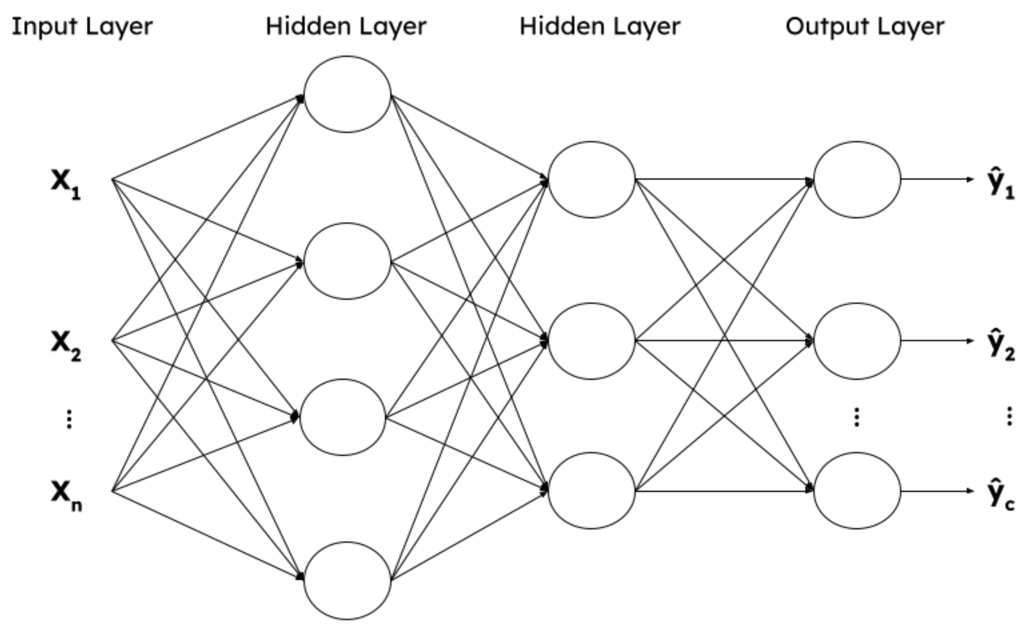


Figure 2.3.2: A simple MLP with an input layer, two hidden layers and an output layer

mathematical operations on the input data. The input layer receives the initial data, which is then transmitted to the next layer, and this process continues until the output layer is reached. The hidden layers process the input data and generate intermediate representations that are passed into the next layer until the output layer is reached. The output layer produces the final output of the network. A basic MLP is depicted in Figure 2.3.2 containing an input layer, two hidden layers, and an output layer. The inputs are given by x_1, x_2, ..., x_n where n is the total number of input and the output is given by y_1, y_2, ..., y_C, where C is the total number of classes. Formally, the activation of the hidden units j of the k-i^{th} layer, z_j^k, is computed as follows:

$$z_j^k = h(\sum_{i=1}^{n} w_{ji}^k x_i + b_{j0}^k) \tag{4}$$

Where, k is the number of a given layer, i ranges from 1 to N (the dimension of the input vector), w_{ji} are the weights, and b_{j0} is the bias. The term a_j^k, is the j-ith input activation of the k-ith layer.

2.3.2 *Convolutional Neural Network*

A neurocognitive machine was proposed in 1984 that was based on the concept of receptive field, and is considered the first realization of the Convolutional Neural Network (CNN), as well as the initial application of the receptive field idea in artificial neural networks [31].

The architecture of CNN draws its inspiration from visual perception, as noted in [32]. CNNs have played a significant role in advancing Image Classification and are now widely used in Computer Vision systems. In recent years, various applications of CNNs to address issues in Natural Language Processing have been witnessed [33]. However, in this section, the fundamental working principle of CNN is discussed as follows.

A CNN architecture is composed of several stages, each of which consists of multiple layers that work together to extract features at varying levels of abstraction. Typically, each stage of a CNN consists of three layers: (i) A convolutional layer, (ii) A ReLU layer, and (iii) A pooling layer. In the convolutional layer, a convolution operation is performed on the input data to extract features. In the ReLU layer, the RELU activation function that sets negative values in the feature maps to zero and preserves positive values is applied to the output of the convolutional layer. This enables introducing non-linearity into the network. A pooling function, such as MAX-pooling or AVG-pooling, is used in the pooling layer to downsample the feature maps, resulting in a decrease in the computational complexity of the network and an increase in its robustness to variations in the input.

A single convolutional layer has n_1 filters, where the number of filters applied in one stage matches the depth of the volume of the output feature maps. At layer l, the output $Y_i^{(l)}$ consists of $n_1^{(l)}$ feature maps of size $n_2^{(l)} \times n_3^{(l)}$. The i^{th} feature map $Y_i^{(l)}$ is computed as folows:

$$Y_i^{(l)} = \sum_{j=1}^{n_1^{(l)}} K_{(i,j)}^{l} \times Y_{(j)}^{l-1} + B_i^{l} \tag{5}$$

where B_i^l is the bias matrix, $K_{(i,j)}^l$ is the filter connecting the j^{th} feature map in layer $(l-1)$ with i^{th} feature map in the layer.

2.3.3 *Gated Recurrent Unit*

In 2014, Kyunghyun Cho et al. introduced gated recurrent units (GRUs) as a gating mechanism in recurrent neural networks [34]. The purpose of GRUs is to solve the vanishing gradient problem which comes with a standard recurrent neural network. To do so, GRU uses the update gate and reset gate which are very similar to the forget gate and input gate of short-term memory (LSTM). However, GRU has fewer parameters than LSTM since it lacks an output gate. The update gate plays a crucial role in determining how much information from the previous time step should be passed on to the current time step. It takes the current input and the previous hidden state as input, and outputs a value ranging from 0 to 1, indicating the proportion of information to retain from the previous hidden state. Conversely, the reset gate decides how much of the previous hidden state should be discarded in favor of the current input. It takes the current input and the previous hidden state as input, and outputs a value ranging from 0 to 1 that represents the amount of information to forget from the previous hidden state. For every element within the input sequence in a GRU network, each layer of the network calculates the following function:

$$
\begin{aligned}
z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) \\
r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r) \\
n_t &= \tanh(W_h x_t + U_h(r_t \odot ht - 1) + b_h) \\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot n_t
\end{aligned} \tag{6}
$$

where x_t, h_t, z_t, r_t, n_t are input vector, output vector, update gate vector, reset gate vector, candidate activation vector respectively, and W, U, b are parameter matrices and vector respectively, h_{t-1} is the hidden state of the layer at time $t-1$ or the initial hidden state at time 0, σ denotes the sigmoid function, and $\odot$ is the element-wise product.

2.4 LANGUAGE MODELS

A language model (LM) assigns a probability distribution to a sequence of words, which is learned by training the model on text corpora in one or multiple languages. The challenge with LMs is that given any sequence of words of length n, to assign a probability $P(w_1, \ldots, w_n)$ to the whole sequence that may not have been seen in the training data. Various modeling approaches have been created to tackle this issue, including the implementation of the Markov assumption which claims that the distribution of a word depends on some fixed number of words that immediately comes before it or the utilization of neural architectures like recurrent neural networks or transformers. As presented in [35], NLMs can be divided into two broad categories: (i) Non-contextual Embeddings and (ii) Contextual Embeddings. Their dissimilarity lies in whether the embedding for a word undergoes dynamic changes depending on its contextual usage.

2.4.1 *Non-contextual embeddings*

In Non-contextual embedding, methods represent each word or subword as a fixed dense vector. Formally, a vector $e_x \in R^{D_e}$ is assigned to each word (or subword) x in a vocabulary V using a lookup table $E \in R^{D_e \times |V|}$, where R^{D_e} represents the dimension of token embeddings. Such embeddings are static by nature, meaning that the embedding for a word remains the same irrespective of its context, making it challenging to represent polysemous words. Word2Vec [36], GloVe [37], FastText [38] are among the most common non-contextual embedding methods. The basic principles of GloVe and FastText are discussed in the following sections since they are utilized in this thesis, i.e., specifically in Chapter 5 and Chapter 8, respectively.

GLOVE The Global Vectors for Word Representation (GloVe) algorithm is an unsupervised learning technique that obtains vector representations for words. To achieve this, it trains on global word-word co-occurrence statistics from a corpus, and the resulting representations exhibit linear substructures within the word vector space. The weighted least squares objective J employed by GloVe aims to minimize the difference between the dot product of the vectors of two words and the logarithm of their co-occurrence count.

$$J = \sum_{i,j=1}^{V} f(X_{ij})\left(\mathbf{w}_i^T \tilde{\mathbf{w}}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2 \tag{7}$$

where w_i and w_j are the word vector of word i and the context word vector of word j respectively, b_i and b_j are the biases of word i and j respectively, X_{ij} is the number of times word i occurs in the context of word j, and f is a weighting function that assigns lower weights to rare and frequent co-occurrences, as they tend to be less informative for capturing the semantic relationships between words.

FASTTEXT One significant drawback of Word2vec and GloVe is their inability to generate embeddings for words that are not present in their vocabulary, also known as Out-of-Vocabulary (OOV) words. The Word2vec model consists of a shallow neural network architecture with two hidden layers, which takes in a large corpus of text data as input and learns the word embeddings by predicting the surrounding words in a given context window. The fastText embedding technique extends Word2Vec by leveraging subword information to create embeddings for words. It accomplishes this by learning representations of character n-grams and then summing the n-gram vectors to represent a word. This approach enhances the word2vec method by including subword details, allowing the embeddings to capture prefix and suffix information. After representing a word using character n-grams, a skip-gram model is employed to learn the embeddings.

2.4.2 *Contextual embeddings*

In order to tackle the challenge of words having multiple meanings and being influenced by context, it is necessary to differentiate the semantics of words based on their specific contexts. Given a text of tokens $t_1, t_2, \ldots, t_n$ where each token $t_i \in V$ is a word or subword, the contextual representation of t_i denoted by h_i is dependent on the surrounding words and it is computed as follows:

$$[h_1, h_2, \ldots, h_n] = f_{enc}(t_1, t_2, \ldots, t_n), \tag{8}$$

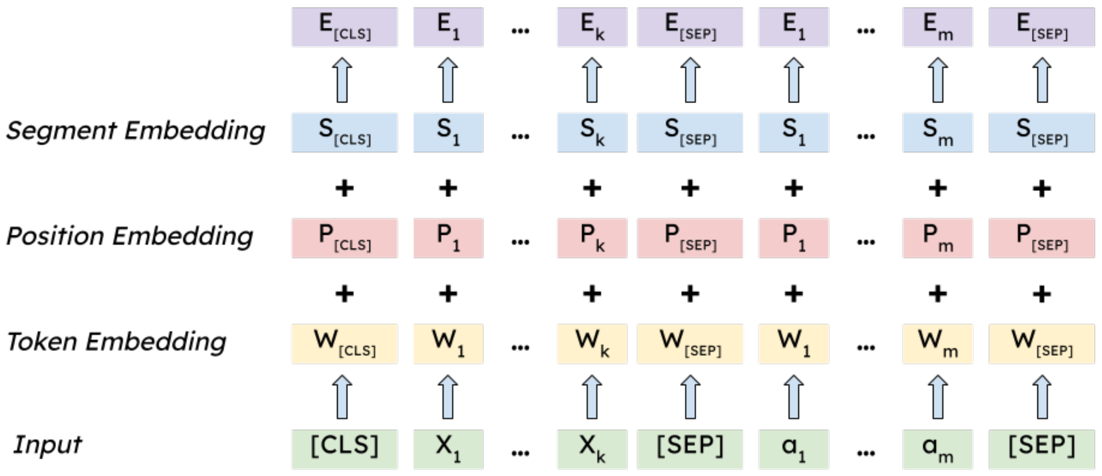
where $f_{enc}(\dot{})$ is a newural encoder. The subsequent sections provide in-depth explanations of the contextual embedding models *BERT* and *DistilBERT*, utilized in the models proposed in this thesis.

BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS (BERT) BERT is a pre-trained natural language processing (NLP) model introduced by Google in 2018 and has since become one of the most popular and powerful NLP models available. BERT is specifically designed to pre-train deep bidirectional representations from unlabelled text, taking into account both the left and right context in all layers. Unlike its predecessors, which were either unidirectional or partially bidirectional, BERT is fully bidirectional, allowing it to simultaneously examine the left and right context of a word. BERT's framework comprises two main steps: pre-training and fine-tuning. During pre-training, the model is trained on unlabeled data using different pre-training tasks. To fine-tune the BERT model, it is first initialized with pre-trained parameters, and all parameters are then fine-tuned using labeled data from downstream tasks.

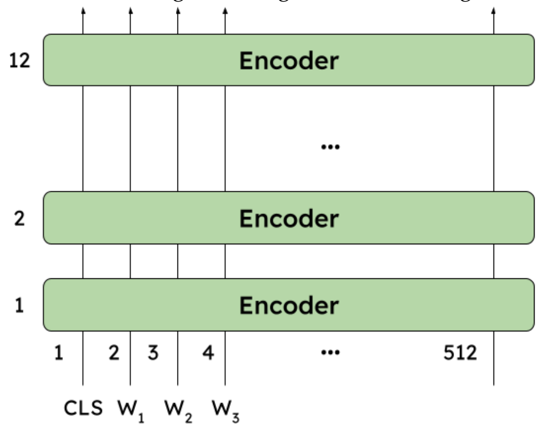
The architecture of BERT's model is a multi-layer, bidirectional Transformer encoder based on the original implementation described in [39]. The original BERT trained on the English language has two model sizes: $\text{BERT}_{\text{BASE}}$ (number of layers L=12, hidden size H=768, number of self-attention heads A=12, Total Parameters=110M) and $\text{BERT}_{\text{LARGE}}$ (L=24, H=1024, A=16, Total Parameters=340M). The input to BERT is a sequence of tokens, which may be a single sentence or two sentences packed together. The first token of every sequence is always a special classification token ([CLS]). The final hidden state corresponding to this token is used as the aggregate sequence representation for classification tasks. The sentences are differentiated in two ways: first, by separating them with a special token ([SEP]) and then, by adding a learned embedding to every token indicating whether it belongs to sentence A or sentence B. The input representation of a token is created by adding together its respective *token, segment, and position embeddings* as shown in Figure 2.4.1a. The purpose of adding positional embedding to each token is to indicate its position in the sentence. Note that each layer applies self-attention, and passes its results through a feed-forward network, and then to the next encoder. The BERT-base model with 12 encoder layers is depicted in Figure 2.4.1b, where the model input consists of a sequence of n words (w1,w2, ...,wn) and special tokens ([CLS] and [SEP]).

BERT is trained using two different unsupervised tasks: i) Masked Language Modeling (MLM) task and ii) Next Sentence Prediction (NSP) task. In the MLM task, only 15% of the input tokens are masked at random, and then those masked tokens are predicted in order to train a deep bidirectional representation. For this scenario, the output softmax over the vocabulary is fed with the final hidden vectors that correspond to the mask tokens, similar to a standard language model. In the NSP task, the model is trained in order to predict whether two sentences are consecutive in a text corpus, i.e., it involves feeding the model two consecutive sentences, and asking it to predict whether the second sentence is a plausible continuation of the first sentence, or whether it is a random sentence from the training corpus. The BERT model can be easily fine-tuned and has been utilized for numerous tasks, such as sentiment analysis, question answering, paraphrase identification, natural language inference, etc. Furthermore, Chapter 5 of this dissertation shows that

BERT plays a significant role in capturing the semantics present in entity and relation descriptions and predicting missing links in a KG.


Segment Embedding
Position Embedding
Token Embedding
Input
E[CLS] E1 ... Ek E[SEP] E1 ... Em E[SEP]
S[CLS] S1 ... Sk S[SEP] S1 ... Sm S[SEP]
P[CLS] P1 ... Pk P[SEP] P1 ... Pm P[SEP]
W[CLS] W1 ... Wk W[SEP] W1 ... Wm W[SEP]
[CLS] X1 ... Xk [SEP] a1 ... am [SEP]

a. The embeddings of the BERT input sequence are the sum of the token embeddings, positional embeddings, and segment embeddings.



b. Illustration of the BERT model

Figure 2.4.1: BERT input sequence embeddings and model illustration

DISTILED BERT (DISTILBERT) DistilBERT is a pre-trained language model introduced by Hugging Face in 2019. It is a smaller and faster version of the BERT model, which achieves state-of-the-art performance on various natural language processing tasks. The model achieves this by using a distillation technique, where a smaller student model, DistilBERT, is trained to mimic the behavior of the larger BERT model, which serves as the teacher model. The DistilBERT model has 40% fewer parameters than the original BERT model while retaining 97% of its language understanding capabilities and being 60% faster, making it more efficient to use in applications with limited computational resources.

Despite its smaller size and faster speed, DistilBERT achieves comparable or even better performance than BERT on various NLP tasks such as sentiment classification and question answering. Moreover, Chapter 5 in this thesis investigates the performance of DistilBERT for LP and demonstrates comparable outcomes with SoTA models.

2.5 NETWORK EMBEDDINGS

Information networks are commonly employed to represent complex relationships spanning various disciplines, such as social networks, citation networks, telecommunication networks, and biological networks. A crucial hurdle related to such networks is identifying a concise and efficient way to represent them so as to facilitate advanced analytical tasks. In order to address this issue, different network embedding techniques have been designed with the purpose of learning low-dimensional latent representations of the nodes present in a network while preserving the network structure. These representations can serve as useful features for a diverse set of graph-related tasks, including visualization, classification, clustering, and link prediction.

As presented in [40], methods for network embedding can be divided into two main categories: unsupervised and semi-supervised. Unsupervised methods do not rely on labeled vertices to generate representations for networks, while semi-supervised approaches use labeled nodes to learn the representations. SemiNE [41], LDE [42], and LANE [43] are some examples of semi-supervised network embedding methods whereas DeepWalk [44], LINE [45], and Node2Vec [20] are among the widely known unsupervised methods. Since Node2Vec is used in Chapter 5 of this thesis, a more detailed explanation of the method is presented as follows.

NODE2VEC Node2Vec learns continuous latent representations for nodes in networks with the likelihood of preserving neighborhood information. It efficiently explores diverse neighborhoods of a given node by using second-order (biased) random walks and then applies the SkipGram [46] word embedding method to learn embeddings by treating the generated walks as sentences. Given a network, Node2Vec selects the next hop by computing second-order transition probabilities using Equation 9.

$$P(u|v,t) = \frac{\alpha_{pq}(t,u)w(u,v)}{\sum_{u' \in N_v} \alpha_{pq}(t,u')w(u',v)} \tag{9}$$

where $u, v \in V$ with V being the set of nodes, N_v denotes the neighboring nodes of v, and $w(u,v)$ is the weight of the edge between the nodes u and v, and α is the bias factor used to reweigh the edge weights depending on the previously visited state and it is computed as shown in Equation 10.

$$Q_{ij} = \begin{cases} \frac{1}{p} & \text{if; } d_{tu} = 0 \\ 1 & \text{if; } d_{tu} = 1 \\ \frac{1}{q} & \text{if; } d_{tu} = 2 \end{cases} \tag{10}$$

where p is the return parameter that controls the likelihood of immediately revisiting a node, q is the in-out parameter controlling the likelihood of revisiting a node's one-hop neighborhood, and d_{tu} is the shortest distance between the nodes t and u. These random walks are then passed to the SkipGram model to learn the node embeddings. Since the SkipGram model aims to learn continuous feature representations for words by optimizing a neighborhood-preserving likelihood objective, in Node2Vec it could be interpreted as aiming to maximize the probability of predicting the correct context node v for a given center node u.

2.6 KNOWLEDGE GRAPH EMBEDDING

Knowledge Graph Embedding (KGE) techniques aim to generate a dense representation of the graph by embedding it into a continuous vector space with fewer dimensions. The generated embeddings can be leveraged for KG refinement tasks such as KG graph completion and triple classification or for various downstream tasks such as question answering and recommendation. The typical approach for knowledge graph embedding involves creating:

- *an entity embedding* for every node, which is represented as a vector with a certain number of dimensions denoted by *'e'*, and
- *a relation embedding* for each edge label, usually represented as a vector with a certain number of dimensions denoted by *'r'*.

The size of the entity and relation vectors is typically limited to a fixed, relatively low range of 50 to 1000 dimensions, as noted in [10]. As noted in [47], the following are typically the steps involved in a KGE model:

1. *Representation space* in which the relations and entities are represented. Pointwise space, manifold, complex vector space, the Gaussian distribution, and discrete space are among the different types of representation learning spaces. In order to ensure the quality of a knowledge graph embedding, the embedding space should follow three essential conditions, i.e., calculation possibility, differentiability, and the possibility of defining the scoring function [48].
2. *Scoring function* which is used to measure the plausibility of triples. In the energy-based learning framework, the scoring function is referred to as the energy function. The two common types of scoring functions in knowledge graph embedding are distance-based functions and similarity-based functions. Corrupted/False/negative

triples are usually assigned lower scores, while true triples (those triples observed in the knowledge graph) tend to have higher scores.

3. *Encoding models* for representing and learning relational interactions by employing specific model architectures such as linear/bilinear models, factorization models, and neural networks. Linear/bilinear mapping is used by linear models to represent relations by putting head entities close to tail entities in the vector space. On the other hand, factorization aims to decompose relational data into low-rank matrices to learn embeddings. Neural networks encode relational data with nonlinear neural activation and more complex network structures by matching the semantic similarity of entities and relations. In neural network-based embedding models, relational data is encoded by applying nonlinear neural activation and more complex network architecture by matching the semantic similarity of entities and relations.

4. *Auxiliary information* to be incorporated into the KGE embedding approaches in order to enhance the representation of knowledge. Multimodal KGE embedding models combine different types of additional information (i.e., auxiliary information), including text literals (textual descriptions of entities), relational paths, images associated with entities, and entity type constraints, with the KG itself in order to enhance the representation of knowledge. Typically in order to consider such auxiliary information when learning embeddings, an ad-hoc scoring function is created for the supplementary data and then incorporated into the overall scoring function.

2.7 KNOWLEDGE GRAPH COMPLETION

Since most KGs are produced using manual, semi-automatic, or automatic methods, a considerable amount of implicit entities and relationships have not been recognized, leading to incompleteness in KGs. Consequently, the goal of knowledge graph completion (KGC) is to address this issue by adding the missing triples (links) to a KG, which are considered accurate but are not currently present in the KG. This task is often addressed with LP methods which aim to estimate the likelihood of the existence of links between entities based on the current observed information in the KG [47]. Taking into account the nature of the missing links, the task of LP can be further divided into the following three types of prediction problems.

- *Entity Prediction* (also known as head and tail prediction) is one of the most common kinds of LP tasks for KGC. It is formulated as predicting the head entity h given the relation and tail entity *(?,r,t)* or predicting the tail entity t given the relation and head entity *(h,r,?)* where "?" represents a missing entity. For example, in refernece to Figure 1.1.1, the missing link <*dbr:GPT-2, genre, dbr:Autoregressive*> could be generated through a head prediction <*?, genre, dbr:Autoregressive*> or a tail prediction <*dbr:GPT-2, genre, ?*> task.

- *Triple Classification* is a task that verifies whether a given triple ⟨h, r, t⟩ is valid or not. The task involves training a binary classifier to identify the validity of the triple, i.e., classifying the triple as true/valid (1) or as false/invalid (0). For example, in refernece to Figure 1.1.1, triple classification helps in identifying if the triple *<dbr:GPT-2, genre, dbr:Autoregressive>* is a true triple for the KG or not.
- *Entity Type Prediction* (also known as entity classification) aims to classify entities into different semantic types and it is given by *<e, type,?>*, where *e* is the entity.

As discussed in [47], LP models can be divided into three broad categories in terms of the methodology used, i.e., embedding-based, path-based, and rule-based.

EMBEDDING-BASED Given the entity prediction task as an example, the embedding-based ranking techniques first learn embedding vectors by using already existing triples. These methods evaluate the scores of all potential entities by substituting the tail entity or head entity with each entity in the set of entities and subsequently rank the top k entities. TransE [49], ConVe [16], and DKRL [50] are among such embedding-based LP techniques.

PATH-BASED Although embedding-based approaches have shown remarkable results in certain benchmarks, they fail to model complex relation paths. In order to tackle this problem, relation path reasoning has been introduced to leverage the path information present in the graph structure. Random walk inference has been extensively studied, and the path-ranking algorithm (PRA) [51] is an example of a method that selects a relational path based on a combination of path constraints and conducts maximum-likelihood classification. Another path-based approach is to leverage reinforcement learning (RL)-based path finding methodologies. In order to enable multihop reasoning, Deep RL is utilized by treating the search for paths between entity pairs as a sequential decision-making process, (i.e., a Markov decision process (MDP)). The RL agent, which is based on policy, is trained to identify a step of relation and extend the reasoning paths by the interaction between the KG environment. Training of the RL agent is accomplished through the use of policy gradients.

RULE-BASED The approach based on rules involves learning logical formulas that explicitly represent statistical regularities and dependencies present in the KG [52]. These rules are then utilized to rank potential candidates for incomplete triples by determining the confidence of the rules being fired.

It should be noted that certain LP methods may fall into two or more of the aforementioned categories (i.e., embedding-based, path-based, and rule-based), thus forming hybrid methods. For example, as explained in [47], there are some methods that combine symbolic reasoning and embedding techniques, aiming to integrate rule-based reasoning, address the sparsity issue of KGs, enhance the accuracy of embeddings, enable effective rule integration, and induce explainable rules.

2.8 evaluation metrics

Different evaluation metrics are used for the different types of link prediction problems discussed above, i.e., entity prediction, triple classification, and entity type prediction. In this thesis, the entity prediction problem is addressed, and hence, the metrics that are used particularly for the entity prediction models (i.e., Mean Rank, Mean Reciprocal Rank (MRR), and Hits@k) are discussed as follows.

mean rank The mean rank (MR) represents the average rank of the true triples, where larger values indicate better performance. MR can be computed as:

$$MR = \frac{1}{|T|} \sum_{t \in T} \text{rank(t)}, \tag{11}$$

mean reciprocal rank The mean reciprocal rank (MRR) is calculated as the arithmetic mean over the reciprocals of ranks of true triples. A higher MRR value implies that the model is more proficient at predicting the relevant links. MRR is frequently employed as a criterion for early stopping since it provides a smooth evaluation that places a stronger weight on small ranks and is less affected by outlier individual ranks [53]. Formally, MRR is defined as:

$$MRR = \frac{1}{|T|} \sum_{t \in T} \frac{1}{\text{rank(t)}}, \tag{12}$$

where T is a set of triples and $rank(t)$ is the rank of the triple.

hits@k Hits@K denotes the fraction of true triples that appear in the first k triples among the top k sorted triples. Larger values indicate better performance. Hits@K is computed as follows:

$$Hits@K = \frac{\{t \in T \mid rank(t) \leqslant k\}}{|T|} \tag{13}$$

Part II

LITERATURE REVIEW

3 KGE MODELS WITH LITERALS IN TRANSDUCTIVE SETTING

Over the past decade, numerous techniques have emerged for generating KGEs that can be used to predict missing links in KGs. Most of these methods have primarily focused on the transductive setting of LP (i.e., a setting where all entities in the test and validation sets are required to be part of the training set as discussed in detail in Chapter 1). Some of these models make use of the different types of literals present in KGs. In this chapter, an extensive literature review of the those KGE models which use literals is provided. The findings of the review are also published as surveys [1, 14].

The rest of this chapter is organized as follows. Section 3.1 provides the rationale for conducting an extensive survey on KGE models with literals. In Section 3.2, the various KGE methods which leverage literals are discussed followed by a presentation of the various applications of KGEs on which the methods are trained or evaluated, in Section 3.3. A discussion of the existing datasets that have been used for the evaluation of KGC is provided in Section 3.4 followed by, the experiments conducted with these approaches on the task of LP, in Section 3.5. Finally, in Section 3.6, a concluding remark on the SoTA methods is presented.

3.1 INTRODUCTION

As mentioned above, there have been various approaches to KGEs, some of which involve the use of literals. A list of the most popular KGE techniques, including the SOTA approaches, is given in Table 3.1.1. The categories presented in this table are inspired by a previous survey work [8] for the models without literals (column 1). The categories are translation-based models, semantic matching models, models incorporating entity types, models incorporating relation paths, models using logical rules, models with temporal information, and models using graph structures. Since the topic of this thesis is on generating KGE models leveraging literals, the main focus of this literature review lies in analyzing the SoTA KGE models which make use of literals. Hence, in this thesis, the techniques which use literals are also grouped with respect to the same set of categories in Table 3.1.1. Moreover, the standard KGE techniques which are extended by the models with literals are listed in Table 3.1.2.

Few attempts have been made to conduct surveys on the techniques and applications of KGEs [8, 13, 102]. The survey [102] is conducted on factorization based, random walk based, and deep learning based network embedding approaches such as DeepWalk, Node2vec, and etc. [8, 13] discuss only RESCAL [67] and KREAR [103] as methods which use attributes of entities for KGEs, and focus mostly on the structure-based embedding methods,

Table 3.1.1: KGE models and their categories.

Categories	**Models without literals**	**Models with Literals**
Translational Distance Models	TransE [49] and its extensions: TransH [54] TransR [55], TransD [56], TranSparse [57], TransA [58] etc.	TransEA [59], DKRL [60], IKRL [61], Jointly(desp) [62], Jointly [63], SSP [64], KDCoE [65], EAKGAE [66]
Semantic Matching Models	RESCAL [67] and Its Extensions: DistMult [50], HolE [68], ComplEx [69], and etc. Semantic Matching with Neural Networks: SME [70], NTN [71], MLP [72], and etc.	LiteralE [73], MKBE [74], MTKGNN [75], KGlove with literals [76], Extended RESCAL [77], LiteralE with blocking [78]
Models using Entity Types	SSE [79], TKRL [80], Type constrained representation learning [81], Rules incorporated KG completion models [82], TRESCAL [83], Entity Hierarchy Embedding [84]	Extended RESCAL [77]
Models using Relation Paths	PTransE [85], Traversing KGs in Vector Space [86], RTRANSE [87], Compositional vector space [88], Reasoning using RNN [89], Context-dependent KGE [90]	KBLRN [91]
Models using Logical Rules	Rules incorporated KG completion models [82], Large-scale Knowledge Base Completion [92], KALE [93], Logical Background Knowledge for Relation Extraction [94], and etc.	
Models using Temporal Information	Time-Aware Link Prediction [95], co-evolution of event and KGs [96], Know-evolve [97]	
Models using Graph Structures	GAKE [98], Link Prediction in Multi-relational Graphs [99]	KBLRN [91]

i.e., methods using non-attributive triples, for example, translation based embedding models listed in Table 3.1.1. However, RESCAL is a matrix-factorization method for relational learning which encodes each object/data property as a slice of the tensor leading to an increase in the dimensionality of the tensor automatically. This method suffers from efficiency

Table 3.1.2: KGE models with literals and their corresponding base models.

Models with literals	**The standard models they extend**
Extended RESCAL [77]	RESCAL [67]
Jointly(desp) [62]	TransE [49]
DKRL [60]	TransE [49]
Jointly [63]	TransE [49]
SSP [64]	TransE [49]
KDCoE [65]	TransE [49]
KGlove with literals [76]	KGlove
LiteralE [73]	DistMult [50], ComplEx [69], ConvE [16]
TransEA [59]	TransE [49]
IKRL [61]	TransE [49]
MTKGRL [100]	TransE [49]
EAKGAE [66]	TransE [49]
MKBE [74]	DistMult [50], ConvE [16]
MADLINK[101]	DistMult[50]

issues if literals are utilized while generating KGEs. Similarly, KREAR only considers those data properties which have categorical values, i.e., fixed number of values and ignores those which take any random literals as values. One of the recent surveys [24] summarizes the methods proposed so far on refining KGs. However, this survey does not confine itself to embedding techniques and also does not consider most of the approaches which are making use of literals. Another very recent related study [104], discusses different aspects of KGE models such as model architectures, training strategies, and hyperparameter optimization but it takes into consideration only those models without literals.

None of the surveys mentioned above cover all the existing KGE models which make use of literals, such as the ones categorized as models incorporating information represented in literals in Table 3.1.1. Taking this into consideration, this chapter addresses the challenges in reference to the research question *C2-RQ1* from Section 1.2 of Chapter 1.

- **C_2-RQ_1**: - *How well do the SOTA KGE approaches which use literals perform for the task of LP?*

In order to answer this question, as part of this thesis, a detailed review of the SOTA KGE models which utilize literals is conducted and published in the surveys [1, 14]. The major difference between the two surveys is that the first one [1] is a short comprehensive overview whereas the second one [14] provides a more extensive theoretical analysis of these models along with an empirical evaluation on the task of link prediction. The findings of these surveys are presented in the rest of this chapter.

3.2 SOTA MODELS

This section investigates KGE models with literals divided into the following different categories based on the types of literals utilized: (i) Text, (ii) Numeric, (iii) Image, and (iv) Multi-modal. A KGE model which makes use of at least two types of literals providing complementary information is considered multi-modal. In the subsequent sections, the descriptions of the models for each of the previously described categories are provided, highlighting their similarities and differences, as well as a discussion of potential drawbacks.

3.2.1 *Models with Text Literals*

In this section, KGE models utilizing text literals are discussed, namely, Extended RESCAL [77], Jointly(desp) [62] , DKRL [60], Jointly [63], SSP [64] , KDCoE [65], KGloVe with literals [76], and MADLINK [101]. A detailed description followed by a summary presenting the comparison of these models is given along with their drawbacks. Moreover, in order to show the differences between the models based on complexity, the number of parameters of each model is presented in Table 3.2.1.

EXTENDED RESCAL aims to improve the original RESCAL approach by extending its algorithm to process literal values more efficiently and to deal with the drawback of sparsity that accompanies tensors. In the original RESCAL approach, relational data is modeled as a three-way tensor X of size $n \times n \times m$, where n is the number of entities and m is the number of relations. An entry $X_{ijk} = 1$ denotes the existence of the triple with i-th entity as a subject, k-th relation as a predicate, and j-th entity as an object. If X_{ijk} is set to 0, it indicates that the triple doesn't exist. A new approach for tensor factorization is proposed which is performed on X. For further details refer to [77]. If attributive triples have to be modeled in such a way, then the literals will be taken as entities even if they cannot occur as subjects in the triples. Including literals may lead to an increment in the runtime since a larger tensor has to be factorized.

In contrast to the original algorithm, the extended RESCAL algorithm handles the attributive triples in a separate matrix. The matrix factorization is performed jointly with the tensor factorization of the non-attributive triples. The attributive triples containing only text literals are encoded in an entity-attribute matrix D in such a way that the rows are entities and the columns are $< data\ type\ relation, value >$ pairs. Given a triple with a textual data type such as `rdfs:label` or `yago:hasPreferredMeaning`, one or more such pairs are created by tokenizing and stemming the text in the object literal. The matrix D is then factorized into $D \approx AV$ with A and V being the latent-component representations of entities and attributes respectively. Despite the advantage that this approach has for handling multi-valued attributes, it does not consider the sequence of words of the literal values. Note that Extended RESCAL represents RDF(S) data in such a way that there is no distinction drawn among A-Box and T-Box, i.e., both classes and instances are modeled equally as entities in a tensor. The T-Box is rather taken as soft constraints instead of letting them impose hard constraints on the model.

JOINTLY(DISP) is an approach which jointly learns embeddings of KGs and a text corpus of entity descriptions, i.e, it uses an alignment model to make sure the entities, relations, and words are represented in the same vector space. This approach consists of three components, namely, knowledge model, text model, and alignment model. The knowledge model is used to capture the semantics of the structured information from the KG. Given a triple $< h, r, t >$, the model defines the plausibility of the triple, same as in [105]:

$$\Pr(h|r,t) = \frac{\exp\{z(h,r,t)\}}{\sum_{\tilde{h}\in I} \exp\{z(\tilde{h},r,t)\}}, \tag{14}$$

where $z(h,r,t) = b - 0.5 \cdot \|h + r - t\|_2^2$, $b = 7$. Analogously, $\Pr(r|h,t)$ and $\Pr(t|h,r)$ are defined.

Then, the loss function of the knowledge model is defined as follows:

$$\begin{aligned} L_K = \sum_{(h,r,t)} [&\log \Pr(h|r,t) + \log \Pr(t|h,r) \\ &+ \log \Pr(r|h,t)]. \end{aligned} \tag{15}$$

The text model adopts the same assumption made in [105] that is if two words occur in the same context then there is a relation between them. Based on this assumption, the text model defines the probability of a pair of words w and v co-occurring in a text window as follows:

$$\Pr(w|v) = \frac{\exp\{z(w,v)\}}{\sum_{\tilde{w}\in V} \exp\{z(\tilde{w},v)\}}, \tag{16}$$

where $z(w,v) = b - 0.5 \cdot \|\mathbf{w} - \mathbf{v}\|_2^2$. Then, the loss function of the text model is given as:

$$L_T = \sum_{(w,v)} \log \Pr(w|v). \tag{17}$$

The role of the third component, the alignment model, is to put the embeddings of the entities, relations, and words into the same vector space. This submodel works by utilizing entity descriptions to align these embeddings. For every word w in the description of entity e, the conditional probability of predicting w given e is defined as :

$$\Pr(w|e) = \frac{\exp\{z(e,w)\}}{\sum_{\tilde{w}\in V}\exp\{z(e,\tilde{w})\}}, \tag{18}$$

where $z(e,w) = b - 0.5 \cdot \|\mathbf{e} - \mathbf{w}\|_2^2$. The entity vector $\mathbf{e}$ in Eq 18 is the same as the entity vector appearing in Eq 14, i.e., an entity has a single unified representation which captures the semantics from both the structured KG and the entity descriptions. $\Pr(e|w)$ is defined analogously. Based on the definition given in Eq 18, the loss function of the alignment model is defined as:

$$L_A = \sum_{e\in\mathcal{E}}\sum_{w\in D_e}[\log\Pr(w|e) + \log\Pr(e|w)], \tag{19}$$

where $\mathcal{E}$ and D_e denote the set of entities and the description of the entity e respectively.

By adopting the joint embedding framework in [105], the main loss of Jointly(desp) is defined as follows:

$$L(\{e_i\},\{r_j\},\{w_l\}) = L_K + L_T + L_A. \tag{20}$$

DKRL extends TransE [49] by utilizing the descriptions of entities. For each entity e, two kinds of vector representations are learned, i.e., structure-based e_s and description-based e_d. These two kinds of entity representations are learned simultaneously into the same vector space but not forced to be unified so that novel entities with only descriptions can be represented. In order to achieve this, given a certain triple $< h, r, t >$ the energy function of the DKRL model is defined as:

$$\begin{aligned} E = &\|h_s + r - t_s\| + \|h_d + r - t_d\| \\ &+\|h_s + r - t_d\| + \|h_d + r - t_s\|, \end{aligned} \tag{21}$$

where h_s and t_s are the structure-based representations, and h_d and t_d are the description-based representations of their corresponding entities.

In order to learn structure-based representations, the TransE approach is directly applied which considers the relation in a triple as the translation from the head entity to the tail entity. On the other hand, Continuous Bag of Words (CBOW) and a deep Convolutional Neural Network (CNN) model have been used to generate the description-based representations of the head and tail entities. In the case of CBOW, short text is generated from the description based on keywords and their corresponding word embeddings are summed up to generate the entity embedding. In the CNN model, after preprocessing the description, pretrained word vectors from Wikipedia are provided as input. This CNN model has five layers and after every convolutional layer pooling is applied to decrease the parameter space of CNN and filter noises. Max-pooling is applied for the first pooling and mean pooling for the last one. The activation function used is either tanh or ReLU. The CNN model works better than CBOW because it preserves the sequence of words.

In order to train DKRL, the following margin-based score function is considered as an objective function and minimized using a standard backpropagation using stochastic gradient descent (SGD)

$$L = \sum_{(h,r,t)\in T} \sum_{(h',r',t')\in T'} max(\gamma + d(h+r,t) \\ -d(h'+r',t'),0), \tag{22}$$

where $\gamma > 0$ is a margin hyperparameter, d is a dissimilarity function and T' is the set of corrupted triples. The representation of the entities can be either structure-based or description-based.

JOINTLY Jointly [63] learns KGEs by leveraging entity descriptions. Specifically, it learns a joint embedding of an entity by combining its structure-based and description-based representations with a gating mechanism. The gate is used to find a balance between the structure-based and the description-based representations. For a certain entity, a representation can be encoded from its descriptions by converting the description into a fixed-length vector. In Jointly, different text encoders have been used such as bag-of-words, LSTM, and Attentive LSTM.

For an entity e, its joint representation $\mathbf{e}$ is a linear interpolation between its structure-based representation $(\mathbf{e_s})$ and description-based representation $(\mathbf{e_d})$, which is defined as:

$$\mathbf{e} = g_e \odot \mathbf{e_s} + (1 - g_e) \odot \mathbf{e_d}, \tag{23}$$

where $\odot$ is an element-wise multiplication and g_e is a gate to balance the two information sources (structure and text) which is computed as $g_e = \alpha(\tilde{g}_e)$ with $g_e = \tilde{g}_e \in \mathcal{R}^d$ being real-value vector stored in a lookup table.

The entity descriptions are encoded using either bag-of-words, LSTM, or Attentive LSTM (ALSTM) encoders in order to generate text-based representations for the corresponding entities. On the other hand, to better model the structure-based embedidngs, entities and relations can be pre-trained with any existing KGE models, such as TransE.

Jointly's score function is inspired by TransE and defined as follows:

$$f(h,r,t;d_h,d_t) = \|(\mathbf{g}_h \odot \mathbf{h}_s + (1-\mathbf{g}_h) \\ \odot \mathbf{h}_d) + r - (g_t \odot \mathbf{h}_t + (1-\mathbf{g}_t) \odot \mathbf{t}_d)\|_2^2. \tag{24}$$

where $\mathbf{h}_s$, $\mathbf{h}_d$, and $\mathbf{g}_h$ are the head entity's structure-based embedding, description-based embedding, and gate respectively whereas $\mathbf{t}_s$, $\mathbf{t}_d$, and $\mathbf{g}_t$ are the tail entity's structure-based embedding, description-based embedding, and gate respectively.

SSP SSP (Semantic Space Projection) [64] is a joint embedding model which learns from both structured/symbolic triples and textual descriptions. Differently from DKRL and Jointly(Desp), where first-order constraints which are weak in capturing the correlation of textual descriptions and symbolic triples are applied, SSP follows the principle that triple embedding is considered always as the main procedure and textual descriptions must interact with triples in order to learn better representation. Therefore, triple embedding is projected onto a semantic subspace such as a hyperplane to allow strong correlation by adopting quadratic constraint.

SSP applies the following scoring function:

$$f_r(h,t) = -\lambda\|\mathbf{e} - \mathbf{s}^T\mathbf{e}\mathbf{s}\|_2^2 + \|\mathbf{e}\|_2^2, \tag{25}$$

where

$$\mathbf{e} \doteq \mathbf{h} + \mathbf{r} - \mathbf{t}, \tag{26}$$

and

$$\mathbf{s} \doteq \frac{\mathbf{s_h} + \mathbf{s_t}}{\|\mathbf{s_h} + \mathbf{s_t}\|}. \tag{27}$$

Note that λ is a suitable hyper-parameter, $\mathbf{h}$ and $\mathbf{t}$ are the structure (symbolic triples) based embedding of the head and tail entities respectively, $\mathbf{s_h}$ and $\mathbf{s_t}$ are the semantic vectors generated from the textual descriptions of the head and tail entities respectively. SSE adopts the Non-negative Matrix Factorization (NMF) topic model to generate description-based semantic vectors for entities ($\mathbf{s_h}$ and $\mathbf{s_t}$), i.e., by treating each entity description as a document and taking the topic distribution of the document as the representation of the corresponding entity.

SSP provides two different settings for training which are referred to as **Std** and **Joint**. In Std, a pre-trained topic model with NMF is used to obtain description-based semantic vectors. These description-based vectors are fixed during training but the other parameters are optimized. On the other hand, in the joint setting, the topic model is also learned simultaneously with the KGEs instead of using fixed pre-trained vectors.

KDCOE KDCoE focuses on the creation of an alignment between entities of multilingual KGs by creating new Inter-Lingual Links (ILLs) based on an embedding approach that exploits entity descriptions. The model uses a weakly aligned multilingual KG for semi-supervised cross-lingual learning. It performs co-training of a multilingual KGE Model (KGEM) and a multilingual entity Description Embedding Model (DEM) iteratively in order for each model to propose a new ILL alternately. KGEM is composed of two components, i.e., a knowledge model and an alignment model, to learn embeddings based on structured information from the KGs (the non-attributive triples). Given a set of languages $\mathcal{L}$, a separate k_1-dimensional embedding space $\mathbb{R}_L^{k_1}$ is used for each language $L \in \mathcal{L}$ to represent the corresponding relations R_L and entities E_L. In order to learn the embeddings for R_L and E_L, the knowledge model adopts TransE and thus uses hinge loss as its objective function. On the other hand, a linear-transformation-based technique that has the best performance in the case of cross-lingual inferences is adopted for the alignment model. This technique employs the following objective function:

$$S_A = \sum_{(e,e')\in I(L_i,L_j)} \|M_{ij}\mathbf{e} - \mathbf{e}'\|_2, \tag{28}$$

where $I(L_i, L_j)$ is ILLs between the languages L_i and L_j, and M_{ij} is a $k_1 \times k_1$ matrix used as a linear transformation on entity vectors from L_i to L_j.

Let S_K be the hinge loss function used by the knowledge model, the KGEM model then minimizes $S_{KG} = S_K + \alpha S_A$, where α is a positive hyperparameter. In the case of DEM model, an attentive gated recurrent unit encoder (AGRU) is used to encode the multilingual entity descriptions. DEM applies multilingual word embeddings in order to capture the semantic information of multilingual entity descriptions from the word level. The two models, i.e., KGEM and DEM, are iteratively co-trained in order for each model to propose a new ILL alternately.

KGLOVE WITH LITERALS KGloVe with literals is an experimental attempt to incorporate entity descriptions in KGloVe KGE approach. The experiment is conducted on DBpedia considering the abstracts and comments of entities as their descriptions. The main goal is to extract named entities from the textual description and for every entity in the text, to replace those words representing it with the entity itself and then take its neighboring words and entities as its context. The approach works by creating two co-occurrence matrices independently and then by merging them at the end so that a joint embedding can be performed. The first matrix is generated using the same technique as in KGloVe [106], i.e., by performing Personalized PageRank (PPR) on the (weighted) graph followed by the same optimization used in the GloVe [37] approach.

In order to create the second matrix, the Named Entity Recognition (NER) task is performed on the entity description text using the list of entities and predicates of the KG as input. The NER step employs a simple exact string matching technique which leads to numerous drawbacks such as missing entities due to having different keywords with the same semantics. All the English words that do not match any entity labels are added to the entity-predicate list. Then GloVe co-occurrence for text is applied to the modified text (i.e., DBpedia abstract and comments) using the entity-predicate and word list as input. Finally, the two co-occurrence matrices are summed up together to create a single unified matrix. The proposed approach has been evaluated on classification and regression tasks and the result indicates that for most of the classifiers used, except SVM, the approach does not bring significant improvement to KGloVe. However, the approach can be improved using parameter tuning with extensive experiments.

MADLINK MADLINK is a multi-hop attentive KGE model that aims to enhance the task of LP by capturing the semantics of entities and relations through the combination of the random walks (paths) and textual entity descriptions. The random walk captures the contextual information of the entities. The selection of the paths also takes into account the significance of an entity with respect to a relation. For a certain relation, its contextual information is captured by considering all the triples containing that relation. To obtain a cumulative representation of the paths extracted for each entity from the KG, MADLINK uses an adapted seq2seq [107] encoder-decoder architecture with an attention layer. The latent representations of entity descriptions provided in natural language text are extracted using SBERT [108]. The DistMult scoring function is used to compute the score of a triple for head or tail prediction.

SUMMARY The basic differences between these models lie in the methods used to exploit the information given in the text literals and combine them with structure-based representation. One major advantage of KDCoE over text literal based embedding models is that it considers descriptions present in multilingual KGs. Jointly(Desp) aligns KGEs and word embeddings on word level, which may lead to losing some semantic information on phrase or sentence level. Jointly applies a gating mechanism that allows to automatically find a balance between the structural and textual information. It also uses an LSTM encoder which enables the model to select the most related information for an entity from its text description according to different relations. Unlike DKRL and Jointly(Desp), SSP focuses on characterizing the stronger correlations between entity descriptions and structured triples by projecting triple embedding onto a semantic subspace such as a hyper-plane, as discussed above. On the other hand, MADLINK exploits the contextual information from the KG through random walk and also captures semantics from entity descriptions by applying a LM. The common drawback among the presented approaches with text literals is they focus mostly on descriptions, which are long natural language text, and thus, other types of text literals, such as names, labels, titles, etc. are not widely considered. Moreover, another way to compare these approaches is by looking at their model complexity. Table 3.2.1 presents the complexity of these models in terms of their number of parameters.

Table 3.2.1: The complexity of the models with text literals in terms of the number of parameters. Θ is the number of parameters in the base model, H is the entity embedding size, N_d is the number of data relations, L is the number of attribute-value pairs, N_r is the number of relations, N_w is the number of words, H' is the word embedding size, $N_0^{(1)}$ is the dimension of input vectors at the first layer, $N_1^{(1)}$ is the dimension of output vectors at layer 1, K is window size, $N_0^{(2)}$ is the dimension of input vectors at the second layer, $N_1^{(2)}$ is the dimension of output vectors at second layer, N_{e_1} and N_{e_2} denote the number of entities in two different languages of a multilingual KG, N_{r_1} and N_{r_2} denote the number of relations in two different languages of a multilingual KG, N is the total number of entities and relations, and M is the total number of entities, relations and words. θ_1 and θ_2 represent the cumulative size of the parameters from the encoder and decoder GRUs, respectively. H'' is the path-based entity embedding size

Model	#Parameter
Extended Rescal	$\Theta + HL$
Jointly(Desp)	$\Theta + N_w H'$
DKRL	$\Theta + N_w H' + N_0^1 K N_1^{(1)} + N_0^{(2)} K N_1^{(2)}$
Jointly(ALSTM)	$\Theta + (2H+4)H$
SSP	$\Theta + (N_e + N_w)H$
KDCoE	$(N_{e_1} + N_{e_2} + N_{r_1} + N_{r_2} + H)H$
	$+(5H' + 3N_w)H'$
KGlove with literals	$(N+1)N + N_w + M$
MADLINK	$2H + H'' + \theta_1 + \theta_2$

3.2.2 *Models with Numeric Literals*

In this section, the analysis of the KGE models which use numeric literals, namely, MT-KGNN [75], KBLRN [91], LiteralE [73], and TransEA [59] are presented followed by a summary. Moreover, in order to show the differences between the models based on complexity, the number of parameters of each model is presented in Table 3.2.2.

MT-KGNN MT-KGNN is an approach for both relational learning and non-discrete attribute prediction on knowledge graphs in order to learn embeddings for entities, object properties, and data properties. It is composed of two networks, namely, the Relational Network (RelNet) and the Attribute Network (AttrNet). RelNet is a binary (pointwise) classifier for triple prediction whereas AttrNet is a regression task for attribute value prediction. Given n, m, and l as entity, relation, and literal embedding dimensions respectively, the model passes as an input $[e_i, r_k, e_j, t]$ to RelNet and $[a_i, v_i, a_j, v_j]$ to AttrNet, where e_i , $e_j \in R^n$, $r_k \in R^m$, t is the classification target which is 0 or 1, $a_i, a_j \in R^l$, and v_i and v_j are normalized continuous values in the interval $[0, 1]$. Note that the inputs to AttrNet, i.e., $[a_i, v_i, a_j, v_j]$, are taken from attributive triples with non-discrete literal values. An embedding lookup layer is used to retrieve the corresponding vector representations given these inputs as one-hot encoded indices.

In RelNet, a concatenated triple is passed through a nonlinear transform and then a sigmoid function is applied to get a linear transform:

$$g_{rel}(e_i, r_k, e_j) = \sigma(\overrightarrow{w}^T f(W_d^T[\overrightarrow{e_i}; \overrightarrow{e_j}; \overrightarrow{r_k}]) \\ + b_{rel}), \tag{29}$$

where $w \in R^{h\times 1}$ and $W_d \in R^{3n\times h}$ are parameters of the network. σ, f, and b_{rel} are the sigmoid function, the hyperbolic tangent function tanh, and a scalar bias respectively. RelNet is trained by minimizing the following cross entropy loss function:

$$L_{rel} = -\sum_{i=1}^{N} t_i \log g_{rel}(\xi_i) \\ +(1-t_i)\log(1-g_{rel}(\xi_i))), \tag{30}$$

where ξ_i denotes triple i in batch of size N and t_i takes the value 0 or 1. In case of AttrNet, two regression tasks are performed, one for the head data properties and another for those of the tail. The following scoring functions are defined for these two tasks:

$$g_h(a_i) = \sigma(\overrightarrow{u}^T f(B^T[\overrightarrow{a_i}; \overrightarrow{e_i}]) + b_{z_1}), \tag{31}$$

and

$$g_t(a_j) = \sigma(\overrightarrow{y}^T f(C^T[\overrightarrow{a_j}; \overrightarrow{e_j}]) + b_{z_2}), \tag{32}$$

where $u, y \in R^{h_a\times 1}$ and $B, C \in R^{2n\times h_a}$ are parameters of AttrNet. h_a is the size of the hidden layer and b_{z_1}, b_{z_2} are scalar biases. Each AttrNet is trained by optimizing Mean Squared Error (MSE) loss function:

$$MSE(s, s^*) = \frac{1}{N}\sum_{i=1}^{N}(s_i - s_i^*)^2. \tag{33}$$

where s and s^* are predicted labels (scores computed by the model) and ground truth labels respectively.
The overall loss of the AttrNet is computed by adding the MSE of the head AttrNet and that of the tail AttrNet as follows:

$$L_{attr} = MSE(g_h(a_i), (a_i)^*) \\ +MSE(g_t(a_j), (a_j)^*), \tag{34}$$

where $(a_i)^*, (a_j)^*$ are the ground truth labels. Finally, the two networks are trained in a multi-task fashion using a shared embedding space.

KBLRN KBLRN works by combining relational (R), latent (L), and numerical (N) features together. The model is designed mainly for the purpose of KG completion. It uses a probabilistic PoE (Product of Experts) method to combine these feature types and train them jointly end to end. Each relational feature is formulated as a logical formula, by adopting the rule mining approach AMIE+ [109], to be evaluated in the KB to compute the feature's value. The latent features are the ones that are usually generated using an embedding approach such as DistMult. Numerical features are used with the assumption that, for some relation types, the differences between the head and tail can be seen as characteristics of the relation itself. Given a triple $d = (h, r, t)$, for each (relation type r, and feature type $F \in \{L, R, N\}$) pair, individual experts are defined based on linear models and DistMult embedding method as follows:

$$f_{(r,L)}(d \mid \theta_{(r,L)}) = \exp((e_h * e_t) \cdot w^r), \tag{35}$$

$$f_{(r,R)}(d \mid \theta_{(r,R)}) = \exp(r_{(h,t)} \cdot w^r_{rel}), \tag{36}$$

$$f_{(r,N)}(d \mid \theta_{(r,N)}) = \exp(\phi(n_{(h,t)}) \cdot w^r_{num}), \tag{37}$$

and

$$f_{(r',F)}(d \mid \theta_{(r',F)}) = 1 \text{ for all } r' \neq r \tag{38}$$

where $w_r, w^r_{rel}, w^r_{num}$ are the parameter vectors for the latent, relational, and numerical features corresponding to the relation r. Also, * is the element-wise product, · is the dot product, and ϕ is the radial basis function (RBF) applied element-wise to the differences of values $n_{(h,t)}$ computed as follows:

$$\begin{aligned}\phi(n_{(h,t)}) = [&\exp(\frac{-\|n^{(1)}_{(h,t)} - c_1\|^2_2}{\sigma^2_1}) \ldots \\ &\exp(\frac{-\|n^{(d_n)}_{(h,t)} - c_{d_n}\|^2_2}{\sigma^2_{d_n}})].\end{aligned} \tag{39}$$

Here, d_n corresponds to the relevant numerical features. A PoE's probability distribution for a triple $d = (h, r, t)$ is defined as follows:

$$p(d \mid \theta_1 \ldots \theta_n) = \frac{\prod_F f_{(r,F)}(d \mid \theta_{(r,F)})}{\sum_c \prod_F f_{(r,F)}(c \mid \theta_{(r,F)})}, \tag{40}$$

where c denotes all possible triples. The parameters of the entity embedding model are shared by all the experts in order to create dependencies between them. In this approach, the PoE are trained with negative sampling and a cross-entropy loss to give high probability to observed triples.

LITERALE LiteralE incorporates literals into existing latent feature models designed for LP. In this approach, without loss of generality, the focus lies on incorporating numerical literals into three state-of-the-art embedding methods: DistMult, ComplEx, and ConvE. Given a base model like Distmult, LiteralE modifies the scoring function f used in Distmult by replacing the vector representations of the entities e_i in f with literal enriched representations e_i^{lit}. In order to generate e_i^{lit}, LiteralE uses a learnable transformation function g which takes e_i and its corresponding literal vectors l_i as inputs and maps them to a new vector. The function g is defined, as shown below, based on the concept of GRU in order to make it flexible, learnable, and capable to decide, if it is beneficial to incorporate the literal information or not:

$$g : \mathbb{R}^H \times \mathbb{R}^{N_d} \rightarrow \mathbb{R}^H, \tag{41}$$

and

$$\mathbf{e}, \mathbf{l} \mapsto \mathbf{z} \odot \mathbf{h} + (1 - \mathbf{z}) \odot \mathbf{e}, \tag{42}$$

where

$$\mathbf{z} : \sigma(\mathbf{W}_{ze}^T \mathbf{e} + \mathbf{W}_{zl}^T \mathbf{l} + \mathbf{b}), \tag{43}$$

and

$$\mathbf{h} = h(\mathbf{W}_h^T [\mathbf{e}, \mathbf{l}]). \tag{44}$$

Note that $\mathbf{W}_{ze} \in \mathbb{R}^{H \times H}, \mathbf{W}_{zl} \in \mathbb{R}^{N_d \times H}, \mathbf{b} \in \mathbb{R}^H$, and $\mathbf{W}_h \in \mathbb{R}^{H + N_d \times H}$ are the parameters of g, σ is the sigmoid function, $\odot$ denotes the element-wise multiplication, and h is a component-wise nonlinearity. The scoring function $f(\mathbf{e}_i, \mathbf{e}_j, \mathbf{r}_k)$ has been replaced with $f(g(\mathbf{e}_i, \mathbf{l}_i), g(\mathbf{e}_j, \mathbf{l}_j), \mathbf{r}_k)$ and trained following the same procedure as in the base model.

TRANSEA TransEA has two component models; a directly adopted translation-based structure embedding model (i.e., TransE) and a newly proposed attribute embedding model. In the former, the scoring function of a given triple $< h, r, t >$, is defined as follows:

$$f_r(h, t) = -||h + r - t||_{1/2}, \tag{45}$$

where $||x||_{1/2}$ denotes either the L1 or L2 norm. The loss function of the structure embedding, for all the relational triples in the KG, is defined as:

$$\begin{aligned} L_R = \sum_{<h,r,t> \in T} \sum_{<h',r,t'> \in T'} & max(\gamma + f_r(h, t) \\ & - f_r(h', t'), 0), \end{aligned} \tag{46}$$

where T' denotes the set of negative triples constructed by corrupting either the head or the tail entity and $\gamma > 0$ is a margin hyperparameter.

For the attribute embedding, it uses all attributive triples containing numeric values as input and applies a linear regression model to learn embeddings of entities and attributes. Given an attributive triple $< e, a, v >$, the scoring function is defined as:

$$f_a(e, v) = -\|\mathbf{a}^T \cdot \mathbf{e} + b_a - v\|_{1/2}, \tag{47}$$

where $\mathbf{a}$ and $\mathbf{e}$ are vectors of attribute a and entity e, b_a is a bias for attribute a. On the other hand, given all the attributive triples with numeric values S, the loss function for the attributive embedding is computed as:

$$L_A = \sum_{<e,a,v>\in S} f_a(e, v), \tag{48}$$

The main loss function for TransEA (i.e., $L = (1 - \alpha) \cdot L_R + \alpha \cdot L_A$) is defined by taking the sum of the respective loss functions of the component models with a hyperparameter to assign a weight for each of the models. Finally, the two models are jointly optimized in the training process by sharing the embeddings of entities.

SUMMARY Despite their support for numerical literals, all the embedding methods discussed fail to interpret the semantics behind units/data typed literals. For instance, given the following two triples taken from DBpedia,

```
<http://dbpedia.org/resource/Anton_Baraniak, dbp:weight, "110.0"^^<http://dbpedia.org/datatype/
    kilogram>,
<http://dbpedia.org/resource/Katelin_Snyder, dbp:weight, "110.0"^^<http://dbpedia.org/datatype/
    pound>
```

the literal value "110.0" from the first triple and the literal value "110.0" from the second triple could be considered exactly the same if the semantics of the types kilogram and pound are ignored. Moreover, most of the models do not employ a proper mechanism to handle multi-valued attributes.

Regarding model complexity, the number of parameters used in each model is presented in Table 3.2.2 to show the complexity in terms of the parameters. It is noted that the complexity of the models depends on the size of the dataset and TransEA has lower complexity as compared to the other models.

3.2.3 *Models with Images*

In this section, KGE models utilizing images of entities, namely, IKRL [61] and MTKGRL [100] are discussed. First, a detailed analysis of the models is presented followed by a summary. Moreover, in order to show the differences between the models based on complexity, the number of parameters of each model is presented in Table 3.2.3.

Table 3.2.2: Complexity of the models with numerical literals in terms of the number of parameters. Θ is the number of parameters in the base model, H is the entity embedding size, N_d is the number of data relations, Λ is the size of the hidden layer in the Attrnet networks of MTKGNN, N_r is the number of relations, and M is attribute embedding size.

Model	#Parameters
LiteralE with g	$\Theta + 2H^2 + 2N_dH + H$
LiteralE with g_{lin}	$\Theta + (N_dH + H)H$
MTKGNN	$\Theta + N_dH + 2(2H\Lambda + \Lambda)$
KBLN	$\Theta + N_rN_d$
TransEA	$\Theta + N_dM$

IKRL IKRL [61] learns embeddings for KGs by jointly training a structure-based representation with an image-based representation. The structure-based representation of an entity is learned by adapting a conventional embedding model like TransE. For the image-based representation, given the fact that an entity may have multiple image instances, an image encoder is applied to generate an embedding for each instance of a multi-valued image relation. The image encoder consists of a neural representation module and a projection module to extract discriminative features from images and to project these representations from image space to entity space respectively.

For the i-th image, its image-based representation p_i in entity space is computed as:

$$\mathbf{p}_i = \mathbf{M} \cdot f(img_i), \tag{49}$$

where $M \in \mathbb{R}^{d_i \times d_s}$ is the projection matrix with d_i and d_s representing the dimension of image features and the dimension of entities respectively. $f(img_i)$ is the i-th image feature representation in image space.

Attention-based multi-instance learning is used to integrate the representations learned for each image instance by automatically calculating the attention that should be given to each instance. The attention for the i-th image representation $p_i^{(k)}$ of the k-th entity is given as:

$$att(\mathbf{p}_i^{(k)}, \mathbf{e}_S^{(k)}) = \frac{\exp(\mathbf{p}_i^{(k)} \cdot \mathbf{e}_S^{(k)})}{\sum_{j=1}^{n} \exp(\mathbf{p}_j^{(k)} \cdot \mathbf{e}_S^{(k)})}, \tag{50}$$

where $\mathbf{e}_S^{(k)}$ denotes the structure-based representation of the k-th entity. The higher the attention the more similar the image-based representation is to its corresponding structure-based representation which indicates that it should be given more importance when aggregating the image-based representations. The aggregated image-based representation for the k-th entity is defined as follows:

$$\mathbf{e}_I^{(k)} = \sum_{i=1}^{n} \frac{att(\mathbf{p}_i^{(k)}, \mathbf{e}_S^{(k)}) \cdot \mathbf{p}_i^{(k)}}{\sum_{j=1}^{n} att(\mathbf{p}_j^{(k)}, \mathbf{e}_S^{(k)})}. \tag{51}$$

Given a triple, the overall energy function is defined by combining four energy functions (i.e., $E(h,r,t) = E_{SS} + E_{II} + E_{SI} + E_{IS}$. These energy functions are based on two kinds of entity representations (i.e, structure-based and image-based representations). The first energy function (i.e., $E_{SS} = \|h_S + r - t_S\|$) is the same as TransE and the second function (i.e., $E_{II} = \|h_I + r - t_I\|$) uses their corresponding image-based representations for both head and tail entities. The third function (i.e., $E_{SI} = \|h_S + r - t_I\|$) is based on the structure-based representation of the head entity and the image-based representation of the tail entity whereas the fourth function (i.e., $E_{IS} = \|h_I + r - t_S\|$) is the exact opposite. These third and fourth functions ensure that both structure-based representation and image-based representations are learned into the same vector space.

Given the energy function $E(h,r,t)$, a margin-based scoring function is defined as follows:

$$\begin{aligned} L = \sum_{(h,r,t)\in T} \sum_{(h',r',t')\in T'} & max(\gamma + E(h,r,t) \\ & -E(h',r',t'), 0), \end{aligned} \tag{52}$$

where γ is a margin hyperparameter and T' is the negative sample set of T generated by replacing the head entity, tail entity or the relation for each triple in T. Note that triples that are already in T are removed from T'.

MTKGRL MTKGRL [100] is a KGE approach that combines structural (symbolic), visual, and linguistic KG representations. The structural representations are created by adopting TransE embedding technique whereas visual embeddings are obtained from the feature layers of deep networks for image classification on the images that are associated with entities. For linguistic representations, pre-trained word embedding technique, specifically the skipgram model, is used. However, the information source for the linguistic representation is not literals from the KG but an external source, i.e., the word embedding model, trained on Google 100B token news dataset. Due to this fact, the model MTKGRL is not considered as multi-modal KGE model in the context of this survey and thus, it is not categorized under 'Models with Multi-modal Literals' (Sec 3.2.4).

MTKGRL defines an energy function for each kind of representation and also their combinations, i.e., structural energy, multi-modal energies, and structural-multi-modal energies. Structural energy is adopted from TransE, which is defined as $E_S = \|\boldsymbol{h_s} + \boldsymbol{r_s} - \boldsymbol{t_s}\|$. The mult-imodal representations for the head and tail entities are computed as $\boldsymbol{h_m} = \boldsymbol{h_w} \oplus \boldsymbol{h_i}$

and $\boldsymbol{t_m} = \boldsymbol{t_w} \oplus \boldsymbol{t_i}$ respectively, where the operator $\oplus$ can be a concatenation operator or a mapping function.

The multi-modal energy function under the translational assumption is given as:

$$E_{M1} = \|\mathbf{h_m} + \mathbf{r_s} - \mathbf{t_m}\|. \tag{53}$$

E_{M1} can be extended by considering the structural embeddings in addition to the multi-modal embeddings as follows:

$$E_{M2} = \|(\mathbf{h_m} + \mathbf{h_s})\mathbf{r_s} - (\mathbf{t_m} + \mathbf{t_s})\|. \tag{54}$$

On the other hand, in order to allow the structural and multi-modal embeddings to be learned in the same vector space, the following structural-multi-modal energies are defined as shown below:

$$E_{SM} = \|\mathbf{h_s} + \mathbf{r_s} - \mathbf{t_m}\| \tag{55a}$$

$$E_{MS} = \|\mathbf{h_m} + \mathbf{r_s} - \mathbf{t_s}\| \tag{55b}$$

The overall energy function, shown in Equation 56, is defined by combining the aforementioned energy functions, i.e., E_S, E_{M1}, E_{M2}, E_{SM}, E_{MS}.

$$\begin{aligned} E(h, r, t) = E_S + E_{M1} + E_{M2} + E_{SM} \\ + E_{MS} \end{aligned} \tag{56}$$

Finally, a margin-based ranking loss function is minimized in order to train the model.

SUMMARY IKRL makes use of the images of entities for KG representation learning by combining structure-based representation with image-based representation. However, given a triple $< h, r, t >$, in order to achieve very good representations for the entities h and t, both entities are required to have images associated with them. The other issue with this model is that an image is considered as an attribute of only those entities it is associated with. For example, if there is an image of two entities e_1 and e_2 but the image is associated with only e_1, then it will be taken as one image instance of e_1 but not of e_2. However, it would be more beneficial to explicitly associate images with all the entities they represent before using them for learning KGE. Some of the main points that make MTKGRL differ from IKRL are: i) in addition to images, MTKGRL uses linguistic embeddings for entities, ii) MTKGRL introduces an additional energy function that considers both linguistic and visual representations of entities as discussed above. These differences allow MTKGRL to learn better representation for KGs as compared to IKRL.

Table 3.2.3: The complexity of the models with images in terms of the number of parameters. Θ is the number of parameters in the base model, H is the entity embedding size, H_i represents the dimension of image features, $\theta_{AlexNet}$ is the number of parameters in AlexNet [110], N_e represents the number of entities, and N_i is the number of images.

Model	#Parameter
IKRL	$\Theta + H_i H + \Theta_{AlexNet}$
MTKGRL	$\Theta + N_e H + N_i H_i$

3.2.4 *Models with Multi-modal Literals*

This section presents an analysis of the embedding models making use of at least two types of literals providing complementary information. First, the category with numeric and text literals is discussed followed by the category with numeric, text, and image. Moreover, in order to show the differences between the models based on complexity, the number of parameters of each model is presented in Table 3.2.4.

3.2.4.1 *Models with Numeric and Text Literals*

LITERALE WITH BLOCKING LiteralE with blocking [78] proposes to improve the effectiveness of the data linking task by combining LiteralE with a CER blocking [111] strategy. Unlike LiteralE, given an attributive triple $< h, d, v >$, in addition to the object literal value v it also takes literals from URI infixes of the head entity h and the data relation d. The CER blocking is based on a two-pass indexing scheme. In the first pass, Levenshtein distance metric is used to process literal objects and URI infixes whereas in the second pass semantic similarity computation with WordNet [29] is applied to process object/data relations. All the extracted literals are tokenized into word lists so as to create inverted indices. The same training procedure as in LiteralE is used to train this model. For every given triple $< h, r, t >$, the scoring function f from LiteralE is adopted to compute scores for all the triples $< h, r, t' >$ in the knowledge graph. A sigmoid function, $p = \sigma(f(.))$, is used to produce probabilities. Then, the model is trained by minimizing the binary cross-entropy loss of the produced probability function vector with respect to the vector of truth values for the triples.

EAKGAE EAKGAE [66] is an approach designed for entity alignment between KGs by learning a unified embedding space for the KGs. The entity alignment task has three main modules: Predicate alignment, Embedding learning, and Entity alignment. The predicate alignment module merges two KGs together by renaming similar predicates so as to create unified vector space for the relationship embeddings. The embedding learning module jointly learns entity embeddings of two KGs using structure embedding (by adapting TransE) and attribute character embedding. The adapted TransE is customized in a way that more focus can be given to triples with aligned predicates. This is obtained by adding a weight α to control the embedding learning over the triples. Thus, the following objective function J_{SE} is defined for the structure-based embedding:

$$J_{SE} = \sum_{t_r \in T_r} \sum_{t_r' \in T_r'} max(0, \gamma + \alpha(f(t_r) - f(t_r'))), \tag{57}$$

and

$$\alpha = \frac{count(r)}{|T|}, \tag{58}$$

where T_r and T_r' are the sets of valid triples and corrupted triples respectively, $count(r)$ is the number of occurrences of the relation r, and $|T|$ is the total number of triples in the merged KG.

On the other hand, the attribute character embedding is designed to learn embeddings for entities from the strings occurring in the attributes associated with the entities. The purpose is to enable the entity embeddings from two KGs to fall into the same vector space despite the fact that the attributes come from different KGs. The attribute character embedding is inspired by the concept of translation in TransE. Given a triple (h, r, a), the data property r is interpreted as a translation from the head entity h to the literal value a i.e. $h + r = f_a(a)$ where $f_a(a)$ is a compositional function. This function encodes the attribute values into a single vector mapping similar attribute values into similar representations. Three different compositional functions SUM, LSTM, and N-gram-based functions have been proposed. SUM is defined as a summation of all character embeddings of the attribute value. In LSTM, the final hidden state is taken as a vector representation of the attribute value. The N-gram-based function, which shows better performance than the others according to their experiments, uses the summation of n-gram combination of the attribute value.

The following objective function is defined for the attribute character embedding:

$$J_{CE} = \sum_{t_a \in T_a} \sum_{t_a' \in T_a'} max(0, [\gamma + \alpha(f(t_a) - f(t_a'))]), \tag{59}$$

$$T_a = < h, r, a > h \in G; f(t_a) = \|h + r - f_a(a)\|,$$

and

$$T_a' = \{< h', r, a > h' \in G\} \cup \{< h, r, a' > a' \in A\},$$

where, T_a and T_a' are the sets of valid attribute triples and corrupted attribute triples with A being the set of attributes in a given KG G. The corrupted triples are created by replacing the head entity with a random entity or the attribute with a random attribute value. Here, $f(t_a)$ is the plausibility score computed based on the embedding of the head entity h, the embedding of the relation r, and the vector representation of the attribute value obtained using one of the compositional functions $f_a(a)$.

The attribute character embedding h_{ce} is used to shift the structure embedding h_{se} into the same vector space by minimizing the following objective function:

$$J_{SIM} = \sum_{h \in G_1 \cup G_2} [1 - \|h_{se}\|_2 \cdot \|h_{ce}\|_2], \quad (60)$$

where, $\|x\|_2$ is the L_2-Norm of vector x. This way the similarity of entities from two KGs is captured by the structure embedding based on the entity relationships and by the attribute embedding based on the attribute values.

All three functions are summed up to an overall objective function J (i.e., $J = J_{SE} + J_{CE} + J_{SIM}$) for jointly learning both structure and attribute embeddings. Finally, the alignment is done by defining a similarity equation with a specified threshold. Moreover, a transitivity rule has been applied to enrich triples in the KGs to get a better attribute embedding result.

TRANSFORMING LITERALS TO ENTITIES Transforming literals of type date and string into entities is another approach introduced recently [112]. In this work, the idea is to enrich the neighborhood structure of entity nodes by transforming literal information into relational triples. This aims to enable reusing existing LP models without modifications in comparison to other existing methods such as *LiteralE*. Three different transformations are described, namely *Literal2Entity*, *Datatype2Entity*, and *Value2Shingles*, which can be implemented as set-operations. Literal2Entity uses the complete literal information without modifications to create a URI, Datatype2Entity reduces the literal information to datatype and language tag and then creates a URI, and Value2Shingles shingles string literals and enriches the graph by these shingles transformed to URIs. These transformations are applied to the existing LP benchmark datasets in order to extend the datasets with additional relational triples. Then, the SOTA KGE models are trained and evaluated on the extended datasets.

The method has several issues: i) It does not consider normalizing the literals before applying the transformations, ii) The Datatype2Entity approach transforms only the data types into entities, disregarding the actual literal values. For example, given the literal node "1862-05-23:xsd:date" the approach transforms only the data type 'xsd:date' to an entity, which leads to losing the semantics present with the actual literal value '1862-05-23'. iii) The transformed datasets may be skewed since all literals only appear at the tail position. v) The Literal2Entity and Value2Shingles approaches may increase the number of relational triples significantly when treating each literal or shingle as an entity, leading to longer training times for the KGE models.

SUMMARY The common drawback with both methods (LiteralE with blocking and EAKGE) is that text and numeric literals are treated in the same way. They also do not consider literal data type semantics or multi-valued attributes in their approach. Furthermore, since EAKGAE is using character-based attribute embedding, it fails to capture the semantics behind the co-occurrence of syllables.

3.2.4.2 *Models with Numeric, Text, and Image Literals*

MKBE MKBE [74] is a multi-modal KGE, in which the text, numeric and image literals are modeled together. The main objective of this approach is to utilize all the observed subjects, objects, and relations (object properties and data properties) in order to predict whether any fact holds. It extends DistMult, which creates embedding for entities and relations, by adding neural encoders for different data types. Given a triple $< s, r, o >$, the head entity s and the relation r are encoded as independent embedding vectors using one-hot encoding through a dense layer. Similarly, if the object o is a categorical value, then it will be represented through a dense layer with a relu activation which has the same number of nodes as the embedding space dimension. On the other hand, if the object o is rather a numerical value, then a feed forward layer, after standardizing the input, is used in order to learn embeddings for o by projecting it to a higher-dimensional space. If o is a short text (such as names and titles), it is encoded using character-based stacked, bidirectional GRUs and the final output of the top layer will be taken as the representation of o. On the contrary, if o is a long text such as entity descriptions, CNN over word embeddings will be used to get the embeddings for o. The object o can also be an image, and in such a case, the last hidden layer of VGG pretrained network on ImageNet [113], followed by compact bilinear pooling, is used to obtain the embedding of o. Given the vector representations of the entities, relations, and attributes, the same scoring function from DistMult is used to determine the correctness probability of triples.

The binary cross-entropy loss, as defined below, is used to train the model:

$$\sum_{(s,r)} \sum_{o} t_o^{s,r} \log(p_o^{s,r}) + (1 - t_o^{s,r}) \log(1 - p_o^{s,r}), \tag{61}$$

where for a given subject relation pair (s, r), binary label vector $t^{s,r}$ over all entities is used to indicate whether $< s, r, o >$ is observed during training. $p_o^{s,r}$ denotes the model's probability of truth for any triple $< s, r, o >$ computed using a sigmoid function.

Moreover, using these learned embeddings and different neural decoders, a novel multi-modal imputation model is introduced to generate missing multi-modal values, such as numerical data, categorical data, text, and images, from information in the knowledge base. In order to predict the missing numerical and categorical data such as dates, gender, and occupation, a simple feed-forward network on the entity embedding is used. For text, the adversarially regularized autoencoder (ARAE) has been used to train generators that decode text from continuous codes, having the generator conditioned on the entity embeddings instead of a random noise vector. Similarly, the combination of BE-GAN structure with pix2pix-GAN model is used to generate images, conditioning the generator on the entity embeddings.

Table 3.2.4: Complexity of the models with multi-modal literals in terms of the number of parameters. Θ is the number of parameters in the base model, H is the entity embedding size, N_d is the number of data relations, N_{char} is the number of characters, and N_i is the number of images, Θ_{CNN} is the number of parameters in the CNN model used in [114], Θ_{ARAE} is the number of parameters in ARAE [115] where instead of using the random noise vector z, the generator is conditioned on the entity embeddings, Θ_{GAN} denotes the sum of the number of parameters in BE-GAN [116] and in pix2pix-GAN [117].

Model	#Parameter
LiteralE with blocking	$\Theta + (N_d H + H)H$
EAKGAE	$\Theta + (N_d + N_{char})H$
MKBE	$\Theta + (2(N_d + 3(N_{char} + H)) + N_i)H$
	$+\Theta_{CNN} + \Theta_{ARAE} + \Theta_{GAN}$

SUMMARY Despite the attempt made in incorporating text literals, numeric literals, and images into the KGE, the model (MKBE) fails to capture the semantics of the data types/units of (numeric) literal values. Besides, similar to IKRL, it takes an image I as an instance of a certain entity e only if, I is initially associated with e in the dataset considered (refer to Section 3.2.3 for more details).

3.3 APPLICATIONS

This section discusses different applications of KG embeddings on which the previously described methods have been trained and/or evaluated.

LINK PREDICTION Most of the models discussed in Section 3.2 have been evaluated on LP tasks. Head and tail prediction are used to evaluate the models LiteralE [73], TransEA [59], KBLRN [91], KDCoE [65], EAKGAE [66], IKRL [61], MKBE [74], MTKGRL [100], Jointly(Desp) [62], Jointly [63], SSP [64], MADLINK [101], and Transforming literals into entities [112]. On the other hand, DKRL [60] has been evaluated on all kinds of LP tasks: head, tail, and relation predictions. In Extended RESCAL [77], two kinds of LP experiments have been conducted on the Yago 2 [118], i.e., i) tail prediction by fixing the relation type to `rdf:type`, and ii) general LP experiments for all relation types. Unfortunately, it is not possible to compare the obtained evaluation results of all these models because the experiments have been carried out on different datasets and also different LP procedures have been followed. Taking this into consideration, in this survey, experiments have been conducted on head and tail prediction tasks for these models (see Section 3.5).

TRIPLE CLASSIFICATION The goal of the triple classification task is the same as that of LP. A potential triple $< h, r, t >$ is classified as 0 (false) or 1 (true), i.e., a binary classification task. The embedding models MTKGNN [75], IKRL [61], MTKGRL [100], Jointly(Desp) [62], Jointly [63], and MADLINK [101] have been evaluated on this task. However, since they do not use a common evaluation dataset, it is not possible to compare the reported results directly.

ENTITY CLASSIFICATION Given a KG G, with a set of entities E and types T and with an entity $e \in E$ and type $t \in T$, the task of entity classification is to determine if a potential entity type pair (e, t) which is not observed in G ($(e, t) \notin G$) is a missing fact or not. This task is an entity type prediction using a multi-label classification algorithm considering the entity types in G as given classes. In DKRL [60], Extended RESCAL [77], and SSP [64], Entity classification has been used for model evaluation.

ENTITY ALIGNMENT Given two KGs G_1 and G_2, the goal of the entity alignment task is to identify those entity pairs (e_1, e_2) where e_1 is an entity in G_1 and e_2 is an entity in G_2 which denote the same real world entities, and hence the integration of G_1 and G_2 can be possible through these unified entities, i.e., entity pairs. Different embedding-based models have been proposed recently for the entity alignment task. Among the models that are included in this survey, EAKGAE [66] and KDCoE [65] have been proposed for the entity alignment task. Specifically, KDCoE [65] uses a cross-lingual entity alignment task which determines similar entities in different languages. Despite the fact that both these models use the same task for evaluation, the entity alignment task, their experimental results cannot be compared since they are based on different datasets.

OTHER APPLICATIONS Attribute-value prediction, nearest-neighbor analysis, data linking, document classification, and relational fact extraction are other application scenarios used for the evaluation of the models under discussion. Attribute-value prediction is the process of predicting the values of (discrete or non-discrete) attributes in a KG. For example, a missing value of a person's weight can be identified using the attribute value prediction task which is commonly seen as a KG completion task. In MTKGNN [75], attribute-value prediction is applied using an attribute-specific Linear Regression classifier for evaluation. The same task has been employed in MKBE [74] for model evaluation by imputing different multi-modal attribute values.

Nearest Neighbor Analysis is a task of detecting the nearest neighbors of some given entities in the latent space learned by an embedding model. This task has been performed in LiteralE [73] to compare DistMult+LiteralE with the base model DistMult. On the other hand, data linking and document classification tasks have been used in LiteralE with blocking [78] and KGlove with literals [76] respectively (refer to [78] and [76] for more details). Relational fact extraction is a task of extracting facts/triples from plain text and has been used as a model evaluation task in Jointly(Desp) [62]. Table A.1.1 summarizes all the applications on which the KG embedding models with literals have been evaluated in their respective original studies.

3.4 EVALUATION BENCHMARK DATASETS

In order to advance the SoTA in KGE (or in KGC in general), it is crucial to thoroughly investigate the benchmark datasets utilized for evaluation. There are different commonly known benchmark datasets available that have been used for the evaluation of the performance of various KGE approaches, mainly LP models. A summary of these benchmarks is given in Table 3.4.1. The sources of the majority of these datasets are Freebase [3], WordNet [29], YAGO [27], Wikidata [2], and NELL [119].

FREEBASE EXTRACTS FB15K [49] and FB15K-237 [15] are among the most popular datasets to evaluate KGC models. Even though the original releases of both datasets do not include any attributive triples, they have been extended with textual and numerical attributes [60, 73, 120]. However, different studies [1, 16, 121] have claimed that FB15K does not possess the required qualities to be actually used as a benchmark, i.e., it contains multiple inverse relations. On the other hand, in FB15K-237 which is a subset of FB15K without inverse relations, all validation and test triples containing entity pairs directly linked in the training set have been removed. Moreover, FB15K-237 contains a significant amount of triples with skewed relations towards either some head or tail entities [121] (more details are provided in Section 6.3 of Chapter 6 where the set of benchmark datasets proposed in this thesis for transductive LP is presented).

WORDNET EXTRACTS Among the WordNet datasets, WN18 [49] and WN18RR [16] are the most popular ones. Both datasets are smaller in size and domain-specific as compared to the other datasets such as FB15K-237. Besides, the original releases do not contain any numerical attributive triples.

YAGO EXTRACTS YAGO3-10 [16] is the widely used dataset among those extracted from YAGO. It is a dataset that contains only relational triples from YAGO3 [26] mostly about locations and people. The dataset has been extended with numerical attributes, textual entity descriptions, and entity images in [74] and only with numerical attributes in [73]. Most of all, as discussed in [122], YAGO3-10 has a significant number of triples with two duplicate relations *isAffiliatedTo* and *playsFor* which makes the dataset easy for a LP task.

WIKIDATA (AND WIKIPEDIA) EXTRACTS Wikidata-authors [123] is a domain-specific dataset containing relational triples from Wikidata where the head entities are persons who are authors or writers. Apart from having a narrow scope and a small set of triples (i.e., 86,376), this dataset does not contain any attributive triples. CoDEx [121] is a recent KGC benchmark extracted from Wikidata and Wikipedia. The relational triples in this dataset are from Wikidata and the attributive triples have been provided as auxiliary information taken from both Wikidata and Wikipedia. The auxiliary information contains Wikidata labels, descriptions of entities and relations along with Wikipedia page extracts for entities. This dataset does not include any numeric attribute and while attempting to retrieve them from Wikidata, there is only a limited number of entities in the dataset with numeric attributes. Moreover, in CoDEx the set of triples already contains classes and this may decrease the level of difficulty of the dataset for tasks other than LP and triple classification which involve classes, i.e., entity typing/classification.

OTHERS There are other datasets such as NELL-995 [124] and MovieLens [74] (see Table 3.4.1 for more details). NELL-995 is a dataset extracted from the 995th iteration of NELL [119]. Due to the fact that the triples in NELL-995 are nonsensical or overly generic, the dataset is not suitable to be used as a KGC benchmark [121]. Moreover, the dataset does not contain any attributive triples. MovieLens [74] is a dataset about movies where relational triples, numerical attributes, and textual attributes are from ML100K [125] and images are movie posters from TMDB[1]. This dataset contains few entities, relations, and triples as compared to the widely used KGC datasets, such as FB15K-237. Moreover, not all of the entities are associated with textual attributes that describe them. Another very recently released benchmark is Kgbench [126] which could be used for both node classification and LP. However, baseline results are only provided for node classification task because the datasets are generated primarily for that particular task. Kgbench provides a set of different domain-specific datasets and in each dataset, the source for the multimodality is mainly images hence, numeric literals are available only for a limited number of entities.

SUMMARY In general, the existing KGC benchmarks do not give proper emphasis to attributive triples, i.e., attributes are treated as auxiliary information. Consequently, the attributive triples are either way unbalanced, less in number, or have few unique attributes. Therefore, in Chapter 6 of this thesis, a new KGC benchmark called **LiterallyWikidata** is presented which properly handles literals, specifically, numerical attributes and textual descriptions.

1 https://www.themoviedb.org/

Table 3.4.1: Existing KGC datasets for the task of LP.

Dataset	**Sources**	**Domain**: Specific (•) Generic (⋆)	**Attributive triples**: Text (•), Numerical (⋆), Image (✓)	
			Original	Extended
CoDEx [121]	Wikidata [2], Wikipedia	⋆	•	
Wikidata-authors [123]	Wikidata	•		
FB15K [49]	Freebase	⋆		• [60] ⋆[120] ⋆[73]
FB15K-237 [15]				⋆[73] • [73]
FB15k-237-OWE [127]			•	
FB20K [60]			•	
FB13 [128]				
FB5M [129]				
FB24K [130]				
FB15K-401 [50]				
WN18 [49]	WordNet [29]	•		
WN18RR [16]				
WN11 [128]				
YAGO3-10 [16]	YAGO	⋆		⋆[73] • ⋆ ✓ [74]
YAGO37 [131]				
YG58K [120]				⋆[120]
NELL-995 [124] and other Nell varieties [132]	NELL [119]			
MovieLens [74]	ML 100K [125], TMDB[a]	•	• ⋆ ✓	
UMLS [133]	UMLS [134]			
kinship [133]	Alyawarra kinship [135]			
Nations [133]	Nations Project [136]			
Countries [137]	Countries data[b]			
Family [87, 138]	Families [139]			

[a] https://www.themoviedb.org/

[b] https://github.com/mledoze/countries

3.5 EXPERIMENTS ON LINK PREDICTION

This section provides an empirical evaluation of the methods discussed in Section 3.2 on the task of LP under a unified environmental setting. In this work, LP is chosen because most of the KG embedding models with literals are trained and evaluated on it. One of the major issues encountered while conducting these experiments is that the source code of some of these models is not openly available and is not easily reproducible. Such methods were excluded from the experimentation. In the subsequent sections, the datasets and the experiments with text, numeric, images, and multi-modal literals are presented.

Based on the results of the experiments, a clear comparison is presented between the models with literals on LP. In addition, these models are also compared with the standard KG embedding approaches that they extend. Note that these models may inherit the problems that already exist in their corresponding base models (i.e., the standard KG embedding models that they extend). For instance, the models that extend DistMult such as DistMult-LiteralE$_g$ inherit the problem of DistMult, which is not capable of properly capturing anti-symmetric relations due to the way its scoring function is defined.

3.5.1 *Datasets*

The performances of the aforementioned models are evaluated using two of the most commonly used datasets for LP, i.e., FB15K [49] and FB15K-237 [15] (refer to Section 3.4 for more details on these datasets). The statistics of these datasets are given in Table 3.5.1.

Table 3.5.1: The number of entities, object relations, data relations, relational triples, train sets, valid sets, and test sets of the FB15K and the FB15K-237 datasets.

	Datasets	
	FB15K	**FB15K-237**
Entities	14951	14541
Object Relations	1345	237
Relational triples	592213	310116
Train sets	483142	272115
Valid sets	50000	17535
Test sets	59071	20466

3.5.2 *Experiments with Text Literals*

As discussed in Section 3.2.1, the embedding models Extended RESCAL, DKRL, KDCoE, and KGloVe with literals utilize text literals. However, all of these models except DKRL are not considered for experimentation due to the following issues:

- The implementation of the model KGloVe with literals is not publicly available and it is not easily reproducible.
- KDCoE is designed specifically for cross-lingual entity alignment task which makes it difficult to apply it for LP.
- MADLINK was introduced after this survey was already conducted and published.
- In the case of Extended RESCAL, practically this method is computationally expensive and thus not considered a feasible embedding model to incorporate literals.

  Moreover, none of the models discussed in this chapter consider Extended RESCAL in their experiments.

In order to conduct experiments with text literals, 15239 English entity descriptions of the entities common in both datasets FB15K and FB15K-237 shown in Table 3.5.1 are taken from LiteralE [73]. The focus lies on the common entity descriptions, i.e., for those entities existing in FB15K but not in FB15K-237 no description is used, because there has already been experiments done using the whole entity descriptions for FB15K dataset in the original paper. This way it would be possible to analyse the effect of the size of the dataset (the entity descriptions) on the performance of the embedding models. The average number of words (tokens) in the descriptions is 143 whereas the maximum and minimum are 804 and 2.

DATASET PRE-PROCESSING For pre-processing of the text (the entity descriptions), spacy.io[2] has been used. This includes tokenization, named entity recognition and conversion of numbers to text, i.e., 16 has been converted to 'sixteen'. After the pre-processing step, all the entities along with the corresponding triples having no or short description of less than 3 words are removed. Also, the triples containing these entities are removed as mentioned by the authors in the paper. Moreover, only one description is chosen randomly for the entities with multiple text descriptions.

EXPERIMENTAL SETUP The hyperparameters used for DKRL are as follows: learning rate 0.001, embedding size 100, loss margin 1, batch size 100 and epochs 1000. For TransE, learning rate 0.01, embedding size 50, margin 1, and epochs 1000 are used. The experiments with DKRL were performed on Ubuntu 16.04.5 LTS system with 503GiB RAM and 2.60GHz speed. On the other hand,the experiments with TransE are performed with TITAN X (Pascal) GPU.

2 https://spacy.io/usage

RUNTIME Note that the codes used in the experiments for both models DKRL and TransE are not implemented in the same environment, i.e., for DKRL, the code that is released by the authors of the paper is used and for TransE the code provided by the authors of TransEA [59] is used. Therefore, it is not fair to compare the runtime results of these two models directly. However, in order to provide some insights into the computational complexity of the models, their runtime results on the FB15K dataset are given as follows. DKRL takes 142 seconds to train 1 epoch using 16 threads whereas the runtime of TransE for a single iteration with batch size 4831 is 3.271 millisecond. This is computed by taking the average of 1000 iterations.

EVALUATION PROCEDURE AND RESULTS The performance of the model is evaluated based on the LP task. For each triple in the test set, a set of corrupted triples is generated with respect to the head or the tail entity. A triple is said to be corrupted with respect to its head entity if that head entity is replaced with any other entity from the KG, and analogously, a triple is corrupted with respect to its tail entity. The set of corrupted triples can also contain true triples that exist in the training, validation, or test set. Since it is not an error to give these true triples better scores than the actual test triple, they are removed from the set of corrupted triples and this is referred to as a filtered setting [49]. In order to check if the model assigns a better score to the actual test triple than the corrupted triples which are obtained by corrupting the test triple, it is evaluated using the metrics MR (Mean Rank), MRR (Mean Reciprocal Rank), and Hits@N (refer to Section 2.8 of Chapter 2 for details about these metrics). First, for every test triple, all of its corrupted triples with respect to head are ranked based on their scores which are computed by the model. Then, the ranks of the actual (true) test triples are taken in order to compute the metrics MR, MRR, and Hits@N. The same procedure is repeated to evaluate the model against the corrupted triples with respect to tails.

The results of LP on FB15K and FB15K-237 datasets are shown in Table 3.5.2 for the models TransE, DKRL with Bernoulli distribution (DKRL$_{Bern}$), and DKRL with Uniform distribution (DKRL$_{unif}$). The Bernoulli distribution for sampling as defined in [54] is a probability distribution, $\frac{tph}{tph+hpt}$, where tph is the average number of tail entities per head entity and hpt is the average number of head entities per tail entity. Given a golden triple $< h, r, t >$, with the aforementioned probability, the triple is corrupted by replacing the head, and with probability $\frac{hpt}{tph+hpt}$, the triple is corrupted by replacing the tail. The results are reported separately for the head entity and tail entity along with the overall results obtained by taking the mean of the head and tail predictions. The best scores are the ones which are highlighted in bold text. The result of the TransE model is presented in order to allow a clear comparison with DKRL because, as shown in Table 3.1.2, DKRL extends TransE. This comparison would help to further analyse the advantages of using text literals for KG embedding.

Note that in the original paper, the result of DKRL on FB15K is slightly better than TransE. However, in our experiments, as the results in Table 3.5.2 indicate, on the FB15K dataset TransE achieves better result than both versions of DKRL on all metrics except MRR and MR. The reason for this is that, as mentioned above, the set of entity descriptions used in our experiments are common for both datasets FB15K and FB15K-237, i.e., there is less entity descriptions in our experiment than there is in the original paper for FB15K. This indicates that the size of the dataset (the entity description has impact on the performance of the model). On the other hand, on the dataset FB15K-237 TransE is outperformed by $\text{DKRL}_{\mathtt{Unif}}$ with respect to MR and by $\text{DKRL}_{\mathtt{Bern}}$ with respect to the rest of the metrics.

Furthermore, the result shows that DKRL model with Bernoulli distribution ($\text{DKRL}_{\mathtt{Bern}}$) has better performance than the model with Uniform distribution ($\text{DKRL}_{\mathtt{unif}}$) for both the datasets. $\text{DKRL}_{\mathtt{Bern}}$ works best for the prediction of head, relation, and tail with respect to MRR, Hits@1, and Hits@3 whereas the $\text{DKRL}_{\mathtt{Unif}}$ method works better according to MR for both the datasets. $\text{DKRL}_{\mathtt{Bern}}$ works slightly better than $\text{DKRL}_{\mathtt{Unif}}$ for FB15K-237 dataset. It is to be noted that DKRL has better improvement over TransE on FB15K-237 as compared to FB15K dataset because the former one does not contain symmetric relations, i.e., incorporating textual data to a clean dataset, such as FB15K-237, allows capturing more semantics.

3.5.3 *Experiment with Numeric Literals*

MT-KGNN, KBLRN, LiteralE, and TransEA are the KG embedding models which make use of numeric literals (see Section 3.2.2). KBLN, the submodel of KBLRN, which excludes the relational information provided by graph feature methods is used in the experiment instead of the main model KBLRN. This is the case because KBLN is directly comparable with the other three models (i.e., MT-KGNN, LiteralE, and TransEA) whereas KBLRN is not. The code[3] for the TransEA model is the original implementation from TransEA [59] where as the source codes[4] for the models MT-KGNN, KBLN, and LiteralE are taken from the implementation in LiteralE [73]. As described in Section. 3.2.2, the structure-based embedding component of MT-KGNN is based on a neural network and it is referred to as RelNet. However, in the version implemented in LiteralE [73], the authors have replaced RelNet with DistMult as a baseline in order to have a directly comparable MT-KGNN-like method to their proposed approach. Thus, in this survey, the MT-KGNN-like model has been used instead of the original MT-KGNN model.

Moreover, the model LiteralE has different varieties depending on the baseline model and the transformation function used. As discussed in Section 3.2, in LiteralE there are two transformation functions: $\mathtt{g}$ (GRU based function) and $\mathtt{lin}$ (a simple linear function), and there are three baseline models - DistMult, ConvE, and ComplEx. Thus, in this experiment, six varieties of the LiteralE model are considered: DistMult-$\text{Literale}_{\mathtt{g}}$, ComplEx-$\text{Literale}_{\mathtt{g}}$, ConvE-$\text{Literale}_{\mathtt{lin}}$, DistMult-$\text{Literale}_{\mathtt{lin}}$, ComplEx-$\text{Literale}_{\mathtt{lin}}$, and ConvE-$\text{Literale}_{\mathtt{lin}}$. The

3 https://github.com/kk0spence/TransEA
4 https://github.com/SmartDataAnalytics/LiteralE

Table 3.5.2: Experiment results using DKRL model on FB15K and FB15K-237 datasets.

FB15K						
		MR	MRR	Hits@1	Hits@3	Hits@10
TransE	Head	142	0.219	**0.241**	**0.447**	**0.622**
	Tail	109	0.249	**0.339**	**0.514**	**0.690**
	All	125	0.234	**0.290**	**0.480**	**0.656**
$\text{DKRL}_{\text{Bern}}$	Head	162	**0.289**	0.179	0.336	0.502
	Tail	122	**0.356**	0.24	0.408	0.577
	All	142	**0.322**	0.209	0.372	0.539
$\text{DKRL}_{\text{Unif}}$	Head	**96**	**0.289**	0.172	0.335	0.52
	Tail	**75**	0.333	0.211	0.383	0.576
	All	**85**	0.311	0.191	0.359	0.548
FB15K-237						
		MR	MRR	Hits@1	Hits@3	Hits@10
TransE	Head	468	0.094	0.081	0.163	0.287
	Tail	255	0.190	0.233	0.373	0.517
	All	361	0.142	0.157	0.268	0.402
$\text{DKRL}_{\text{Bern}}$	Head	145	**0.294**	**0.184**	**0.337**	**0.507**
	Tail	98	**0.359**	**0.244**	**0.410**	**0.585**
	All	122	**0.327**	**0.214**	**0.374**	**0.546**
$\text{DKRL}_{\text{Unif}}$	Head	**104**	0.275	0.166	0.312	0.494
	Tail	**77**	0.322	0.209	0.363	0.552
	All	**91**	0.298	0.187	0.337	0.523

datasets, the experimental setup, and the evaluation results are discussed in the subsequent sections.

ATTRIBUTIVE TRIPLES In order to conduct the experiments with numeric literals, both the datasets FB15K and FB15K-237 given in Table 3.5.1 are extended with a set of 23521 attributive triples, containing only numeric literals, with 118 data relations. These triples are created based on the attributive triples from TransEA [59]. In TransEA, the authors have provided a set of attributive triples where the object values are numeric. However, it is not possible to directly use this data as the literal values are normalized in the interval [0-1] as required by the model but the other models in this experiment use the original un-normalized literal values instead. Therefore, it was necessary to query Freebase to replace the normalized object literal value for each (subject, data relation) pair from the TransEA

Table 3.5.3: Runtime of models considered in the experiments with numeric literals. The resutls are per single iteration and reported in milliseconds.

	Time(ms)
DistMult-LiteralE$_{lin}$	31.575
DistMult-LiteralE$_{g}$	37.138
ComplEx-LiteralE$_{lin}$	39.269
ComplEx-LiteralE$_{g}$	52.346
ConvE-LiteralE$_{lin}$	43.386
ConvE-LiteralE$_{g}$	50.439
KBLN	86.825
DistMult	29.679
ComplEx	33.526
ConvE	40.970

attributive triples data. Moreover, only those data relations which occur in at least 5 triples are taken into consideration.

EXPERIMENTAL SETUP For both datasets, the hyperparameters for TransEA are: epoch 3000, dimension 100, batches 100, margin 2, and learning rate 0.3 and for TransE they are described in Section 3.5.2. For the other models, same as in LiteralE, the hyperparameters used for both datasets are: learning rate 0.001, batch size 128, embedding size 100 (for DistMult, ComplEx and their extensions with literals) and 200 (for KBLN, and MT-KGNN, ConvE, and ConvE's extensions), embedding dropout probability 0.2, label smoothing 0.1, and epochs 1000 for ConvE and 100 for the rest. TITAN X (Pascal) GPU has been used for the models LiteralE, KBLN, and MT-KGNN.

RUNTIME As in the experiments with text literals, not all the models in the experiments with numeric literals are implemented in the same environment, i.e., for TransEA the code that is released by the authors of the paper is used and for the other models the code that is provided by the authors of LiteralE are used. Therefore, direct comparison of the runtime of TransEA and the other models would not be possible. However, the runtime of each of the models is computed on FB15K dataset so as to give insights into the models computational complexity. The running time of TransEA is 3.271 ms per a single iteration with batch size of 4831. For the other models their runtime for a single iteration of batch size 128 is shown in Table 3.5.3. Note that the runtime results reported here are the average over runtime values of 1000 iterations.

EVALUATION PROCEDURE AND RESULTS The same evaluation metrics discussed in Section 3.5.2 are applied to evaluate the performance of models using numeric literals. As shown in Table 3.5.4, according to the overall result, the model KBLN has considerably better performance than the other models in all metrics except MR. The results from the ComplEx-LiteralE$_g$ model show that it is capable to produce a highly competitive performance having the second-best results with respect to the same metrics. This is the case due to the fact that this model is able to handle the inverse relations in FB15K by applying the complex conjugate of an entity embedding when the entity is used as a tail and its normal embedding when it is the head. Moreover, the model KBLN also achieves better results when compared to the standard models without literals presented in Table 3.5.5 with respect to all metrics except MR.

Another possible analysis to make is to compare the results of the standard models presented in Table 3.5.5 with the results of their extensions shown in the 'both head and tail Prediction' part of Table 3.5.4. For instance, ComplEx-LiteralE$_g$ achieves better performance than its base model ComplEx according to all metrics which indicates that using numeric literals with ComplEx by applying the approach in LiteralE is beneficial. However, this is not the case with DistMult and ConvE. One reason for this can be the fact that the number of attributive triples used in our experiment is not as big as in the original paper of LiteralE, i.e, increasing the number of numeric literals may improve the result as already seen in the original paper of literalE.

On the other hand, referring to the overall result on FB15K-237 dataset as shown in Table 3.5.6, the model DistMult-LiteralE$_g$ outperforms the other models according to all metrics. This entails that applying LiteralE to DistMult on FB15K-237 provides better performance than applying it to other baseline models. It should be noted that the DistMult-LiteralE$_g$ model outperforms the DistMult model on the FB15K-237 dataset. This could be attributed to the absence of any symmetric relation in this dataset. While the DistMult model struggles to model asymmetric relations on FB15K, the addition of literals might introduce noise. However, in the case of FB15K-237, the incorporation of literals improves the performance of DistMult because symmetric relations are absent. Regarding the two transformation functions g and lin, the function g leads to better results than lin according to the results on both dataset.

3.5.4 *Experiment with Images*

Note that it is not possible to compare the whole of MKBE [74] with any other model as it is the only embedding model which utilizes text literals, numeric literals, and images together. Therefore, its sub-model *S+I* which uses only images has been compared with the embedding model IKRL [61]. Since this comparison has already been done by the authors of MKBE [74], the result shown in Table 3.5.8 is directly taken from their paper. They have compared the models DistMult+S+I, ConvE+S+I, and IKRL where S stands for structure and I for Image. Both DistMult+S+I and ConvE+S+I are sub-models of MKBE which use

Table 3.5.4: LP results on FB15K dataset using filtered setting.

Head Prediction					
Models	MR	MRR	Hits@1	Hits@3	Hits@10
DistMult-LiteralE$_{lin}$	121	0.495	0.383	0.559	0.697
ComplEx-LiteralE$_{lin}$	71	0.76	0.697	0.801	0.876
ConvE-LiteralE$_{lin}$	52	0.612	0.51	0.678	0.795
DistMult-LiteralE$_{g}$	72	0.581	0.479	0.642	0.762
ComplEx-LiteralE$_{g}$	63	0.768	**0.707**	0.809	0.878
ConvE-LiteralE$_{g}$	**49**	0.72	0.65	0.762	0.849
KBLN	77	**0.775**	0.705	**0.827**	**0.892**
MT-KGNN	73	0.702	0.617	0.758	0.855
TransEA	103	0.285	0.367	0.609	0.728
Tail Prediction					
Models	MR	MRR	Hits@1	Hits@3	Hits@10
DistMult-LiteralE$_{lin}$	145	0.447	0.337	0.507	0.645
ComplEx-LiteralE$_{lin}$	101	0.704	0.64	0.743	0.821
ConvE-LiteralE$_{lin}$	**74**	0.567	0.465	0.63	0.746
DistMult-LiteralE$_{g}$	94	0.528	0.425	0.589	0.712
ComplEx-LiteralE$_{g}$	93	0.711	0.65	0.746	0.821
ConvE-LiteralE$_{g}$	79	0.657	0.586	0.698	0.783
KBLN	90	**0.727**	**0.656**	**0.776**	**0.848**
MT-KGNN	91	0.65	0.562	0.708	0.806
TransEA	75	0.314	0.417	0.671	0.805
Both Head and Tail Prediction					
Models	MR	MRR	Hits@1	Hits@3	Hits@10
DistMult-LiteralE$_{lin}$	133	0.471	0.36	0.533	0.671
ComplEx-LiteralE$_{lin}$	86	0.732	0.668	0.772	0.848
ConvE-LiteralE$_{lin}$	**63**	0.589	0.487	0.654	0.77
DistMult-LiteralE$_{g}$	83	0.554	0.452	0.615	0.737
ComplEx-LiteralE$_{g}$	78	0.739	0.678	0.777	0.849
ConvE-LiteralE$_{g}$	64	0.688	0.618	0.73	0.816
KBLN	83	**0.751**	**0.68**	**0.801**	**0.87**
MT-KGNN	82	0.676	0.589	0.733	0.83
TransEA	74	0.299	0.392	0.64	0.766

Table 3.5.5: LP results with models without literals on FB15K using filtered setting.

FB15K					
Models	MR	MRR	Hits@1	Hits@3	Hits@10
DistMult	119	0.67	0.589	0.723	0.817
ComplEx	127	**0.692**	**0.614**	0.742	0.833
ConvE	**50**	0.689	0.593	**0.757**	**0.852**
TransE	125	0.234	0.290	0.480	0.656

only relational triples and Images. The result indicates that ConvE+S+I outperforms the other two models in all metrics on the YAGO-10 dataset (refer to MKBE [74] for more details).

3.5.5 *Experiment with Multi-modal Literals*

As discussed in Section 3.2, the existing multi-modal embeddings are categorized into two groups: i) models that use text literals, numeric literals, and images and ii) models with text and numeric literals. However, since MKBE is the only one in the first category, only its submodel $S + I$ can be compared to IKRL (as explained in Section 3.5.4). Regarding the models with text and numeric literals, i.e., *LiteralE with blocking*, *EAKGAE*, and *transforming literals into entities*, they are not included in the experiment as well. The issue with EAKGAE is the same as that of KDCoE, i.e., it is trained on an entity alignment task whereas *LiteralE with blocking* is not included as its code is not publicly available. The approach *transforming literals into entities* is not part of the experiment because it was released after the survey was conducted. However, the LiteralE model, which leverages numeric literals, is adapted to also make use of textual literals along with numerical literals, leading to the creation of the *DistMult-LiteralE$_g$-text* model, as presented in Table 3.5.9. Note that DistMult is chosen here as a baseline due to the fact that the best result in the experiments with numerics on the FB15K-237 dataset is achieved using this model as discussed in Section 3.5.3.

The datasets listed in Table 3.5.1 are also used for this experiment along with additional text attributive triples which are descriptions of entities. As shown in Table 3.5.9, the results are compared with LiteralE with just numeric literals (DistMult-LiteralE$_g$) and DKRL (a model using only text literals) so as to investigate the benefits of utilizing information represented by different types of literals. As the results indicate, combining text and numeric literals on FB15K dataset with DistMult-LiteralE$_g$-text approach does not produce better results as compared to the other models DistMult-LiteralE$_g$ and DKRL$_{\mathtt{Bern}}$. As mentioned before, this dataset contains a set of inverse relations which may lead to having a triple whose inverse has a different label. Given the fact that DistMult fails to model such asymmetric relations, incorporating more literals with DistMult may introduce much noise than improving the performance. On the other hand, for FB15K-237 dataset, according to

Table 3.5.6: LP results on FB15K-237 dataset using filtered setting.

Head Prediction

Models	MR	MRR	Hits@1	Hits@3	Hits@10
DistMult-LiteralE$_{lin}$	245	0.377	0.279	0.422	0.568
ComplEx-LiteralE$_{lin}$	371	0.36	0.271	0.4	0.538
ConvE-LiteralE$_{lin}$	**208**	0.388	0.296	0.427	0.572
DistMult-LiteralE$_{g}$	209	**0.413**	**0.320**	**0.456**	**0.591**
ComplEx-LiteralE$_{g}$	315	0.366	0.277	0.404	0.543
ConvE-LiteralE$_{g}$	236	0.317	0.229	0.345	0.501
KBLN	381	0.386	0.295	0.426	0.564
MT-KGNN	437	0.383	0.295	0.423	0.559
TransEA	389	0.111	0.094	0.197	0.342

Tail Prediction

Models	MR	MRR	Hits@1	Hits@3	Hits@10
DistMult-LiteralE$_{lin}$	426	0.195	0.119	0.214	0.349
ComplEx-LiteralE$_{lin}$	575	0.17	0.104	0.185	0.306
ConvE-LiteralE$_{lin}$	362	0.187	0.112	0.204	0.338
DistMult-LiteralE$_{g}$	359	**0.215**	0.137	0.234	0.371
ComplEx-LiteralE$_{g}$	493	0.175	0.106	0.19	0.312
ConvE-LiteralE$_{g}$	459	0.131	0.07	0.137	0.256
KBLN	501	0.207	0.128	0.23	0.362
MT-KGNN	580	0.191	0.12	0.208	0.338
TransEA	**203**	0.206	**0.25**	**0.409**	0.57

Both Head and Tail Prediction

Models	MR	MRR	Hits@1	Hits@3	Hits@10
DistMult-LiteralE$_{lin}$	335	0.286	0.199	0.318	0.458
ComplEx-LiteralE$_{lin}$	473	0.265	0.187	0.292	0.422
ConvE-LiteralE$_{lin}$	285	0.287	0.204	0.315	0.455
DistMult-LiteralE$_{g}$	**284**	**0.314**	**0.228**	**0.345**	**0.481**
ComplEx-LiteralE$_{g}$	404	0.27	0.191	0.297	0.427
ConvE-LiteralE$_{g}$	347	0.224	0.149	0.241	0.378
KBLN	441	0.296	0.211	0.328	0.463
MT-KGNN	508	0.287	0.207	0.315	0.448
TransEA	296	0.158	0.172	0.303	0.456

Table 3.5.7: LP results with models without literals on FB15K-237 dataset using filtered setting.

Models	MR	MRR	Hits@1	Hits@3	Hits@10
DistMult	630	0.280	0.201	0.309	0.438
ComplEx	623	0.288	0.207	0.318	0.448
ConvE	**273**	**0.310**	**0.222**	**0.343**	**0.484**
TransE	361	0.142	0.157	0.268	0.402

Table 3.5.8: MRR results on LP task on YAGO-10 taken from MKBE [74].

	YAGO-10			
Models	MRR	Hits@1	Hits@3	Hits@10
DistMult+S+I	0.342	0.235	0.352	0.618
ConvE+S+I	**0.566**	**0.471**	**0.597**	**0.72**
IKRL	0.509	0.423	0.556	0.663

all the measures except MR, DistMult-LiteralE$_g$-text model works better for the head entity prediction compared to the other two models. For tail entity prediction, DKRL$_{Bern}$ works better with respect to all measures for the same dataset.

3.6 DISCUSSION AND OUTLOOK

In this chapter, an extensive survey of the existing KGE models with literals is presented. The survey provides a detailed analysis and categorization of these models based on the proposed methodology along with their application scenarios and limitations. Additionally, an analysis of the existing KGC benchmark datasets and a set of LP experiments comparing the performances of the SoTA models are included. As previously mentioned, this survey is conducted in order to address the research question 'C_2 : RQ_1 - *How well do the SoTA KGE approaches which use literals perform for the task of LP?*' defined in Section 1.2 of Chapter 1. In order to answer the research question, the results of the survey are summarized below, by discussing the techniques utilized to integrate relational and attributive triples, demonstrating the common approaches for combining different types of literals, and identifying the limitations of current KGE approaches.

COMBINING RELATIONAL TRIPLES AND ATTRIBUTIVE TRIPLES In order to use both data sources, i.e., relational and attributive triples together for representation learning, in broader terms, the following two techniques are considered in the models discussed in this survey:

Table 3.5.9: LP results on FB15K and FB15K-237 datasets using filtered set.

FB15K							
	Models	MR	MRR	Hits@1	Hits@3	Hits@10
Head	DistMult-LiteralE$_g$	**72**	**0.581**	**0.479**	**0.642**	**0.762**
	DKRL$_{Bern}$	162	0.289	0.179	0.336	0.502
	DistMult-LiteralE$_g$-text	93	0.516	0.405	0.582	0.711
Tail	DistMult-LiteralE$_g$	**94**	**0.528**	**0.425**	**0.589**	**0.712**
	DKRL$_{Bern}$	122	0.356	0.24	0.408	0.577
	DistMult-LiteralE$_g$-text	119	0.463	0.351	0.532	0.66
All	DistMult-LiteralE$_g$	**83**	**0.554**	**0.452**	**0.615**	**0.737**
	DKRL$_{Bern}$	142	0.322	0.209	0.372	0.539
	DistMult-LiteralE$_g$-text	106	0.489	0.378	0.557	0.685

FB15K-237						
	Models	MR	MRR	Hits@1	Hits@3	Hits@10
Head	DistMult-LiteralE$_g$	209	0.413	0.320	0.456	0.591
	DKRL$_{Bern}$	**145**	0.294	0.184	0.337	0.507
	DistMult-LiteralE$_g$-text	207	**0.416**	**0.323**	**0.462**	**0.594**
Tail	DistMult-LiteralE$_g$	359	0.215	0.137	0.234	0.371
	DKRL$_{Bern}$	**98**	**0.359**	**0.244**	**0.410**	**0.585**
	DistMult-LiteralE$_g$-text	354	0.223	0.142	0.246	0.385
All	DistMult-LiteralE$_g$	284	0.314	0.228	0.345	0.481
	DKRL$_{Bern}$	**122**	**0.327**	0.214	**0.374**	**0.546**
	DistMult-LiteralE$_g$-text	280	0.319	**0.232**	0.354	0.489

- *Handling literals separately:* defining one task per data source like in TransEA or using a separate encoder for literals as in DKRL. The two tasks are trained simultaneously to make sure that for every entity the semantics available in both data sources are used to learn its embedding. The embeddings of the entities learned based on each data source can be unified in the vector space or not depending on how the model works. For instance, Jointly(Disp) learns unified representation for entities whereas DKRL generates two representations per entity and does not force them to be unified.
- *Incorporating literals directly into entity embeddings:* as in LiteralE, one way is to extend a certain latent feature method by directly enriching the embeddings with information from literals via a learnable parameter and use the same scoring function from the latent feature method.

Capturing and combining the heterogeneous types of literals The following are some possible ways to combine different kinds of literals, i.e., text, numeric, etc. together for representation learning.

- *Encoding each type of literal separately:* In order to capture the semantics of literals, different encoders can be used for different types of literals, for example, CNN for textual descriptions. Then, as shown in MKBE, each attributive triple can be treated the same as structured triples and use a single scoring function for training.
- *Incorporating information present in every kind of literal directly into the entity embedding:* as in LiteralE, for a given entity, first the literals associated with it are encoded as vectors - using one vector per type of literal. Then, a mapping function is used to map all these vectors (including the structure-based vector representation of the entity) into a single vector.

Summary of the limitations of the SOTA models: As mentioned in Section 3.2 or seen from the result of the experiments in Section 3.5, these embedding models have different drawbacks such as:

- The effect that data types/units have on the semantics of literals has not been fully leveraged by the models.
- Most of the embedding models which make use of numerical literals, such as LiteralE, TransEA, MT-KGNN, and KBLN consider only the year part of date typed literals and ignore the month and day values. This hinders the ability to properly capture the information represented in such kind of literal. For example, given the following three date typed literal values:

  ```
  "1999-10-29"^^xsd:date,
  "1999-04-14"^^xsd:date,
  "1999-10-30"^^xsd:date,
  ```

  a model which utilizes only the year part of these values considers all of them to be exactly the same despite the fact that the first date value is closer to the third value than it is to the second value.
- Most of the models also do not have a proper mechanism to handle multi-valued attributes, i.e., they randomly select one value for each entity-attribute pair.
- The performance of most of the models is dependent on the dataset used for training and testing which shows that these models are not robust. For example, referring to Table 3.5.9, the results of the model DistMult-LiteralE$_g$-text indicate that combining text and numeric literals yields better performance on FB15K-237 but not on FB15K due to the technique used in the model and the nature of the datasets (see Section 3.5.5).

- Not all the models are effective in combining different types of literals. For example, the performance of DistMult-LiteralE$_g$-text (numeric + text literals), which combines text and numeric literals, on the dataset FB15K is lower as compared to DistMult-LiteralE$_g$ (only numeric literals).
- Only a few approaches have been proposed for multi-modal KGEs and none of them take into consideration literals with URIs connected to items such as audio, video, or pdf files.

Therefore, to put it concisely, even though some studies have been made in an attempt to leverage literals for the task of KGE, there remain significant gaps in fully utilizing the semantics present in different kinds of literals and the information associated with them such as data types.

The answer provided above for the research question $C_2 : RQ_1$ clearly emphasizes the need for further exploration on how to handle various types of literals that obtain different inherent semantics, as indicated by the mentioned limitations of current models. For instance, a possible perspective that arises from this detailed analysis is that there is a need to properly handle the data typed literals such as the values of the data relation *weight* given in *kilogram* and *pound*. One possible solution to target this issue could be to normalize these literal values to standardized measures and to treat different measures like weights and lengths separately in the representation learning process. One cannot expect that by leaving out available information present in the original KG, its latent representation as being only an approximation of the original KG, will perform equally well on tasks that depend on its semantic information content. Overall, the inclusion of datatyped literals with a proper representation of their semantics into the representation learning process will increase the model's semantic content and might thereby lead to quality improvement.

4 LEVERAGING LITERALS FOR THE TASK OF LP IN INDUCTIVE SETTING

As discussed in Chapter 3, various KGE techniques have emerged for generating KGEs that can be used to predict missing links in KGs in a transductive setting. However, most of these methods may not work well with real-world, dynamic graphs where new entities are frequently added. Therefore, several inductive LP methods have been proposed in order to fill this gap by enabling predictions about entities that are not observed during the training phase, and some of these methods make use of literals. In this chapter, the focus lies on providing reviews of these inductive LP approaches.

The rest of this chapter is organized as follows. In Section 4.1, the motivation for generating inductive LP methods is discussed, followed by Section 4.2 where the details of the existing methods are provided. A discussion of the existing datasets that have been used for the evaluation of inductive LP methods is provided in Section 4.3. Finally, a concise remark summarizing the key points is presented in Section 4.4.

4.1 INTRODUCTION

Adapting most of the existing transductive LP models such as RotatE [140], Distmult [50], ComplEx [69], and TransE [49] for inductive settings requires expensive re-training in order to learn embeddings for unseen entities. Therefore, such models are not applicable to making predictions with unseen entities. There are some models among those presented in Chapter 3 for transductive LP, such as DKRL, which could also be applied to make predictions with entities unseen during training. However, since those models are not created with a special focus on inductive LP, they do not perform well when used in an inductive setting. For instance, DKRL utilizes a CNN-based entity description encoder with a LP objective. It falls short in terms of performance since it does not consider stop words, resulting in the loss of some of the semantics present in the entity descriptions. Additionally, its CNN architecture is not up-to-date with the latest advancements in neural networks, such as self-attention.

This led to the creation of some inductive LP methods which give more emphasis on inductive LP such as BLP [18]. The focus of this chapter lies in providing a literature review on those models. These inductive LP models operate either in *dynamic evaluation* where in a test triple, a new entity may appear at the head, tail, or both positions or *transfer evaluation* where in a test triple, both entities at the head and tail position are new [18]. In both these settings, relations are usually assumed to be known during training. In the subsequent section, the SoTA inductive LP models with and without literals are discussed

4.2 THE SOTA METHODS

Most of the SoTA LP models proposed for inductive settings can be broadly divided into rule-based and embedding-based categories.

4.2.1 *Rule-based methods*

Rule-based approaches learn logical formulas or patterns from KGs, which are the explicit representation of statistical regularities and dependencies encoded in them [52]. The learned rules are utilized to rank candidates and predict missing links, based on the confidence of the rules that are triggered. AMIE [141], RuleN [52], NeuralLP [142], and DRUM [143] are the widely known rule-based inductive LP methods. Note that, as shown in Table 4.2.1, none of these models consider using literals.

AMIE AMIE derives logical rules from knowledge bases given support and confidence thresholds. It can generate rules even in the absence of explicit counter-examples by employing the Partial Completeness Assumption (PCA) technique. PCA enables AMIE to estimate the quality of rules under the Open World Assumption (OWA) by guessing counter-examples for rules.

RULEN Similar to AMIE, RuleN also aims to learn rules but with a different language bias. While AMIE calculates confidence using the entire KG, RuleN approximates confidence by randomly selecting a sample. AMIE is generally considered complete with precise confidence, whereas RuleN does not exhibit the same level of completeness or precision. However, the sampling mechanism in RuleN allows for the possibility of discovering longer path rules.

NEURALLP Despite the fact that rule-based approaches such as AIME and RuleN are inherently applicable to inductive settings, they are prone to limited expressiveness and scalability issues. In order to address this issue, NeuralLP is proposed and it works by learning first-order logical rules in an end-to-end differentiable model.

DRUM DRUM is an end-to-end differentiable rule mining system for mining first-order logical rules from KGs and provides improvement over NeuralLP. It adopts bidirectional RNNs to learn rule structures and scores simultaneously, where the learned rules are used to predict missing links. Similar to NeuralLP, applying the differentiable mining enables DRUM to learn (path-like) rules and their scores in a more continuous fashion.

4.2.2 *Embedding-based methods*

Embedding-based methods perform LP by learning embeddings for entities. GraphSAGE [144], Graph2Gauss [145], GNNs for OOKB (GNN for out-of-knowledge-base) [146], LAN [147], GraIL [148], KEPLER [19], BLP [18], and QBLP [149] are models proposed for inductive LP. As presented in Table 4.2.1, among these models, only some of them make use of literals.

GRAPHSAGE GraphSAGE is an inductive framework that aims to generate embeddings by sampling and aggregating features from a node's local neighborhood. It encodes node features with two graph convolutional layers and samples a fixed-size set of neighbors to aggregate information instead of the full neighbor set. The framework utilizes three aggregators: mean, LSTM, and pooling. GraphSAGE with a mean aggregator can be considered as an inductive version of GCN. However, as explained in [150], GraphSAGE is unable to differentiate between various graph structures. Moreover, as discussed in [18], approaches like GraphSAGE require predefining a set of attributes (e.g., bag-of-words) before training to learn entity representations, which restricts their application in downstream tasks.

GRAPH2GAUSS Graph2Gauss is a technique that examines the 2-hop and 3-hop node neighborhood sampling strategies for binary graph embedding. It leverages node feature information (e.g., textual descriptions) to generate node embeddings for previously unseen entities.

GNNS FOR OOKB GNNs for OOKB (GNN for out-of-knowledge-base) approach which generates entity embeddings for unseen entities by aggregating neighbor entity embeddings through GNNs. The drawbacks of this approach lie in the fact that it requires the new nodes (i.e., the unseen entities) to be surrounded by known nodes and fail to handle entirely new graphs as discussed in [148].

LAN LAN is another approach that uses GNN to aggregate neighborhood information to perform inductive link prediction. Similar to *GNNs for OOKB*, it fails to make predictions with those entities which are not surrounded by observed/known entities.

GRAIL GraIL [148] is a method that reasons over local subgraph structures to predict missing links in KGs. However, GraIL fails to model semantic correlations between relations, which are common in KGs.

KEPLER KEPLER is a unified model for Knowledge Embedding (KE) and pre-trained language representation by encoding textual entity descriptions with a pre-trained LM as their embeddings, and then jointly optimizing the KE and LM objectives. However, due to the additional language modeling objective, KEPLER is quite expensive to compute and requires more training data.

BLP BLP utilizes a pretrained LM for learning representations of entities via an LP objective which is inspired by the work DKRL [60]. Specifically, the pre-trained BERT model is fine-tuned to encode entities for the inductive LP task. It aims to demonstrate the power of LMs in facilitating the generalizability of entity embeddings on downstream tasks.

QBLP QBLP is a model proposed to extend BLP for hyper-relational KGs by exploiting the semantics present in qualifiers. It adopts an inductive approach, where the inference can be performed using a graph that comprises both seen and unseen entities (i.e., semi-inductive or dynamic evaluation setting) or exclusively unseen entities (i.e., fully inductive or transfer evaluation setting).

4.2.3 *Other Approaches*

KG-BERT [151] utilizes the BERT LM model as an encoder for entities and relations for the task of triple classification. This method takes textual descriptions of entities and relations in triples as input sentences to the BERT model. However, this approach neglects the structural information of the entities. Given a KG with millions of entities, completing every triple using KG-BERT would require millions of inference steps through the MLM model (as explained in [152]). In order to address this problem with the inference time, a method named MLMLM [152] has been proposed which uses a mean likelihood method to compare the likelihood of different text of different token lengths sampled from a Masked LM to perform LP. Unlike embedding-based methods, this approach does not employ an entity embedding step instead, it lets the model directly output the predicted entity. Even though MLMLM could predict missing links with unseen entities, it learns embeddings neither for entities nor for relations.

4.3 BENCHMARKS IN INDUCTIVE LP SETTINGS

The most common datasets that have been used in existing works for the evaluation of inductive LP are Wikidata5M [19], FB15k-237, and WN18RR. Additionally, WD20K [149] is introduced in QBLP [149] for the evaluation of inductive models on hyper-relational KG. The datasets that are used in this thesis, those created for evaluation on triple-based KG instead of hyper-relational KGs, are discussed in more detail as follows.

WIKIDATA5M Wikidata5M is a large-scale KG dataset with triples extracted from the Wikidata KG with aligned entity descriptions from Wikipedia pages to facilitate LP with unseen entities. Its inductive version comprises 4,594,458 entities, 822 relations, and 20,510,107 triples in total. Wikidata5M has been used to evaluate the performance of different inductive LP models such as KEPLER, MLMLM, and BLP. The dataset contains entities that are not part of the training set in the test set but all of the relations in the test set also appear in the training set.

Table 4.2.1: Inductive LP models. The symbol (✓) denotes that the model uses textual literals, while the symbol (×) represents the opposite.

Models	Incorporating literals (i.e., textual literals)
AMIE	×
RULEN	×
NEURALLP	×
DRUM	×
GRAPHSAGE	✓
GRAPH2GAUS	✓
GNNS FOR OOKB	×
LAN	×
GRAIL	×
KEPLER	✓
BLP	✓
QBLP	✓
KG-BERT	✓
MLMLM	✓

FB15K-237 FB15k-237 is initially created for the evaluation of transductive LP with 14541 entities, 237 relations, and 310,116 triples in total. However, it has been used by splitting it into training, validation, and test sets considering inductive LP in a semi-inductive setting. For instance, in BLP 10%, 10%, and 80% of entities and their associated triples are selected to form test validation, and training graphs, respectively.

WN18RR Similar to FB15k-237, WN18RR is first created for evaluation in a transductive setting. However, it has been adopted to evaluate inductive models such as BLP and MLMLM. FOr instance, in BLP 10%, 10%, and 80% of entities and their associated triples are selected to form test validation, and training graphs, respectively. This dataset comprises 40,943 entities, 11 relations, and 93,053 triples in total.

4.4 DISCUSSION AND OUTLOOK

In this chapter, discussions on inductive LP approaches are provided by grouping most of them into rule-based and embedding-based categories. One of the benefits of using an embedding-based LP method over rule-based approaches is that the embeddings of entities and relations learned in the LP task could also be leveraged in other downstream tasks. Furthermore, literals are not considered by any of the rule-based approaches. On the contrary, some of the embedding-based approaches take into account the use of textual literals, while other types of literals are mostly disregarded. Additionally, in this chapter, a review of the existing benchmark datasets for inductive LP is provided. Most of these datasets, such as FB15K-237 and WN18RR, were originally created for transductive LP and hence, they had to be adjusted (i.e., their training, validation, and test sets have to be modified) in order to make them suitable for the evaluation of inductive LP methods.

The major drawbacks of the existing inductive LP approaches and the benchmark datasets are given as follows, along with the efforts made in this thesis to address them.

- None of the LP methods consider learning embeddings for unseen relations. To fill this gap, in this thesis, an inductive LP model *RAILD* is introduced in Chapter 5. *RAILD* aims to learn embedding for both unseen entities and unseen relations.
- The current benchmark datasets available for the evaluation of inductive LP models do not contain unseen relations in their test sets and hence, they do not support evaluation with relations that are not already observed during training. Therefore, in order to address this gap, in this thesis a dataset named *Wikidata68K* is proposed in Chapter 5 to evaluate inductive LP models with unseen relations as well as unseen entities.

Part III

KGE WITH LITERALS IN INDUCTIVE SETTING

5 RELATION AWARE INDUCTIVE LINK PREDICTION

Many KGE techniques are based on optimizing LP objectives [8, 153], resulting in embeddings that model relations in a vector space. However, some methods are limited to making predictions involving only entities observed during training, which makes them unsuitable to be applied in the real world for dynamic graphs where new entities are constantly added. In order to address this gap, as discussed in Chapter 1 and Chapter 4, the concept of inductive LP that involves making predictions with entities that are not seen during training has been introduced. It has been demonstrated that textual literals (entity descriptions) can play a vital role in providing useful semantics to enable learning embeddings for unseen entities in inductive LP [18]. However, as presented in detail in the literature review in Chapter 4, the existing inductive LP methods fail in dealing with unseen relations; the relations are either assumed to be present in the training set or the methods do not learn embeddings for the unseen relations. Moreover, there is a lack of benchmark datasets to perform an evaluation of an inductive LP approach with unseen relations. Therefore, this chapter bridges the research gap by proposing a novel KGE model along with a new benchmark dataset [154] for inductive LP.

The rest of the chapter is organized as follows. To begin with, the motivation behind the work is provided in Section 5.1, followed by the problem formulation in Section 5.2. A detailed discussion of the proposed model is given in Section 5.3 followed by the findings of the experiments on the LP task in Section 5.4. Finally, concluding remarks with directions for future work are given in Section 5.5.

5.1 INTRODUCTION

Recently, KGs have gained massive attention for use in various applications such as question answering, information retrieval, recommender systems, etc [8]. As discussed in Chapter 1, although KGs are effective in representing structured data, the underlying symbolic nature of such triples usually makes KGs hard to manipulate. Moreover, due to the open-world assumption, KGs are never complete [155] and hence, there arises the need for automated KGC systems. In order to tackle this problem, various KGE methods which map entities and relations into a low-dimensional vector space by leveraging the LP objective function have been proposed (ref. to the reviews in Chapter 3 and Chapter 4). According to these reviews, most of the well-known LP approaches are either introduced only for transductive settings, such as TransE[49], DistMult [50], ComplEx [69], and LiteralE [73] , or are inductive models but without paying attention to unseen relations like BLP [18].

In contrast to entities for which usually their corresponding descriptions are used as features, relations are typically randomly initialized, as demonstrated in BLP [18]. While using the textual descriptions of relations is the most straightforward way to obtain their features, this approach is problematic when the descriptions are either too short or entirely unavailable. Therefore, in such cases, it is necessary to generate features for relations using the structural information available. This indicates that there is a need for a method that generates features for relations so that inductive LP could be performed with previously unseen relations while learning embeddings for the relations.

To this end, in this work, a novel approach *Relation Aware Inductive Link preDiction (RAILD)* is introduced. RAILD is designed to predict missing links in KGs by taking into account both unseen entities and unseen relationships. To the best of our knowledge, *RAILD* is the first approach that handles unseen relations, i.e., learns embeddings for unseen relations. It works by fine-tuning a pre-trained Language Model (LM) to encode textual descriptions of entities and relations. Moreover, it generates a graph-based relation features by first applying a novel algorithm named *Weighted and Directed Network of Relations (WeiDNeR)* to build a directed relation-relation network from the triples available in the KG and then, generating embeddings for the relations in the network using Node2Vec model which leverages contextual information based on random walks. Then, these embeddings are in turn used as features for the relations for the LP task. Moreover, RAILD also utilizes the textual descriptions of the relations as features, by either combining them with the features generated by the feature generator component or separately.

Figure 5.1.1 provides an example of inductive LP settings followed in this work. Generally, inductive LP is divided into two categories: i) semi-inductive and ii) fully-inductive. In the semi-inductive setting, it is possible for unseen entities to appear at the head, tail, or both positions whereas, in the fully-inductive setting, both head and tail entities are unseen. In this definition, relations are often overlooked, i.e., they are usually assumed to be seen in the training set and hence are randomly initialized or as in MLMLM [152], they could be encoded using their labels (or corresponding text descriptions) but without learning representations (embeddings) for them. Hence, this work divides the settings into three categories for clarity, i.e., semi-inductive (with seen relations), fully-inductive (with seen relations), and truly-inductive (with unseen entities and unseen relations).

The effectiveness of the model is evaluated against different the SoTA models in all inductive settings, including semi-inductive, fully-inductive, and truly-inductive, using benchmark datasets FB15K-237, WN18RR, and a newly created dataset named Wikidata68K. The results show that RAILD outperforms the existing models. The following are the main contributions of this chapter.

- A novel algorithm is introduced to build a *relation-relation network*, i.e., **WeiDNeR**, for the purpose of generating features for relations in a KG solely from the contextual information present in the graph structure.

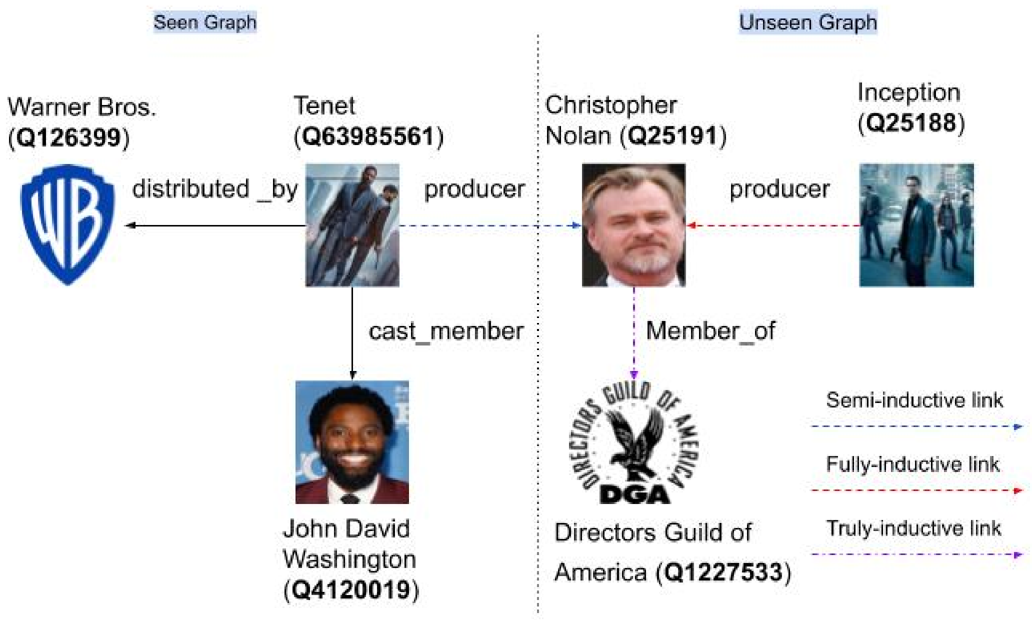


Figure 5.1.1: An example illustrating different settings of inductive LP tasks, i.e., semi-inductive (the link from *Tenet* to *Christopher Nolan*), fully-inductive (the link from *Inception* to *Christopher Nolan*), and truly-inductive (the link from *Christopher Nolan* to *Directors Guild of America*) settings

- The results indicate that instead of randomly initializing relations in inductive LP, encoding the relations like the way entities are encoded by utilizing proper features leads to outperforming the SoTA inductive LP models on triple-based KGs.
- An experiment-supported evidence is provided showing that the algorithm introduced to generate features, WeiDNeR, enables producing competitive results as compared to using textual descriptions of relations as features.
- As part of the work, a novel LP dataset named **Wikidata68K**[1] which contains unseen relations in the validation and test sets is introduced along with an automated pipeline to generate such datasets. Creating this dataset was required as there exists no such kind of evaluation dataset due to the fact that, to the best of our knowledge, no existing LP work deals with unseen relations. The results obtained with the proposed model on this challenging dataset are provided which could be seen as a first attempt to facilitate further research in the community on the topic of LP with unseen relations.

1 https://doi.org/10.5281/zenodo.7066504

5.2 PROBLEM FORMULATION

As mentioned above, existing inductive LP models operate either in a *semi-inductive* or *fully-inductive* setting. In both settings, relations are usually assumed to be known during training. For the sake of clarity, in this work, predicting with unseen relations is defined separately named as *truly-inductive* LP setting. Following the definition of KG provided in Chapter 2, a KG G consists of a set of triples T, $T \subset E \times R \times (E \cup L)$, where E, R, L, are the set of entities, relations between the entities, and literals respectively. Given T_{tr}, T_{va}, and T_{te} as sets of training, validation, and test triples where E_{tr} & R_{tr}, E_{va} & R_{va}, and E_{te} & R_{te} are their corresponding set of entities and relations respectively, the three inductive LP settings can be formally defined as follows:

- **Semi-inductive setting** For every triple $< h, r, t > \in T_{va}$ or $< h, r, t > \in T_{te}$, either or both of $h \notin E_{tr}$ & $t \notin T_{tr}$ may hold true while $R_{va} \subseteq R_{tr}$ and $R_{te} \subseteq R_{tr}$.
- **Fully-inductive setting** For every triple $< h, r, t > \in T_{va}$ or $< h, r, t > \in T_{te}$, both $h \notin E_{tr}$ & $t \notin E_{tr}$ holds true while $R_{va} \subseteq R_{tr}$ and $R_{te} \subseteq R_{tr}$.
- **Truly-inductive setting** For every triple $< h, r, t > \in T_{va}$ or $< h, r, t > \in T_{te}$, either or both of $h \notin E_{tr}$ & $t \notin E_{tr}$ holds true while there exist a set $R_v \subseteq R_{va}$ and a set $R_t \subseteq R_{te}$ where $R_v \nsubseteq R_{tr}$ and $R_t \nsubseteq R_{tr}$.

In this work, the focus lies on predicting missing links between entities in the three inductive LP settings defined above. Hence, the challenges in reference to C1-RQ1 and C1-RQ2 presented in Section 1.2 of Chapter 1 are addressed in this chapter.

- **C1-RQ1:** *Can utilizing both graph-based features and description-based embeddings for relations improve SOTA inductive LP models? Moreover, can this approach enable inductive LP with unseen relations?*
- **C1-RQ2:** *Could the existing inductive LP benchmark datasets with textual literals be extended to perform inductive LP evaluation with unseen relations?*

5.3 RAILD: RELATION AWARE INDUCTIVE LINK PREDICTION

As mentioned before, RAILD fine-tunes BERT pre-trained model to encode entities with a LP task. The general architecture of the proposed approach is given in Figure 5.3.1. Differently from BLP where relations are randomly initialized, in RAILD the same pre-trained BERT model is also applied to encode relations using their corresponding textual descriptions, as shown in Figure 5.3.1 component $\textcircled{2}$. In addition to encoding relations using BERT, a feature generator component that is based solely on graph structure is also proposed. Hence, two kinds of vectors can be generated as features for relations, i.e., text-based

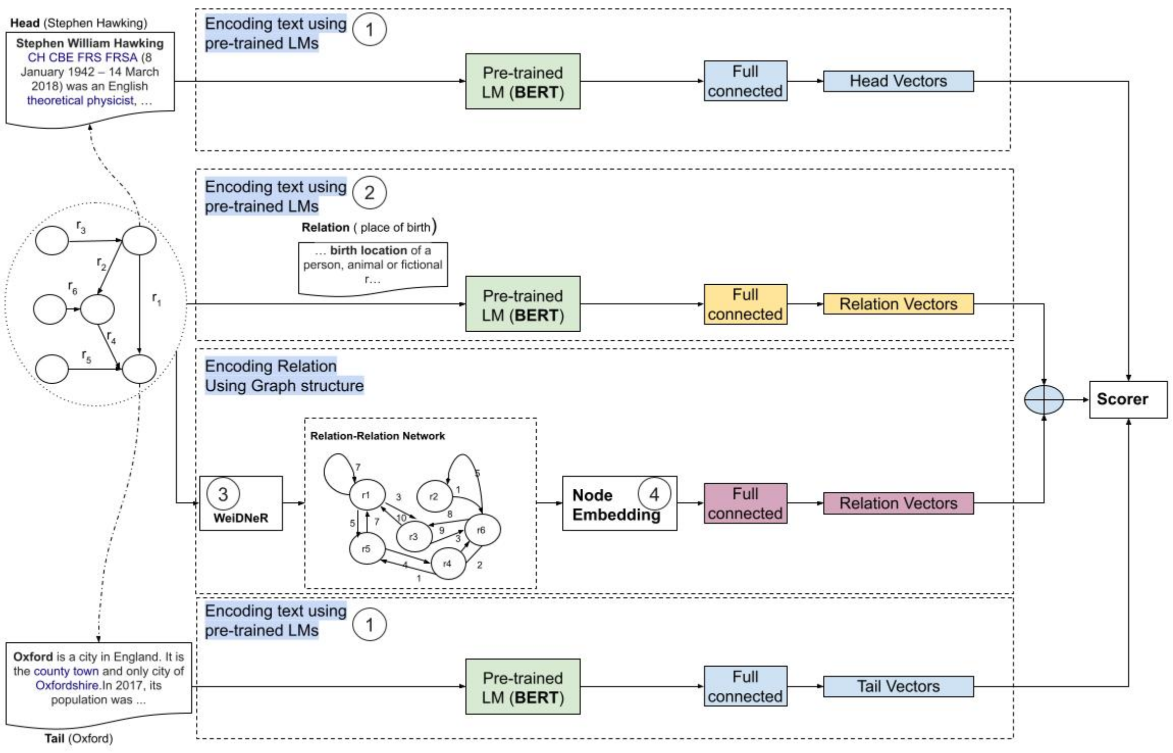

Head (Stephen Hawking)
Stephen William Hawking CH CBE FRS FRSA (8 January 1942 – 14 March 2018) was an English theoretical physicist, ...
Encoding text using pre-trained LMs 1
Pre-trained LM (BERT)
Full connected
Head Vectors
Encoding text using pre-trained LMs 2
Relation (place of birth)
... birth location of a person, animal or fictional r...
Pre-trained LM (BERT)
Full connected
Relation Vectors
r1 r2 r3 r4 r5 r6
Encoding Relation Using Graph structure
Relation-Relation Network
3 WeiDNeR
Node 4 Embedding
Full connected
Relation Vectors
Scorer
Encoding text using pre-trained LMs 1
Oxford is a city in England. It is the county town and only city of Oxfordshire.In 2017, its population was ...
Tail (Oxford)
Pre-trained LM (BERT)
Full connected
Tail Vectors

Figure 5.3.1: RAILD framework

and graph-based. Given a triple $< h, r, t >$, the two feature vectors generated for the relation r are concatenated into a single vector. Since concatenation of the two relation vectors leads to doubling the output vector dimension, the vectors of the head/tail entities are also duplicated (concatenating the head vector with itself to match the size of the concatenated relation vector). Then, the resulting head $\mathbf{h}$, tail $\mathbf{t}$ and relation $\mathbf{r}$ vectors are passed to the LP scoring function.

Textual description encoding is performed by passing the text as input to a pre-retrained LM (specifically BERT but any other transformer based LM model could be used as well) and then passing the obtained vector from BERT through a feed-forward layer, as shown in Figure 5.3.1 component ① and ②. For the graph-based feature generation for relations, two major steps are applied, i.e., building a relation-relation network (Figure 5.3.1 component ③) and generating node embeddings for the created network where the nodes are relations (Figure 5.3.1 component ④). In the subsequent sections, the different components of the proposed model are presented in detail. First, encoding textual descriptions using pre-trained BERT is discussed followed by the description of the WeiDNeR algorithm. Then, the node embedding model applied in this work is presented. Finally, the chosen scoring functions are analyzed.

5.3.1 *Encoding Textual Descriptions using BERT*

Textual descriptions of entities contain information that would provide useful semantics while learning KG representations. In order to make use of such text data for representation learning, both static embedding models such as SkipGram [46] and contextual embedding models like BERT have been extensively applied with different machine learning and natural language processing tasks. The power of transformer networks [156] in encoding text to contextualized vectors has been well received. In particular, pre-trained embedding models such as BERT provide an advantage to fine-tune the model on other downstream tasks. In BLP, pre-trained BERT is fine-tuned on the inductive LP task and it showed promising results as compared to other methods. RAILD differs from BLP in two significant aspects. Firstly, unlike BLP which only employs the encoder for entities, RAILD utilizes it to encode both entities and relations. Secondly, in RAILD, the fine-tuning process is enhanced by incorporating an additional component that integrates structure-based features specifically for relations as shown in Figure 5.3.1.

Let $d = (w_1, ..., w_k)$ be an entity or relation description, as described in detail in Chapter 2, the BERT tokenizer first adds two special tokens [CLS] and [SEP] to the beginning and end of d, respectively $([CLS], w_1, ..., w_k, [SEP])$. BERT takes this as an input leading to a sequence of k + 2 contextualized embeddings as an output, i.e., $\mathrm{BERT}(D) = [h_{CLS}, h_1, ..., w_k, h_{SEP}]$. As in BLP and many other works which employ BERT for text encoding, this work also utilizes the contextualized vector $h_{CLS} \in \mathbb{R}$ where h is the hidden size in the BERT architecture. Once h_{CLS} is obtained, it will be given as an input to a linear layer that reduces the dimension of the representation, to yield the output entity or relation embedding $h = Wh_{CLS}$, where $w \in \mathbb{R}^{d \times h}$ is the weight with d being the chosen embedding dimension. Note that, as shown in Figure 5.3.1, the weights are shared with the linear layer that is applied to the relation embeddings obtained with the Node2Vec model.

5.3.2 *Weighted and Directed Network of Relations (WeiDNeR)*

Following the KG definition provided in Section 2.2 of Chapter 2, let $G = (R, E, T \subseteq E \times R \times E)$ be a KG with $r_1, r_2 \in R$, $T_1 \subseteq (T \cap (E \times r_1 \times E))$, and $T_2 \subseteq (T \cap (E \times r_2 \times E))$. WeiDNeR's design operates on the assumption that the higher the number of common entities appearing in the sets of triples T_1 and T_2, the higher the probability that the relations r_1 and r_2 could be semantically similar. Hence, based on this assumption, an algorithm is proposed which generates a directed and weighted network graph $N_{rel} = (V, M \subseteq V \times V, w)$ where the nodes V are relations in the input KG (i.e., $V \subseteq R$), M is the set of edges connecting the nodes, and $w : M \mapsto \mathbb{R}$ assigns weight to each edge. Algorithm 1, step by step, explains the process of creating the network graph. If there is a direct link between two nodes $\textcircled{r_1}$ and $\textcircled{r_2}$ in N_{rel} network generated using this algorithm, then the following statement holds true. $|\mathtt{head}(T_1) \cap \mathtt{head}(T_2)| > 0$ OR $|\mathtt{tail}(T_1) \cap \mathtt{tail}(T_2)| > 0$ OR $|\mathtt{tail}(T_1) \cap \mathtt{head}(T_2) > 0|$ where $\mathtt{head}(T_i)$ and $\mathtt{tail}(T_i)$ are the sets of entities occurring at

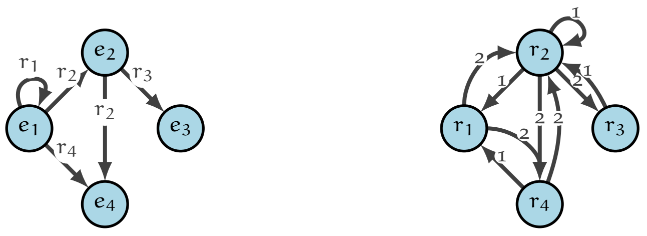


Figure 5.3.2: An example to show how Algorithm 1 works; taking the graph in the left, it produces the graph in the right.

the head and tail positions in the set of triples T_i respectively. If there is no direct link, then the statement becomes false, i.e., the two relations are not associated with any common entity in the input KG. A step-by-step description of the algorithm is given below along with an illustrative example provided in Figure 5.3.2.

Algorithm description. Taking a KG $G = (R, E, T \subseteq E \times R \times E)$ with $\{< h_i, r_j, t_k > \mid < h_i, r_j, t_k > \in T\}$ where $h_i, t_k \in E$ and $r_j \in R$ as an input and generates a relation-relation network N_{rel}. For each pair of distinct relations (r_a, $r_b \in R$) it performs the following steps:

- it counts the number of pair of triples where the relation in the first triple is r_a and in the second is r_b and the tail entity in the first triple is the same as the head entity in the second triple (i.e., refer to line 3).
- it counts the number of pairs of triples where the relation in the first triple is r_a and in the second is r_b and the head entity in the first triple is the same as the tail entity in the second triple (i.e., refer to line 4).
- it computes the number of entities shared by the triples associated with r_a and r_b at the exact same position at the head or at the tail, (i.e., refer to line 5).
- If $\#\mathtt{direct} + \#\mathtt{indirect} > 0$, then an edge from node r_a to node r_b will be created with the summed result given as a weight for the edge (i.e., refer to lines 6 to 10).

On the other hand, if r_a and r_b are the same, then the following will be performed.

- it counts the number of pairs of triples where the relations in both triples is r_a and the tail entity in the first triple is the same as the head entity in the second triple (i.e., refer to line 12)
- it computes the number of entities shared by the triples associated with r_a at the exact same position at the head or at the tail. (refer to line 13)
- If $\#\mathtt{direct} + \#\mathtt{indirect} > 0$, then an edge from node r_a to node r_b will be created with the summed result given as a weight for the edge (refer to lines 14 to 16).

Algorithmus 1 : WeiDNeR - An algorithm to generate a directed and weighted relation-relation network

```
Data : $T \leftarrow$ Triples in KG
Result : $N_{rel}$
1  for each pair of relations $< r_a, r_b >$ do
2      if $r_a \neq r_b$ then
3          $\#direct_{<r_a,r_b>} \leftarrow \big|\{(\langle h_1, r_a, t_1\rangle, \langle h_2, r_b, t_2\rangle) : \langle h_1, r_a, t_1\rangle \in T, \langle h_2, r_b, t_2\rangle \in T, t_1 = h_2\}\big|$;
4          $\#direct_{\langle r_b,r_a\rangle} \leftarrow \big|\{(\langle h_1, r_a, t_1\rangle, \langle h_2, r_b, t_2\rangle) : \langle h_1, r_a, t_1\rangle \in T, \langle h_2, r_b, t_2\rangle \in T, h_1 = t_2\}\big|$;
5          $\#indirect \leftarrow \big|\{(\langle h_1, r_a, t_1\rangle, \langle h_2, r_b, t_2\rangle) : \langle h_1, r_a, t_1\rangle \in T, \langle h_2, r_b, t_2\rangle \in T, (h_1 = h_2 \vee t_1 = t_2)\}\big|$;
6          $Weight_{\langle r_a,r_b\rangle} = \#direct_{\langle r_a,r_b\rangle} + \#indirect$;
           $Weight_{\langle r_b,r_a\rangle} = \#direct_{\langle r_b,r_a\rangle} + \#indirect$;
7          if $Weight_{\langle r_a,r_b\rangle} > 0$ then
8              $N_{rel} \leftarrow N_{rel} \bigcup \{\langle r_a, r_b, Weight_{\langle r_a,r_b\rangle}\rangle\}$;
9          if $Weight_{\langle r_b,r_a\rangle} > 0$ then
10             $N_{rel} \leftarrow N_{rel} \bigcup \{\langle r_b, r_a, Weight_{\langle r_a,r_b\rangle}\rangle\}$;
11     else
12         $\#direct \leftarrow \big|\{(\langle h_1, r_a, t_1\rangle, \langle h_2, r_b, t_2\rangle) : \langle h_1, r_a, t_1\rangle \in T, \langle h_2, r_b, t_2\rangle \in T, t_1 = h_2, (h_1 \neq t_1 \vee h_1 \neq t_2)\}\big|$;
13         $\#indirect \leftarrow \big|\{(\langle h_1, r_a, t_1\rangle, \langle h_2, r_b, t_2\rangle) : \langle h_1, r_a, t_1\rangle \in T, \langle h_2, r_b, t_2\rangle \in T, ((h_1 = h_2 \wedge t_1 \neq t_2) \vee (t_1 = t_2 \wedge h_1 \neq h_2))\}\big|$;
14         $Weight_{\langle r_a,r_a\rangle} = \#direct + \#indirect$;
15         if $Weight_{\langle r_a,r_a\rangle} > 0$ then
16             $N_{rel} \leftarrow N_{rel} \bigcup \{r_a, r_a, Weight_{\langle r_a,r_a\rangle}\}$;
```

5.3.3 *Node Embeddings*

Features for the nodes in a given network N_{rel} can be generated leveraging the network's structural information. In order to generate these features, it is possible to apply a node embedding model and hence, in this work, Node2Vec [20] is used. As discussed in detail in the *Foundaiton* part of this thesis, i.e., Section 2.5 of Chapter 2, Node2Vec is based on the skip-gram model, which is a neural network architecture for generating word embeddings. Specifically, it uses biased second-order random walks to explore node neighborhoods and SkipGram word embedding to learn embeddings. It selects the next hop using second-order transition probabilities by applying Equation 9. The SkipGram model intends to learn continuous word feature representations by optimizing a likelihood objective that preserves the local context. In the case of Node2Vec, this objective can be interpreted as maximizing the likelihood of correctly predicting the context node v for a given center node u.

5.3.4 *Training Procedure*

The graph-based features for relations in training, validation, and test sets are created separately and used as inputs for when models are trained. In a truly-inductive setting, only the triples from the training set are used to generate the graph-based features for the relations appearing in the training. Similarly, for relations in the validation and test sets, only triples from the validation and test sets are used respectively. This is performed in order to avoid using information from the unseen graphs (i.e., from validation and test sets) to learn features during training. Similarly, in both semi-inductive and fully-inductive settings, only the triples from the training set are taken as input to generate the graph-based feature for the relations.

Once the features of the entities and relations are generated or encoded, then they are used to optimize the model for LP by applying stochastic gradient descent. For each positive triple $< e_i, r_j, e_k >$, a positive score S_p is computed. Then, a corrupted negative triple is created by replacing the head or the tail entity with a random entity, and its score S_n is computed.

5.3.5 *Computational complexity*

As it is discussed in the previous sections, RAILD uses text-based and graph-based encoders. The text-based encoder is used for both entities and relations whereas the graph-based encoder is used only to encode relations. Note that the graph-based features for relations are pre-computed and hence, the major part of the computational cost of training the model comes from text-based encoder. The BERT encoder used has a complexity of $O(n^2)$ for encoding a sentence of length n. This entails that for training RAILD, the time complexity would be $O(|T|n^2)$ where T is the set of triples. The length of sentences n is in practice fixed and assuming the n is the same for all entities and relations, the complexity would remain linear with respect to the number of triples in the KG, up to a constant factor.

During testing, the text-based encoder is applied only for unseen entities and unseen relations while the embeddings for seen entities and seen relations can be pre-computed. Hence, the LP for a given entity and relation is linear in the number of entities and relations in the graph.

5.4 EXPERIMENTS

In this section, the details on experimentation including the baselines, the datasets, the experimentation settings, and the results are discussed. Our implementation and the datasets are made publicly available[2].

5.4.1 *Datasets*

The three inductive LP settings discussed in Section 5.2 are considered for experimentation. The datasets FB15K-237 [15] and WN18RR [16], with their respective splits provided in [18], are used to evaluate RAILD in the semi-inductive setting. In a fully-inductive setting, the model is evaluated on dataset Wikidata5M [19] and compared against SoTA models. The statistics of these datasets are provided in Table 5.4.1. To the best of our knowledge, there are no benchmark datasets that contain unseen relations in their validation and test sets. To address this issue and to enable the evaluation of RAILD with unseen relations, a new dataset Wikidata68K is created taking Wikidata5M as raw data. The pipeline developed to create this dataset is inspired by [17] and is constituted of the following steps.

1. Input: Raw data $\mathcal{T}$ containing triples from which the dataset will be created, and a set of pairs of relations and their types $\mathcal{RT}$. In Wikidata, there exists a metaclass (`Q107649491`: type of Wikidata property) with instances that are types of properties (i.e., relations). For example, `Q29546443` (Wikidata property for items about books) is an instance of `Q107649491` and the property `P123(publisher)` is an instance of `Q29546443`. Therefore, `(P123, Q29546443)` could be an entry in $\mathcal{RT}$.

2. Removing relations which occur in less than N number of triples ($N = 3$, for Wikidata68K).

3. Removing inverse relations, entities and relations without a label, and duplicate relations.

4. Randomly splitting the set of relations into three R_1, R_2, and R_3 while trying to keep the same type of relations in the same set based on $\mathcal{RT}$ and extract their corresponding triples $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$ from $\mathcal{T}$.

2 https://github.com/GenetAsefa/RAILD

Table 5.4.1: Dataset statistics

	WN18RR	**FB15K-237**	**Wikidata5M**	**WD20K(25)**
Relations	11	237	822	333
	Training			
Entities	32,755	11,633	4,579,609	17,275
Triples	69,585	215,082	20,496,514	38,023
	Validation			
Entities	4,094	1,454	7,374	3,092
Triples	11,381	42,164	6,699	4,072
	Test			
Entities	4,094	1,454	7,475	2,615
Triples	12,087	52,870	6,894	3,329

Wikidata68K

	Training	Validation	Test
Entities	55,488	6,559	5,813
Relations	72	37	44
Triples	667,413	67,892	45,512

5. Creating K-cores for each of $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$. (For Wikidata68K, the value of k is set to 10, 6, and 5 for $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$ respectively).

6. Removing relations which are skewed towards either the head or the tail at least 50% of the time, from each of $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$.

5.4.2 *Baselines*

For semi-inductive and fully-inductive settings, RAILD could be compared with SoTA models like BLP and KEPLER on FB15K-237, WN18RR, and Wikidata5M datasets. Since there exists no SoTA model which handles unseen relations, four different baselines Glove-BOW$_t$, Glove-DKRL$_t$, BE-BOW$_t$, and BE-DKRL$_t$ are created by extending the baselines in BLP, i.e., Glove-BOW, Glove-DKRL, BE-BOW, and BE-DKRL respectively to also encode relations using their textual descriptions in the same way they encode entities. These models are different re-implementations of DKRL [60] where Glove-DKRL uses Glove embeddings as an input to the DKRL architecture whereas Glove-BOW is the Bag-Of-Word baseline of DKRL. Furthermore, BE-BOW, and BE-DKRL are other varieties that use context-insensitive BERT

Embeddings (BE). Note that the baselines created in this work are used for evaluation in the truly-inductive setting on Wikidata68K dataset and to compare them with RAILD.

5.4.3 *Experimentation Setting*

SCORING TransE, SimplE, DistMult, and ComplEx are some of the well known translational models with `TransE` being the simplest among all. ComplEx handles antisymmetric relations better than both TransE and DistMult [69]. However, TransE could also perform well in some cases, for example, in BLP the best performing scoring function is TransE followed by ComplEx. Hence, in this work, the scoring functions `TransE` and `ComplEx` are selected.

MODEL SELECTION For the Node2Vec model, number of walks=1000, length=10, window size=10, epochs=100, dim=768 are used for FB15K-237 with semi-inductive split and Wikidata5M with fully-inductive split. For WN18RR with semi-inductive split, number of walks=5000, length=10, window size=5, epochs=100, dim=768 are used. For Wikidata68K, number of walks=10, length=10, window size=10, epochs=100, and dim=768. Similar to [18], for all newly created baselines and RAILD models, a grid search is run on FB15K-237 and the hyperparameter values with the best performance on the validation set are chosen. Then, these values are reused for training with the other datasets. For the BOW and DKRL baselines, inspired by [18], learning rate: 1e-5, 1e-4, 1e-3, L2 regularization coefficient: 0, 1e-2, 1e-3 are applied. Adam optimizer is used with no learning rate schedule, and the models are trained for 80 epochs with a batch size of 64 with WN18RR, FB15k-237, and 40 epochs with a batch size of 254 with Wikidata68K.

For the RAILD models, loss function: margin, negative log-likelihood, learning rate: 1e-5, 2e-5, 5e-5, L2 regularization coefficient: 0, 1e-2, 1e-3 are used. Adam optimizer with a learning rate decay schedule with a warm-up for 20% of the total number of iterations is used. The models are trained for 40 epochs (80 epochs for models which combine text and graph-based features for relations) with a batch size of 64 with WN18RR and FB15k-237, and 5 epochs with a batch size of 128 with Wikidata5M. In all the experiments, the negative sample size is set to 64.

5.4.4 *Results*

Two main varieties of RAILD, i.e., `RAILD-TransE` and `RAILD-ComplEx`, are created with the scoring functions TransE and ComplEx respectively. The results obtained with the different inductive LP settings are discussed in the subsequent sections.

Table 5.4.2: LP results on **semi-inductive** setting on WN18RR and FB15K-237 datasets. Models with the suffix (*) in their names are those proposed for semi-/Fully inductive LP and their results are taken from [18] whereas those with the suffix ($_t$) are our baselines. RAILD-TransE and RAILD-ComplEx are the models proposed in this work.

	FB15K-237				WN18RR			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
Glove-BOW*	0.172	0.099	0.188	0.316	0.170	0.055	0.215	0.405
Glove-DKRL*	0.112	0.062	0.111	0.211	0.115	0.031	0.141	0.282
BE-BOW*	0.173	0.103	0.184	0.316	0.180	0.045	0.244	0.450
BE-DKRL*	0.144	0.084	0.151	0.263	0.139	0.048	0.169	0.320
BLP-TransE*	0.195	0.113	0.213	0.363	0.285	0.135	0.361	0.580
BLP-DistMult*	0.146	0.076	0.156	0.286	0.248	0.135	0.288	0.481
BLP-ComplEx*	0.148	0.081	0.154	0.283	0.261	0.156	0.297	0.472
BLP-SimplE*	0.144	0.077	0.152	0.274	0.239	0.144	0.265	0.435
Glove-BOW$_t$	0.1464	0.0813	0.1636	0.2681	0.1589	0.0465	0.2085	0.3812
Glove-DKRL$_t$	0.1131	0.0678	0.1176	0.1990	0.1111	0.0283	0.1362	0.2749
BE-BOW$_t$	0.1569	0.0857	0.1780	0.2923	0.1810	0.0424	0.2483	0.4529
BE-DKRL$_t$	0.1385	0.0817	0.1473	0.2477	0.1342	0.0461	0.1636	0.3090
RAILD-TransE	**0.2163**	**0.1268**	**0.2411**	**0.3974**	0.2909	0.1360	0.3689	0.5997
RAILD-ComplEx	0.1971	0.1169	0.2121	0.3639	**0.3204**	**0.1772**	**0.3895**	**0.6087**

5.4.4.1 *Results in semi-inductive setting*

The results obtained with the semi-inductive setting on WN18RR and FB15K-237 are shown in Table 5.4.2. The table compares 2 different varieties of RAILD (i.e., RAILD-TransE and RAILD-ComplEx) with the different models from [18] and our baselines. These results show that `RAILD-TransE` outperforms all the other models on FB15K-237 w.r.t. all metrics. On the contrary, on WN18RR RAILD-ComplEx provides the best result w.r.t. all metrics whereas the second best results are obtained with RAILD-TransE w.r.t. all metrics except Hits@1. Although TransE is a less elaborate model than ComplEx, it provides better results when used with RAILD on FB15K-237 and competitive results on WN18RR. Same is the case with the results obtained in the truly-inductive setting on Wikidata68K (see Section 5.4.4.3). This suggests that the expressiveness of TransE could be highly improved with RAILD which has a more expressive encoder.

5.4.4.2 *Results in fully-inductive setting*

Table 5.4.4 shows the results obtained with the fully-inductive setting on the dataset Wikidata5M. Due to limited computational resources, for the experiment on Wikidata5M the distilled version of Bert, i.e., DistilBert [157] is used since it is cheaper to train as com-

Table 5.4.3: Ablation studies with all 4 datasets using TransE scoring function.

	FB15K-237				WN18RR			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
RAILD-TransE	**0.2163**	**0.1268**	**0.2411**	**0.3974**	**0.2909**	0.1360	**0.3689**	**0.5997**
RAILD-TransE(w/o feat)	0.2130	0.1267	0.2363	0.3872	0.2906	**0.1377**	0.3672	0.5944
RAILD-TransE(w/o txt)	0.2030	0.1168	0.2258	0.3777	0.2855	0.1312	0.3640	0.5945
	Wikidata68K				Wikidata5M			
RAILD-TransE	0.0285	0.0059	**0.0283**	**0.0688**	**0.4551**	0.2200	**0.6345**	**0.8489**
RAILD-TransE(w/o feat)	**0.0300**	**0.0077**	**0.0283**	0.0661	0.4529	0.2274	0.6190	0.8376
RAILD-TransE(w/o txt)	0.0137	0.0014	0.0130	0.0320	0.4522	**0.2304**	0.6163	0.8378

Table 5.4.4: LP results on Wikidata5M dataset using DistilBERT instead of BERT for RAILD models

	MRR	Hits@1	Hits@3	Hits@10
Glove-BOW*	0.343	0.092	0.531	0.756
Glove-DKRL*	0.362	0.082	0.586	0.798
BE-BOW*	0.282	0.077	0.403	0.660
BE-DKRL*	0.322	0.097	0.474	0.720
BLP-TransE*	**0.478**	**0.241**	**0.660**	**0.871**
KEPLER [19]	0.402	0.222	0.514	0.730
MLMLM [152]	0.284	0.226	0.285	0.348
RAILD-TransE (**DistilBERT**)	0.4551	0.2200	0.6345	0.8489

pared to Bert. However, it should be noted that since DistilBert is a slimmed-down version of BERT with fewer parameters it may lead it to be less powerful than Bert. Although MLMLM is not an embedding-based LP model, it is compared with our approach on Wikidata5M. It can be seen that even with DistilBert RAILD-TransE trained on Wikidata5M outperforms KEPLER and MLMLM w.r.t. almost all metrics.

5.4.4.3 *Results in truly-inductive setting*

Table 5.4.5 presents the results obtained on Wikidata68K (see Section 5.4.1 for details). The set of training, validation, and test relations are mutually exclusive. Moreover, 89% of validation entities and 74% of test entities are not seen during training. For datasets like Wikidata68K, it is not possible to just randomly initialize the relations (i.e., it is required for an LP model to have features for relations). Therefore, in order to assess the capability of the proposed model RAILD on such a challenging dataset (Wikidata68K), the baselines

Table 5.4.5: LP results on Wikidata68K datasets

	MRR	Hits@1	Hits@3	Hits@10
Glove-BOW$_t$	0.0119	0.0005	0.0146	0.0295
Glove-DKRL$_t$	0.0031	0.0005	0.0029	0.0064
BE-BOW$_t$	0.0184	0.0005	0.0225	0.0474
BE-DKRL$_t$	0.0055	0.0006	0.0052	0.0125
RAILD-TransE	**0.0285**	**0.0059**	**0.0283**	**0.0688**
RAILD-ComplEx	0.0157	0.0027	0.0125	0.0351

discussed in Section 5.4.2 are created. The best results on this dataset are obtained with RAILD-TransE w.r.t. all the metrics.

As compared to the other datasets, the results obtained on Wikidata68K, in general, are low. This is mostly attributed to the nature of the dataset as explained above, i.e., the relations sets in the train validation and test sets being 100% mutually exclusive. Moreover, the WeiDNeR algorithm is applied to the training set, the validation set, and the test set separately so as to avoid generating features using unseen graphs for training. As this is the first work, to the best of our knowledge, to ever make an attempt to perform LP with unseen relations, it would facilitate further research in the community to redirect the focus to unseen relations as well as entities.

5.4.4.4 *Ablation studies*

As the results given in Table 5.4.3 for the datasets FB15K-237, WN18RR, and Wikidata5M indicate, according to almost all the metrics, combining text-based and graph-based features for relations (i.e., RAILD-TransE) provides better results than using them separately. Moreover, RAILD-TransE(w/o text) model variant which uses only graph-based relation features is competitive with its counterpart text-based variant RAILD-TransE(w/o feat) and specially on Wikidata5M it even provides slightly better hits@1 and hits@10 results than RAILD-TransE(w/o feat). This indicates that the RAILD-TransE(w/o text) could be used in cases where KGs do not contain textual descriptions for their relations. Similarly, combining the two kind of features for Wikidata68K provides better results w.r.t. hits@10 and equal or competitive results w.r.t. the other metrics as compared to RAILD-TransE(w/o feat). Note that both RAILD-TransE(w/o feat) and RAILD-TransE(w/o txt) outperform the BLP models which share the same scoring function. For instance, RAILD-TransE(w/o txt) which uses only graph-based features for relations outperforms all the baselines including BLP-TransE [18] which randomly initializes relations, on both datasets FB15K-237 and WN18RR w.r.t. almost all the metrics.

Table 5.4.6: LP results with semi-inductive setting on **WD20K(25)**. #QP denotes the number of qualifiers per statement.

	#QP	MRR	Hits@1	Hits@10
BLP-TransE*	0	0.1245	0.0598	0.2343
QBLP*	0	0.1702	0.0882	0.2950
QBLP*	2	0.2036	0.1177	0.3226
QBLP*	4	**0.2105**	**0.1232**	0.3009
QBLP*	6	0.1950	0.1114	0.3160
RAILD-TransE (w/o feat)	0	0.1586	0.0761	**0.3313**

5.4.4.5 *Additional Experiments*

In addition to the experiments discussed above further experiments are also performed to compare the performance of the proposed model with QBLP [149] which is an inductive LP model developed for hyper-relational KG. The same set of optimal hyperparameter values from FB15K-237 datasets are used. RAILD is compared with QBLP on a WD20K(25) dataset [149] for hyper-relational KG. The dataset statistics considering only the triples (removing qualifiers) are given in Table 5.4.2 and the results in Table 5.4.6. The performance of our model could be negatively impacted by the fact that the size of this dataset is very small as compared to the other datasets used in this work such as FB15K-237. Moreover, since there are relations that occur only in a few triples, generating relations features using WeiDNeR could not be applied as these relations become outliers, i.e., they could not be linked to any other relation (see Algorithm 1). Therefore, only RAILD-TransE(w/o feat) could be used. Despite the argument stated above, RAILD scores the best Hits@10 as compared to QBLP which makes use of qualifiers.

5.5 CONCLUSION AND OUTLOOK

In this work, a novel inductive LP model `RAILD` which handles unseen relations is introduced. It works by fine-tuning pre-trained LMs with a LP objective. Textual descriptions of entities and relations are used to generate features for the corresponding entities and relations. Moreover, a novel algorithm, i.e., WeiDNeR, is proposed to generate a directed and weighted relation-relation network given a KG. The experimental results show that RAILD achieves the SoTA results for the semi-inductive inductive LP task on all the datasets used while providing comparable results with the SOTA models that use BERT in a fully-inductive setting using the distilled version of BERT. Furthermore, RAILD outperforms all the baselines in the truly inductive setting. The answers to the two major research questions that are formulated in this work, in Section 5.2, are given as follows:

- **C1-RQ1:** *Can utilizing both graph-based features and description-based embeddings for relations improve SoTA inductive LP models? Moreover, can this approach enable inductive LP with unseen relations?*
  - The graph-based features of relations in a KG are obtained by generating relation-relation network and then learning embeddings for the nodes of the network (i.e., relations) with a network embedding model. The description-based embeddings of entities and relations are encoded using BERT. The results of the extensive experiments, provided in Table 5.4.2, indicate that combining graph-based and description-based embeddings of relation provides better results for inductive LP as compared to the existing models that randomly initialize relations.

    Moreover, as shown in Table 5.4.5, RAILD can also be used to learn embeddings for unseen relations and hence, enable inductive LP involving relations that are not observed during training.
- **C1-RQ2:** *Could the existing inductive LP benchmark datasets with textual literals be extended to perform inductive LP evaluation with unseen relations?*
  - A new benchmark dataset named Wikidata68K is introduced that extends the existing inductive dataset Wikidata5M to enable evaluation with unseen relations. Wikidata68K has been used to evaluate the inductive LP model that is proposed in this chapter, i.e., RAILD.

The following research directions will be considered for future work:

- Incorporating other types of literals for the relations such as numerical literals into the model.
- Leveraging rules (logical formulas) to capture more semantics and further enhance the model.
- Investigating the WeiDNeR algorithm further for the LP task where few-shot relations exist.
- Adapting the proposed approach to hyper-relational KGs.

This chapter focuses entirely on inductive LP by proposing a novel KGE model along with a new dataset for an inductive setting. It highlights the importance of literals in capturing semantics, especially when they are combined with contextual information from graphs. In contrast, the subsequent chapters of this thesis focus on literals for transductive LP, with a set of novel KGC datasets tailored for transductive LP introduced in the next chapter.

Part IV

KGE WITH LITERALS IN TRANSDUCTIVE SETTING

6 LP BENCHMARK WITH LITERALS

As discussed in Chapter 1, literals play a crucial role in providing semantics when learning KGEs. Therefore, a significant amount of work in KGEs [1] utilize the different types of literals such as text and numeric literals. In order to properly examine the performance of these KGE approaches, the datasets used to conduct experiments with these models should contain high-quality triples with text and numeric literals. However, the existing KGE benchmark datasets are not created taking into consideration such models as explained in detail in the literature review in Section 3.4 of Chapter 3. In order to fill this gap, in this chapter, a set of benchmark datasets are proposed with the purpose of facilitating research in utilizing literals for the task of KGE and KGC in general.

The rest of the chapter is organized as follows. Section 6.1 discusses the motivation behind creating a new set of KGC benchmark datasets followed by Section 6.2, where a detailed description of the procedure followed to generate the proposed benchmark datasets is presented. Section 6.3 demonstrates the comparison between existing datasets and the newly created datasets whereas Section 6.4 presents benchmarking experiments on the generated datasets using KGE models, both with and without literals. Finally, concluding remarks along with directions for future work are stated in Section 6.5.

6.1 INTRODUCTION

The performance of various KGE approaches, mainly LP models, has been evaluated using some commonly known KGC datasets. Most of these datasets except CoDEx [121], as explained in detail in the literature review in Section 3.4 of Chapter 3, are outdated and easy for LP tasks such as FB15K [49] and FB15K-237 [15] which are subsets of the no longer maintained KG Freebase [3]. Moreover, attributive triples have not been handled properly in any of the current datasets. For instance, in CoDEx-M [121], it is not possible to find a single datatype property in Wikidata with numerical literal values for some of the entities. Apart from numerical properties, the major existing datasets also contain a significant number of entities for which there is no textual description available. For instance, in CoDEx among the total number of 77,951 entities, 17,276 of them do not have textual descriptions in English, i.e., they are not represented in English Wikipedia. Hence, in those studies which combine KG and textual entity descriptions for representation learning (such as DKRL [60]) it is common to filter out these entities in order to train the embedding models. This indicates that a high-quality benchmark that covers both relational and attributive triples is required to evaluate the performance of the SoTA KGC models.

This chapter addresses the challenges in reference to the research question C2-RQ2 from Section 1.2 of Chapter 1.

- **C2-RQ2**: *How to extract high-quality benchmark datasets from popular KGs such as Wikidata, focusing primarily on literals?*

In order to tackle this question, in this work, a KGC benchmark named **LiterallyWikidata** which properly combines attributive triples with relational triples by taking into account the aforementioned concerns is presented. **LiterallyWikidata** consists of a set of KGC datasets extracted from Wikidata and Wikipedia. In addition to Github, all of the datasets are made available also on Zenodo under Creative Commons Attribution 4.0 International license to ensure long-term findability through a persistent identifier[1].

The contributions of this work are summarized as follows:

- **Datasets**: **LiterallyWikidata** which is a benchmark containing three subsets of Wikidata varying in size and structure is introduced. Each of these subsets contains both relational and attributive triples along with entity types.

- **Automatic dataset creation pipeline**: As compared to the way the current benchmarks are created, for instance, CoDEx, the pipeline used in this work requires very little human intervention. In CoDEx, the first step taken was defining a set of initial classes in some specific domains whereas in our pipeline it is not required for the domains and initial classes to be predefined. Moreover, it is possible to adapt the pipeline to create new datasets with newer Wikidata dumps.

- **Benchmarking**: Extensive KGC experiments have been conducted on **LiterallyWikidata** for selected embedding models with and without attributive triples on the task of LP.

- **Review of existing LP datasets**: A review of the existing KGC datasets in terms of their sources, domain, and support for literals has been conducted and presented in Table 3.4.1.

6.2 DATASET CREATION

In this section, the procedure followed to create the LiterallyWikidata benchmark is discussed in detail. First, attributive triples with numerical literals are extracted from the Wikidata full dump from 07 September, 2020[2]. Then, relational triples are retrieved from the dump for the entities with the attributive triples. Once the triples are extracted, duplicate triples are filtered out and different datasets varying in size and structure are generated, namely, **LitWD1K**, **LitWD19K**, and **LitWD48K**. Finally, each of the datasets is divided into

1 The details including the DOI are given under the reference [158]
2 https://dumps.wikimedia.org/wikidatawiki/

training, validation, and testing triples. Note that classes explicitly have not been considered as entities in this framework in order to enable the adaptability of the datasets for tasks other than LP such as entity type prediction. Classes in Wikidata are those items which occur either as the value/object in an instance-of (`P31`) statement/triple or they are subject or value/object in a subclass-of (`P279`) statement. In the subsequent sections, the steps taken to generate the datasets are discussed in detail, i.e., i) extracting attributive triples, ii) extracting relational triples, and iii) filtering the triples.

6.2.1 *Extracting Attributive Triples*

Note that in this phase the main focus is on extracting attributive triples with datatype properties taking numerical values. Therefore, the first step is identifying those data type properties in Wikidata. The Wikidata properties which are typed with any of the three Wikimedia datatypes *Wikimedia:Time*, *Wikimedia:GlobeCoordinate*, and *Wikimedia:Quantity* are considered, in this work, as properties taking numeric values.

wikimedia:time Those properties which take *point in time* values, such as `P569` (date of birth) are categorized as *Wikimedia:Time* properties.

wikimedia:globecoordinate The values of *Wikimedia:GlobeCoordinate* typed properties such as `P625` (coordinate location), are geographic coordinates given as latitude-longitude pairs. We have separated these pairs by attaching the postfix “longtiude” and “latitude” to the ID of the properties. For instance, the triple

`<Q100000 P625 "Point(5.7678 50.8283)"^^geo`[3]`:wktLiteral .>`

is transformed into the following two triples:

`<Q100000 P625_Longtiude "5.7678"^^xsd`[4]`:double .>` and

`<Q100000 P625_Latitude "50.8283"^^xsd:double .>`

Note that some entities have multiple values per property. For such entities, splitting their corresponding triples might create a logical problem, i.e., it would be difficult to associate longitude and latitude values once the triples are split. Therefore, only one triple per <*entity, property*> pair has been randomly selected before splitting.

wikimedia:quantity Properties of wikimedia type *Wikimedia:Quantity* take quantities representing decimal numbers, such as `P2049` (width). In the case of these properties, for every <*entity, property*> pair statements ranked as “preferred” are retrieved if there are any. Otherwise, all statements which are ranked as “Normal” are extracted. In Wikidata, such statements have units associated with their values. These units might be either SI units or non-SI units. Those values with non-SI units are normalized to their corresponding

3 http://www.opengis.net/ont/geosparql#
4 http://www.w3.org/2001/XMLSchema#

SI unit whenever possible. There are still properties with more than one unit after normalization. These units are either not normalizable or are outliers. For each statement with a non-normalizable unit, the unit is attached to the ID of the property as a postfix. For example, the property `P3362` (Operating Income) takes currencies such as `Q4916` (Euro), `Q4917` (United States Dollar), and `Q25224` (Pound sterling), as units that could not be converted to one base unit and thus, they will be combined with the property ID as in `P3362_Q4916`, P3362_Q4917, and `P3362_Q25224` respectively. For each property, units that occur less than 1% of the time are considered outliers and are removed.

Note that the extracted triples with the aforementioned data type properties do not include those entities which satisfy at least one of the following conditions:

- The entities do not have site-links at least to the English Wikipedia. This step is required in order to support those LP models which leverage textual descriptions of entities.
- The entities have types only from the set of subclasses of the class `Q17379835` (Wikimedia page outside the main knowledge tree). This is imposed in order to keep only those entities that describe real-world concepts.

6.2.2 *Extracting Relational Triples*

Those triples with properties of Wikibase type *wikibase:Item*, are referred to as relational triples in this work. Once the entities with numerical literals are obtained as discussed above in Section 6.2.1, the next step is to extract relational triples for these entities. At this phase, we address both inverse properties and symmetric properties as follows:

- **Inverse properties:** Given two inverse properties p_1 and p_2 connected with the property `P1696` (inverse property) where the frequency of p_1 is greater than or equal to that of p_2, the subject and object entities of those triples with p_2 have been swapped and p_2 is replaced with p_1.
- **Symmetric Properties:** In these relational triples, every relation, except `P1889` (different from) whose head-tail pairs overlap with its tail-head pairs at least 50% of the time is considered as symmetric and hence, for each pair of redundant triples belonging to this relation, only one of them is kept. The property `P1889` (different from) has been removed due to the fact that it occurs in a significantly high number of triples but the semantic information captured in this property is not that much beneficial for KGE approaches to learn better KG representation.

6.2.3 *Filtering the Triples*

Taking as inputs the extracted attributive and relational triples, the goal in this phase is to create three datasets that vary in structure and size to be used for different purposes. The smallest dataset could be used for debugging and testing KGE models with and without literals whereas the medium size dataset would suit for evaluating KGE approaches on multiple tasks in general. On the other hand, the largest dataset could be used for few-shot evaluations in addition to general evaluations for KGEs. In this section, these datasets are referred to as small, medium, and large. The following three steps are applied to create these datasets:

SEEDING ENTITIES. The top N entities with the highest number of datatype properties are considered as seed entities. The value of N is $200,000$ for the small and large datasets and $50,000$ for the medium datasets. Different values have been tried out for N and those particular values are chosen because they suit well to generate appropriate-sized datasets.

EXTENDING THE SEED ENTITIES. At this phase, fractions of the relational triples are taken by extending the seed entities with their **one-hop** entities for the small and large datasets and with their **two-hop** neighbors for the medium dataset.

CREATING K-CORES. The size of the triples extracted using the steps discussed so far is huge as it is from the entire Wikidata dump. Hence, the relational triples have been further filtered into $k-cores$, i.e., maximal-subgraphs G' of a given graph G where each node in the sub-graphs has at least a degree of k [159]. The value of k is 15 for the small and medium datasets and 6 for the large datasets. Note that the values for k are determined by taking into consideration both the size and structure of the datasets to be generated. The value of k is less for the largest dataset as compared to the others because this dataset is intended to be used for few-shot evaluations. In case of few-shot evaluations, it would be possible to see the advantages of literals in learning representations for entities occurring in few structured triples. Once the k-cores are created, some triples have been removed from each of the k-cores due to the following factors:

- Either the head or the tail entity doesn't have a summary section on the corresponding English Wikipedia page or the section contains less than 3 non-stop words.
- All entities having exactly the same Wikipedia pages for various reasons have been excluded in order to avoid having meaningless descriptions.
- Relations (object properties) with more than 50% subject-object overlap have been considered as duplicates and only one of them is kept.
- Relations occurring less than 3 times have been removed to ensure that every relation has a chance to appear in the training, validation, and test sets.
- Attributes (data properties) skewed 100% of the time towards a single (head) entity have been excluded.

In the subsequent sections, the created small, medium, and large datasets are referred to as LitWD1K, LitWD19K, and LitWD48K respectively. The statistics and analysis of these datasets are presented in Table 6.2.1. Each of these datasets has been split into 90/5/5 train/valid/test sets. While splitting the datasets, we have ensured that the entities which occur in validation and test sets also occur in the respective training sets. Moreover, the test sets do not contain any relation which is 100% skewed towards a single head or tail entity. LitWD48K contains more than double the number of entities in LitWD19K. However, both datasets have almost the same number of structured triples. This is due to the way the datasets are created, i.e., LitWD19K is based on two-hop whereas LitWD48K is based on one-hop as discussed above. Table 6.2.1 also presents a summary of the analysis of the datasets in terms of graph connectivity, diameter, and density.

Table 6.2.1: Dataset Statistics and Analysis

		LitWD1K	LitWD19K	LitWD48K
Statistics	#Entities	1,533	18,986	47,998
	#Relations	47	182	257
	#Attributes	81	151	291
	#StruTriples	29,017	288,933	336,745
	#AttrTriples	10,988	63,951	324,418
	#Train	26,115	260,039	303,117
	#Test	1,451	14,447	16,838
	#Valid	1,451	14,447	16,838
Analysis	Connectivity	Yes	Yes	No[a]
	Diameter	5	7	8[b]
	Density	0.01235	0.0008	0.00014

[a] LitWD48K contains 3 connected components and the largest component contains 47,994 entities.

[b] The diameter of the largest component of LitWD48K is 8.

6.2.4 *Textual Information*

In addition to the relational and attributive (numerical) triples discussed in Section 6.2.2 and Section 6.2.1, textual information about the entities and relations has also been extracted. The textual information includes **Wikidata labels**, **aliases**, and **descriptions of entities, relations, and attributes**. Moreover, for each entity, the **summary** sections of the corresponding English, German, Russian, and Chinese Wikipedia pages have been extracted. The statistics of the text literals for each dataset are given in Table 6.2.2.

Table 6.2.2: Short and long text literals extracted from Wikidata and Wikipedia for entities, relations and attributes. The values are presented in percentages.

	Wikipedia Summary				Wikidata (entity/relation/attrb) (en)		
	en	de	ru	zh	labels	aliases	descriptions
LitWD1K	100	78	72	66	100/100/100	38/83/81	95/98/100
LitWD19K	100	80	65	39	100/100/100	44/87/81	99/99/100
LitWD48K	100	88	75	29	100/100/100	47/87/79	99/99/100

6.2.5 *Domain of the Datasets*

Since the pipeline developed in this study to create LiterallyWikidata framework does not require pre-defining the domains or classes of entities or relations, the created datasets are generic and their domains could be identified only after they are created. Based on the types/classes of entities, People, Geography, Entertainment, Transportation, Sport, Travel, Business, and Research are among the domains covered in LiterallyWikidata. The classes/-types of the entities are also released along with the datasets.

6.3 COMPARISON WITH EXISTING DATASETS

LP benchmark datasets are usually characterized based on the nature of the relations such as symmetricity, inversion, skewness, and cartesian product (fixed-set). LP with symmetric/inverse/cartesian product relations is easy and does not require a complex embedding model [121, 122]. It could also be done with simple rule-based approaches. Here, the comparison will be with two existing datasets, FB15K-237 as the most popular extension of FB15K and CoDEx-M as the most recent dataset extracted from Wikidata. In order to make a fair comparison, the LitWD19K dataset is chosen to be compared against these datasets as it is comparable to both in terms of size.

SKEWNESS As reported in CoDEx [121], 15.98% and 1.26% of test triples in FB15K-237 and CoDEx-M contain relations which are skewed 50% or more toward a single head or tail entity. As mentioned above, when splitting the LiterallyWikidata datasets, any relation which is 100% skewed towards a single head or tail entity in each of the datasets was excluded. However, for a fair comparison with the numbers reported in CoDEx [121], we also consider skewed relations as relations which are skewed 50% or more (instead of 100%) towards a single head or tail entity and find 6.48% of the test sets of LitWD19K to contain such skewed relations. This number does not have much of an impact as its coverage of the test set is low and also as already mentioned, none of the relations are 100% skewed.

SYMMETRICITY 4.01% of the triples in CoDEx-M contain symmetric relations [121]. In case of FB15K-237, every validation and test triple containing entity pairs that are directly linked in the training set were removed, which leads to deleting any symmetric relations from its test/validation sets. LitWD19K does not contain any symmetric relation in the entire dataset not only test/valid sets.

INVERSION Similar to the existing datasets FB15K-237 and CoDEx-M, LitWD19K also do not contain any inverse relations (see section 6.2.2 for more details).

CARTESIAN PRODUCT OR FIXED-SET RELATIONS As reported in [121], about 12.7% of test triples in FB15K-237 contain fixed-set relations, i.e., relations which connect entities to fixed sets of values. On the other hand, both CoDEx-M and our dataset (LitWD19K) do not contain any such kind of relation.

6.4 BENCHMARKING EXPERIMENTS ON LINK PREDICTION

In this section, the benchmarking experiments conducted on the LP task are discussed. The chosen KGE approaches, the model selection strategy, and the obtained results are presented. Note that there are properties in the LiterallyWikidata datasets which take date values. In order to treat those date values as numeric literals, for the experiments, the values are converted to decimals. This allows leveraging the semantics present in all parts of the date values, i.e., the year, the month, the days, and so on.

6.4.1 *KGE Models*

In this word, the models DistMult-LiteralE, DistMult, and ComplEx have been chosen to conduct the benchmarking experiments. The model DistMult-LiteralE was selected because the main focus of this study lies in providing benchmark datasets for KGE with literals whereas the other models DistMult and ComplEx are included to show the comparisons with and without literals. **DistMult** scores a given triple using a diagonal bilinear interaction function between the head and tail entity embeddings and the relation embeddings - $f(h, t, r) = h^T diag(r)t$. This model can only deal with symmetric relations due to the fact that $f(h, t, r) = f(r, t, h)$. **ComplEx** is an extension of DistMult, which uses complex-valued embeddings in order to better handle asymmetric relations. **DistMultLiteral** extends DistMult by modifying the scoring function f such that the entity embeddings of h and t are replaced with their respective literal enriched representations h^{lit} and t^{lit}. More details on these models are provided in Chapter 3.

6.4.2 *Model Selection*

As it is demonstrated in [53], in addition to a model's architecture, the combination of the training approach and the loss function used also plays an important role to determine a model's performance. Hence, in this work, pytorch-based configurable KGE framework **Pykeen**[5] is used to search from a large range of hyperparameters listed in Table 6.4.1. First, around 70 different combinations of datasets, models, training approaches, losses, regularizers and optimizers (for example, **LitWD1K + DistMult + LCWA + CEL + LP + Adam**) were defined as configurations. Then, for each of these configurations, **random search** has been used to perform the hyper-parameter optimizations over all other hyper-parameters in order to select the best models. The details on the training approaches, losses, and search strategies are given as follows:

TRAINING APPROACHES AND LOSS FUNCTIONS The models have been trained based on the sLCWA (Stochastic Local Closed World Assumption) and LCWA (Local Closed World Assumption) approaches. The sLCWA training approach has been used with UNS (Uniform Negative Sampler) to generate negative samples. The loss functions Cross Entropy Loss (CEL) and Binary Cross Entropy Loss (BCEL) are used together with LCWA whereas BCEL and Margin Ranking Loss (MRL) are used with sLCWA. In order to learn more about these training approaches and losses refer to [53].

SEARCH STRATEGIES For each configuration with LitWD1K, a maximum of 100 trials are generated within a bound of 24 hours for DistMult and DistMultLiteral, and 36 hours for ComplEx. During each trial, the model is trained for 1000 epochs. On the other hand, for LitWD19K and LitWD48K a maximum of 100 trials are generated within 48 hours for DistMult and DistMultLiteral, and 60 hours for ComplEx. Every trial is run for a maximum of 500 epochs where early stopping is performed by evaluating the model every 25 epochs with a patience of 50 epochs on the validation set using MRR. Finally, for each dataset and embedding model pair (e.g., LitWD1K+DistMult), the best configuration is chosen based on the evaluation result on the validation set. Then, evaluation is carried out using the test set by retraining the selected models on each dataset for 1000 epochs. In order to make sure that the results reported are consistent, the retraining is done three times for all models on LitWD1K and for DistMult on LitWD19K and since the results are very close, the retraining is run only once for the rest of the experiments.

The experiments with LitWD1K and LitWD19K are run on TITAN X (Pascal) 12 GB whereas

5 https://pykeen.readthedocs.io/en/latest/

those on LitWD48K are run on NVIDIA Tesla V100S-PCIE-32GB. The optimal hyperparameter values for each of the models on all the datasets are provided along with the datasets on Github[6].

Table 6.4.1: Hyper-parameter search space

Hyper-parameter	Range
Embedding dimension	{64,128,256}
Initialization	{Xavier}
Optimizers[a]	{Adam, Adadgrad}
Regulaizer	{None, L1, L2}
Weight for L1 and L2	[0.01, 1.0)
Learning Rate (log scale)	[0.001, 0.1)
Batch size	{128, 256, 512, 1024}
Input dropout[b]	{0,0.1,0.2,0.3,0.4,0.5}
Training Approach[c]	
sLCWA	
Loss	{BCEL, MRL}
Number of Negatives	{1, 2, ... , 100}
Margin for MRL	{0.5, 1.5, ... , 9.5}
LCWA	
Loss	{BCEL, CEL}
Label Smoothing (log scale)	[0.001, 1.0)

[a] Both Adam & Adagrad are evaluated using DistMult & DistMultLiteral on LitWD1K and using DistMult on LitWD19K & LitWD48K(sLCWA). The result indicates that Adagrad performs better than Adam on the smallest dataset whereas Adam is better on the larger ones. Hence, for that reason and also due to the fact that Adam is known for addressing the problem of decreasing learning rate in Adagrad, for the two larger datasets, Adam is used for the rest of the experiments in order to reduce computational cost.

[b] The input dropout range is applied to DistMultLiteral

[c] Both sLCWA & LCWA are evaluated using DistMult & DistMultLiteral on all three datasets and learned that LCWA performs better at all times. Hence, only LCWA is used for the rest of the experiments.

6 https://github.com/GenetAsefa/LiterallyWikidata

6.4.3 *Results*

The results of the experiments on LP are presented in Table 6.4.2. Three different comparisons can be made from the results, i.e., i) Baselines vs. Models with literals, ii) between baselines, and ii) proposed datasets vs. existing datasets.

- **Baselines vs. Models with literals:** Here, DistMult is compared with DistMutLiteral because DistMutLiteral is a literal-enriched KGE that extends DistMult. As the results indicate, for all three datasets DistMultLiteral outperforms DistMult w.r.t. almost all metrics. This indicates that making use of literals (numeric literals) improves entity representations.
- **Baselines vs. Baselines:** When comparing the baselines (models that do not leverage literals), ComplEx, and DistMult, it can be seen that ComplEx performs better than DistMult on the largest dataset LitWD48K. On the other two datasets, the results of the two models are comparable.
- **Proposed datasets vs. Existing datasets:** In order to show the level of difficulty of the proposed datasets, here the results of the two baselines are compared on LitWD19K and the existing datasets FB15K-237 and CoDEx-M. For both ComplEx and DistMult, w.r.t. all metrics, the results on LitWD19K are worse than those on FB15K-237 and CoDEx-M.

Table 6.4.2: Results of LP

	Dataset	Model	MRR	Hits@1	Hits@10
Ours	LitWD1K	DistMult	0.419	0.283	0.697
		ComplEx	0.413	0.28	0.673
		DistMultLiteral	**0.431**	**0.297**	**0.703**
	LitWD19K	DistMult	0.195	0.138	0.308
		ComplEx	0.181	0.122	0.296
		DistMultLiteral	**0.245**	**0.168**	**0.399**
	LitWD48K	DistMult	0.261	0.195	0.4
		ComplEx	0.277	**0.207**	0.428
		DistMultLiteral	**0.279**	0.204	**0.434**
Existing*	FB15K-237	DistMult	0.343	0.250	0.531
		ComplEx	0.348	0.253	0.536
	CoDEx-M	ComplEx	0.337	0.262	0.476

* The results are copied from LibKGE (https://github.com/uma-pi1/kge)

6.5 CONCLUSION AND OUTLOOK

This chapter presents LiterallyWikidata which is a set of KGC benchmark datasets extracted from Wikidata and Wikipedia with a special focus on literals. It contains 3 datasets, namely, LitWD1K, LitWD19K, and LitWD48K, that vary in size and structure. Each of these datasets contains both relational and attributive triples along with entity types. Benchmarking experiments have been conducted on the task of transductive LP using KGE models with and without literals. Moreover, the datasets are compared with existing datasets both conceptually and experimentally. The answer to the major research question that is formulated in this work, in Section 7.1, is given as follows:

- **C_2-RQ_2**: *How to extract high-quality benchmark datasets from popular KGs such as Wikidata, focusing primarily on literals?*
  - Generating high-quality benchmark datasets can be achieved by following the pipeline introduced in Section 6.2, which gives proper attention to literals. The pipeline contains the process of generating attributive and relational triples with different types of literals and also applies various filtering mechanisms to produce a very sound set of datasets. The existing datasets FB15K-237 (popular) and CoDEx (recent) are both valuable datasets for LP with KGC models that do not make use of literals. However, as shown in Table 6.4.2, the LiterallyWikidata benchmark datasets created with the proposed pipeline are appropriate for the evaluation of KGC models with or without literals on the transductive LP tasks. Hence, the release of LiterallyWikidata addresses the need for high-quality benchmark datasets and may foster research on more sophisticated KGE models that exploit the additional semantics provided by literals.

In future work, the following research directions will be considered to further investigate LiterallyWikidata.

- **More tasks**: Using the datasets with other tasks such as triple classification.
- **More Experiments**: Conducting experiments with text literals and also by fusing relational triples, numeric literals, short text literals (aliases and labels), and long text literals together. Moreover, experiments with more varieties of KGE models will be performed.
- **Detailed analysis**: Conducting further analysis on the datasets in terms of compositionality will be undertaken, so as to explore its use for models which leverage paths.
- **Studying data bias**: Bias in training data is one of the crucial aspects of Machine Learning that needs to be carefully addressed. Since Wikidata is one of the crowd-sourced KGs, it is susceptible to biases. These biases in Wikidata reflect the real-world and hence, LiterallyWikidata may as well be biased. However, the current version of

the dataset is not yet de-biased. An investigation will be carried out on whether de-biasing should be done and what methods should be used for such purpose.

In the next chapter, a novel KGE model for transductive LP task is introduced and one of the datasets from LiterallyWikidata is among the datasets used for the evaluation of the model.

7 IMPROVING LITERAL-BASED KGE MODELS

As discussed in Chapter 1, numeric literals contain useful semantics which could be leveraged when performing LP in KGs. To this end, some KGE methods have been proposed which make use of numeric literals (See Chapter 3 for more details). These literals could be used even further to extract implicit information from the KG when performing KGC. To achieve this, in this chapter, a novel KGE approach is presented which utilizes numeric literals for improved representation learning of KGs with a focus on the LP task in a transductive setting. The rest of the chapter is organized as follows. To begin with, the motivation behind the proposed approach is discussed in Section 7.1, followed by the problem formulation along with an overview of the base models considered in this work in Section 7.2. A detailed discussion of the proposed methodology is given in Section 7.3 whereas the experiment settings and the results obtained are provided in Section 7.4. Finally, Section 7.5 summarizes the chapter and provides directions for further research.

7.1 INTRODUCTION

KGs have recently gained massive attention for representing structured knowledge about a particular domain. As explained in Section 1, in addition to relational triples, these KGs also include triples with numeric literals (i.e., attributive triples). For example, in Figure 7.1.1, the triple `<Paper_B, nominatedFor, Best_Paper_by_Emerging_Authors>` is a relational triple whereas `<Anna, age, 24>` is an attributive triple. These literals contain valuable semantics which can be combined with relational triples to predict missing links in KGs. Few of the embedding-based approaches (KGE models) such as LiteralE [73], LiteralE-AT [160], and transforming literals into entities [112] perform LP by utilizing the literals present in the KGs [1].

In contrast to the above-mentioned models, this chapter presents a novel method named LitKGE to explore the advantages of utilizing numeric literal information not only for those entities which are directly connected to the literals but also for others that are indirectly associated with the literals. This enables the generation of numerical features for entities that do not occur in the set of attributive triples. The goal is to reveal implicit information and enhance the task of LP. This could be achieved using the property paths leading to numerical literals as features of those entities which are indirectly associated with the literals through graph traversal. For instance, in Figure 7.1.1 given a paper (i.e., `Paper_A`, `Paper_B`, or `Paper_C`), the average age and the average number of `papers` of its authors could be used to determine if the paper can be nominated for the `Best_Paper_by_Emerging_Authors` award. Moreover, this information also has a role in providing the semantics required to determine whether `Paper_A` should be close to `Paper_B` or `Paper_C` in the vector space. This is done by generating and treating property paths that lead to literal nodes as features of entities. `author_age` and `author_paper` are among the property paths that could be generated from Figure 7.1.1. For the entity `Paper_C`, the average of the age values of Lina and Martin (i.e., 60) is taken as the value for the first feature `(<Paper_C, author_age, 60>)`. The generated features could be used to further enrich the attributive triples and in turn to improve the LP task.

Generating features for entities in KGs has been investigated recently in Literal2Feature [161] for basic machine learning tasks such as regression and clustering of entities and the results achieved indicate that literals contain useful semantics about entities. In LitKGE, literal-based features of entities are considered as a source of semantics for the task of KGC, specifically LP. However, the feature generation procedure of LitKGE is different from that of Literal2Feature. In Literal2Feature, a graph traversal algorithm (either Breadth-First Search or Random Walk) is applied directly to the input RDF graph in order to gather literals for each entity and consider them as a feature vector for the given entity. Then, given the generated path, it keeps only the properties and the literal value, ignoring the entities appearing between the starting entity and the literal node. Differently from Literal2Feature, in this work, LitKGE proposes a novel algorithm named `WeiDNeR_extended` which generates a weighted and directed relation-relation/attribute network, which is used as an input to a random walk algorithm to generate property paths. Then these property paths are used to collect literals to leverage them as features of entities. The `WeiDNeR_extended` algorithm enables LitKGE to efficiently generate more sound features by using the weights in the resulting network graph.

Hence, the approach proposed in this work LitKGE aims to enhance the performance of KGE models with literals. Furthermore, it could be integrated with many KGE models which use numerical literals such as LiteralE, TransEA [120], etc. The contributions of this work are summarized as follows:

- A novel approach named LitKGE which generates features for entities in order to enhance the standard KGE models such as LiteralE, is introduced. LitKGE is a universal method which can be integrated with any KGE model which utilizes literals.
- LitKGE is evaluated on three LP datasets: FB15K-237[15], YAGO3-10[16], and LitWD48K[17]. These datasets are extended with the generated features and the extensions are made publicly available to facilitate further research.
- The experimental results indicate that making the implicit information explicit enabled LitKGE to outperform the SOTA models.

7.2 PRELIMINARIES

In this section, the formal definition of the LP problem that is being addressed in this chapter is given. Moreover, a detailed discussion on the LiteralE KGE model which is used

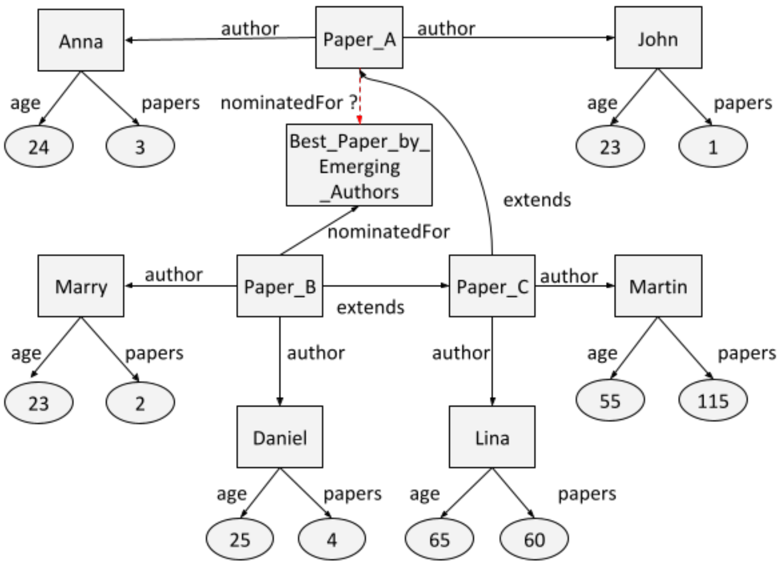


Figure 7.1.1: An example graph with entity nodes depicted as rectangles and literals as ovals.

as a base model to be improved with our approach is provided along with a description of the chosen scoring functions DistMult and ComplEx.

7.2.1 *Problem Definition*

Following the definition of KG provided in Chapter 2, let $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{D}, \mathcal{L}, \mathcal{T}\}$ be a KG where $\mathcal{E} = \{e_1, e_2, ..., e_{N_d}\}$ representing the set of entities in G, $\mathcal{R} = \{r_1, r_2, ..., r_{N_r}\}$ denotes the set of relations connecting two entities, $\mathcal{D} = \{d_1, d_2, ..., d_{N_d}\}$ as a set of attributes (a.k.a data relations) connecting entities to their corresponding literals, and $\mathcal{L}$ is the set of literal values. The triples in $\mathcal{G}$ are represented as $\mathcal{T} \subseteq ((\mathcal{E} \times \mathcal{E} \times \mathcal{R}) \cup (\mathcal{E} \times \mathcal{L} \times \mathcal{D}))$. Given $\mathcal{G}$, the task of LP can be formulated by a function $\Phi : \mathcal{E} \times \mathcal{E} \times \mathcal{R} \rightarrow \mathbb{R}$ which assigns a score to each triple $(e_i, e_j, r_k) \in \mathcal{E} \times \mathcal{E} \times \mathcal{R}$, where a higher score indicates that the triple is more likely to be valid. As discussed above, most of the KGs consist of numeric literals to provide semantic information about entities. In this work, the focus is to make use of the numeric literals to learn embeddings of the entities and relations for the purpose of predicting the missing links between entities in a KG. Hence, this chapter tackles the challenges in reference to the research question C2-RQ3 from Section 1.2 of Chapter 1.

- **C_2-RQ$_3$**: *Does generating entity features based on property paths and incorporating these features into existing KGE models result in improving LP tasks?*

7.2.2 *LiteralE*

As presented in Section 3, LiteralE is a universal approach that incorporates literals into latent feature methods (KGE models) such as DistMult and ComplEx via a learnable parametric function. This function takes as input an entity embedding and a literal feature vector and maps them to a new vector of the same dimension as the entity embedding. The resulting literal enriched vector is then passed as input to the given LP scoring function. As per the experimental results presented in a survey conducted on KGE models with numeric literals [1], LiteralE outperforms all other KGE models which use literals like KBLN, MTKGNN, and TransEA. Given a base model like DistMult with a scoring function $f = f_{distmult}$ shown in Equation 62 or ComplEx with $f = f_{complex}$ in Equation 63, LiteralE modifies f by replacing the original entity vectors e_i in f with literal enriched representations e_i^{lit} using a flexible and learnable GRU-based transformation function g.

$$f_{distmult}(e_i, e_j, r_k) = e_i^T diag(r_k) e_j \tag{62}$$

$$\begin{aligned} f_{complex}(e_i, e_j, r_k) &= Re(\langle e_i, e_j, r_k \rangle) \\ &= \langle Re(e_i), Re(e_j), Re(r_k) \rangle \\ &+ \langle Im(e_i), Im(e_j), Re(r_k) \rangle \\ &+ \langle Re(e_i), Im(e_j), Im(r_k) \rangle \\ &- \langle Im(e_i), Re(e_j), Im(r_k) \rangle \end{aligned} \tag{63}$$

The function g is shown in Equation 64 which takes as an input e_i and its corresponding literal vector l_i and maps them to a new vector. Note that l_i is the i-th row of a literal matrix $\mathbf{L} \in \mathbb{R}^{N_e \times N_d}$ as the literal vector of the i-th entity. The entry L_{ik} in L is the k-th literal value of the i-th entity if an attributive triple with the i-th entity and the k-th data relation (i.e., attribute) exists in the KG, and zero otherwise.

$$\begin{aligned} g : \mathbb{R}^H \times \mathbb{R}^{N_d} \to \mathbb{R}^H, \\ \mathbf{e}, \mathbf{l} \mapsto \mathbf{z} \odot \mathbf{h} + (1 - \mathbf{z}) \odot \mathbf{e}, \end{aligned} \tag{64}$$

where

$$\mathbf{z} : \sigma(\mathbf{W}_{ze}^T \mathbf{e} + \mathbf{W}_{zl}^T \mathbf{l} + \mathbf{b}), \tag{65}$$

and

$$\mathbf{h} = h(\mathbf{W}_h^T [\mathbf{e}, \mathbf{l}]). \tag{66}$$

Note that $\mathbf{W}_{ze} \in \mathbb{R}^{H \times H}, \mathbf{W}_{zl} \in \mathbb{R}^{N_d \times H}, \mathbf{b} \in \mathbb{R}^H$, and $\mathbf{W}_h \in \mathbb{R}^{H+N_d \times H}$ are the parameters of g, σ is the sigmoid function, $\odot$ denotes the element-wise multiplication, and h is a component-wise nonlinearity. The scoring function $f(\mathbf{e}_i, \mathbf{e}_j, \mathbf{r}_k)$ would be replaced with $f(g(\mathbf{e}_i, \mathbf{l}_i), g(\mathbf{e}_j, \mathbf{l}_j), \mathbf{r}_k)$.

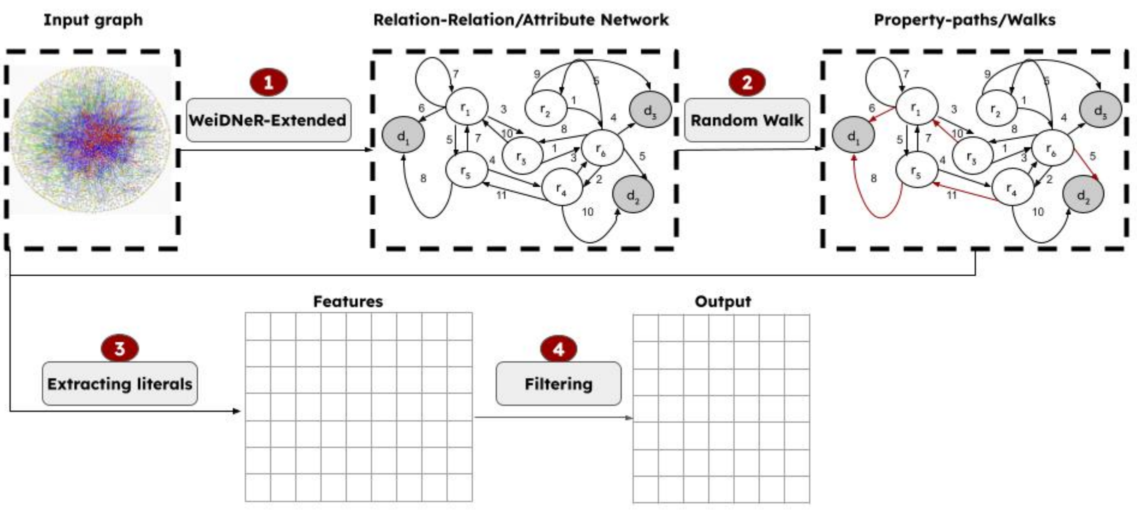


Figure 7.3.1: Feature generation pipeline: given a KG as input, it generates a feature matrix containing numerical features for the entities in the KG.

7.3 LITKGE

In this Section, the proposed approach LitKGE is presented in detail. It consists of two major components, i.e., i) Generating numeric features and ii) Incorporating numeric features into KGE models. These components are discussed in the subsequent sections.

7.3.1 *Generating features*

As discussed in Chapter 1 given a KG, implicit features of entities could be made explicit by traversing the KG using property paths. Hence, LitKGE generates such paths which lead to literal values to create features for entities. The overall phases in the feature generation process are depicted in Figure 7.3.1.

The phases are described as follows:

- **Input graph**: A KG that contains entity nodes, literal nodes, and properties (relations and attributes) is taken as input to generate numerical features.

- **WeiDNeR-Extended**: A novel algorithm named WeiDNeR-Extended is introduced which is an extension of the WeiDNeR algorithm developed for the RAILD model presented in Chapter 5. The are two major differences between WeiDNeR and WeiDNeR-Extended: i) WeiDNeR is designed based on the assumption that given a KG $\mathcal{G} = (\mathcal{R}, \mathcal{E}, \mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E})$, $r_1, r_2 \in \mathcal{R}$ and $\mathcal{T}_1 \subseteq (\mathcal{T} \cap (\mathcal{E} \times r_1 \times \mathcal{E}))$, $\mathcal{T}_2 \subseteq (\mathcal{T} \cap (\mathcal{E} \times r_2 \times \mathcal{E}))$, the higher the number of common entities between $\mathcal{T}_1$ and $\mathcal{T}_2$, the higher the probability that r_1 and r_2 could be semantically similar. In WeiDNeR-Extended, the assump-

tion is formulated as Given a KG $\mathcal{G} = (\mathcal{R}, \mathcal{A}, \mathcal{E}, \mathcal{L}, \mathcal{T}_R \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}, \mathcal{T}_A \subseteq \mathcal{E} \times \mathcal{A} \times \mathcal{L})$, $p_1 \in \mathcal{R}$, $p_2 \in \mathcal{R} \cup \mathcal{A}$ and $\mathcal{T}_1 \subseteq (\mathcal{T}_R \cap (\mathcal{E} \times p_1 \times \mathcal{E}))$, $\mathcal{T}_2 \subseteq ((\mathcal{T}_R \cup \mathcal{T}_A) \cap (\mathcal{E} \times p_2 \times \mathcal{E}|\mathcal{L}))$, the higher the number of matches between tail entities in $\mathcal{T}_1$ and head entities in $\mathcal{T}_2$, the higher the probability that p_1 and p_2 appear close to each other in the original KG. This restriction to matching only the tail of $\mathcal{T}_1$ and the head of $\mathcal{T}_2$ is performed in order to keep only outgoing links when generating property paths. ii) As described in i), WeiDNeR generates a relation-relation network taking only relational triples as inputs, i.e., excluding literals, hence, there are no attributes in the resulting network. However, in WeiDNeR-Extended, the network is intended to contain relation-relation as well as relation-attribute connections. Algorithm 2 describes the steps followed by WeiDNeR-Extended to generate the network.
Algorithm Description: Given two properties p_i and p_j,

– if p_i and p_j are different relations connecting entities, it computes the weight $Weight_{<p_i,p_j>}$ by counting the number of pair of triples where the relation in the first triple is p_i and in the second is p_j and the tail entity in the first triple is the same as the head entity in the second triple. If $Weight_{<p_i,p_j>}$ is greater than zero, then p_i and p_j are considered as nodes in the output network, and a directed link from p_i to p_j is created with the computed weight. $Weight_{<p_j,p_i>}$ is computed similarly and if it is greater than 0, then a directed link from p_j to p_i is created and assigned the computed weight (i.e., refer to lines 4- 9).

– if p_i is a relation connecting entities and p_j denotes an attribute connecting entity to a literal node, the weight $Weight_{<p_i,p_j>}$ by counting the number of pair of triples where the relation in the first triple (a relational triple) is p_i and in the second triple (an attributive triple) is p_j and the tail entity in the first triple is the same as the head entity in the second triple. If $Weight_{<p_i,p_j>}$ is greater than zero, then p_i and p_j are considered as nodes in the output network, and a directed link from p_i to p_j is created with the computed weight. The same analogy applies if p_i is an attribute and p_j is a relation so as to decide whether there should be a link from p_j to p_i and to compute the weight $Weight_{<p_j,p_i>}$ (i.e., refer to lines 11- 17).

– if p_i and p_j are relations and they are exactly the same, then the weight $Weight_{<p_i,p_i>}$ by counting the number of pair of triples where the relation in the first triple (a relational triple) is p_i and in the second triple (an attributive triple) is p_j and the tail entity in the first triple is the same as the head entity in the second triple. If $Weight_{<p_i,p_i>}$ is greater than zero, then p_i is considered as a node in the output network, and a link from p_i to itself (i.e., a loop)is created with the computed weight.

Note that the motivation behind creating a relation-relation/attribute network (i.e., proposing the WeiDNeR-Extended algorithm) is to make sure that within a property path that is generated as a feature, the properties are semantically related. For instance, considering the example relation-relation/attribute graph in the feature generation pipeline in Figure 7.3.1, when the random walk algorithm is applied to this graph and takes $r3$ as a starting node, it selects $r1$ as the next node instead of $r6$ because the weight on the link from $r3$ to $r1$ (i.e., 10) is bigger than the weight from $r3$ to $r6$ (i.e., 1). Moreover, having the weights in such relation-relation/attribute networks allows the generation of less-sparse features. This approach is novel due to the fact that none of the existing KGE methods utilized such a technique to generate features for entities.

- **Random walk**: The 2^{nd} order-based biased random walk approach used in [20] is applied to generate network neighborhoods for nodes in the WeiDNeR-Extended network.

- **Extracting literals:** The generated walks/property paths with the random walk graph traversal are used as templates to obtain literals for entities in the input KG. Given an entity, some property paths could have multiple values and in such cases, their average is taken. For example, taking the graph in Figure 7.1.1 as the input KG and $author\_age$ as a property path generated with a random walk, for Paper_B there would be two possible literal values 23 and 25 through the entities Marry and Daniel respectively (i.e., <Paper_B, author_age, 23> and <Paper_B, author_age, 25>) and taking the average would result in <Paper_B, author_age, 24>.

- **Filtering:** In this step, those property_paths/features which have values for only one entity are removed. This is done in order to avoid having features that do not provide much semantics to help compare entities but rather make the feature matrix sparse.

- **Output**: The resulting feature matrix $\mathcal{F} \in \mathbb{R}^{N_e \times N_f}$ contains the generated numerical features for the entities in the input KG, where N_e and N_f denote the number of entities and the number of features respectively. Note that each entry $\mathcal{F}_{ik}$ contains the k-th feature value of the i-th entity if a feature value is generated for the i-th entity and the k-th feature using the feature generation pipeline, and zero otherwise.

7.3.2 *Incorporating features into KGE models*

Once the features are generated using the procedure discussed in Section 7.3.1, they can be utilized while learning KGEs. This can be achieved by incorporating them in any KGE model which utilizes literals. For instance, TransEA and LiteralE are among those KGE models which can be improved using the LitKGE approach. TransEA is an extension of TransE model where an attribute embedding component is integrated into TransE. The attributive embedding component uses all attributive triples with numeric literals as input

Algorithmus 2 : WeiDNeR-Extended - An algorithm to generate a directed and weighted relation-relation/attribute network.

```
   Data : $T \leftarrow$ Relational and Attributive Triples in KG
   Data : $R \leftarrow$ Properties in Relational Triples (i.e., Relations)
   Data : $D \leftarrow$ Properties in Attributive Triples (i.e., Data relations/Attributes)
   Result : $N_{rel}$
1  for each pair of properties $\langle p_i, p_j \rangle$ do
2      if $p_i \neq p_j$ then
3          if $p_i, p_j \in R$ then
4              $Weight_{\langle p_i,p_j \rangle} \leftarrow |\{(\langle h_1,p_i,t_1\rangle,\langle h_2,p_j,t_2\rangle) : \langle h_1,p_i,t_1\rangle \in T, \langle h_2,p_j,t_2\rangle \in T, t_1 = h_2\}|$;
5              $Weight_{\langle p_j,p_i \rangle} \leftarrow |\{(\langle h_1,p_i,t_1\rangle,\langle h_2,p_j,t_2\rangle) : \langle h_1,p_i,t_1\rangle \in T, \langle h_2,p_j,t_2\rangle \in T, h_1 = t_2\}|$;
6              if $Weight_{\langle p_i,p_j \rangle} > 0$ then
7                  $N_{rel} \leftarrow N_{rel} \bigcup \{\langle p_i, p_j, Weight_{\langle p_i,p_j \rangle}\rangle\}$;
8              if $Weight_{\langle p_j,p_i \rangle} > 0$ then
9                  $N_{rel} \leftarrow N_{rel} \bigcup \{\langle p_j, p_i, Weight_{\langle p_i,p_j \rangle}\rangle\}$;
10         else if $p_i \in R$ & $p_j \in D$ then
11             $Weight_{\langle p_i,p_j \rangle} \leftarrow |\{(\langle h_1,p_i,t_1\rangle,\langle h_2,p_j,l\rangle) : \langle h_1,p_i,t_1\rangle \in T, \langle h_2,p_j,l\rangle \in T, t_1 = h_2\}|$;
12             if $Weight_{\langle p_i,p_j \rangle} > 0$ then
13                 $N_{rel} \leftarrow N_{rel} \bigcup \{\langle p_i, p_j, Weight_{\langle p_i,p_j \rangle}\rangle\}$;
14         else
15             $Weight_{\langle p_j,p_i \rangle} \leftarrow |\{(\langle h_1,p_j,t_1\rangle,\langle h_2,p_i,l\rangle) : \langle h_1,p_j,t_1\rangle \in T, \langle h_2,p_i,l\rangle \in T, t_1 = h_2\}|$;
16             if $Weight_{\langle p_j,p_i \rangle} > 0$ then
17                 $N_{rel} \leftarrow N_{rel} \bigcup \{\langle p_j, p_i, Weight_{\langle p_j,p_i \rangle}\rangle\}$;
18     else
19         if $p_i \in R$ then
20             $Weight_{\langle p_i,p_i \rangle} \leftarrow |\{(\langle h_1,p_i,t_1\rangle,\langle h_2,p_j,t_2\rangle) : \langle h_1,p_i,t_1\rangle \in T, \langle h_2,p_j,t_2\rangle \in T, t_1 = h_2, (h_1 \neq t_1 \vee h_1 \neq t_2)\}|$;
21             if $Weight_{\langle p_i,p_i \rangle} > 0$ then
22                 $N_{rel} \leftarrow N_{rel} \bigcup \{p_i, p_i, Weight_{\langle p_i,p_i \rangle}\}$;
```

and applies a linear regression model to learn embeddings of entities and attributes. One way to improve TransEA with LitKGE would be to use the generated features in LitKGE and treat them as attributes and predict the values using the attributive embedding component. On the other hand, LiteralE is a universal KGE model which incorporates literals to the standard KGE models such as DistMult and ComplEx. In this work, LiteralE is selected as a base model to improve it with LitKGE due to the fact that it performs better than TransEA and other KGE models which use literals as presented in [1]. In addition to its performance, LiteralE is a universal model, i.e., as mentioned above it can be applied to many standard models, which implies that applying LitKGE to LiteralE is in turn applying it to the base scoring models. The modification to the LiteralE architecture with LitKGE is depicted in Figure 7.3.2. Let $\mathbf{L} \in \mathbb{R}^{N_e \times N_d}$ be a matrix representing literals and $\mathbf{F} \in \mathbb{R}^{N_e \times N_f}$ be a newly introduced feature matrix generated using the procedure presented in Section 7.3.1. The final new matrix would be a combination of L and F as $L_F = [L; F]$, $\mathbf{L}_F \in \mathbb{R}^{N_e \times N_d + N_f}$ Then, the mapping function used in LiteralE (i.e., g in Equation 64) is modified as g_{lf} in Equation 67.

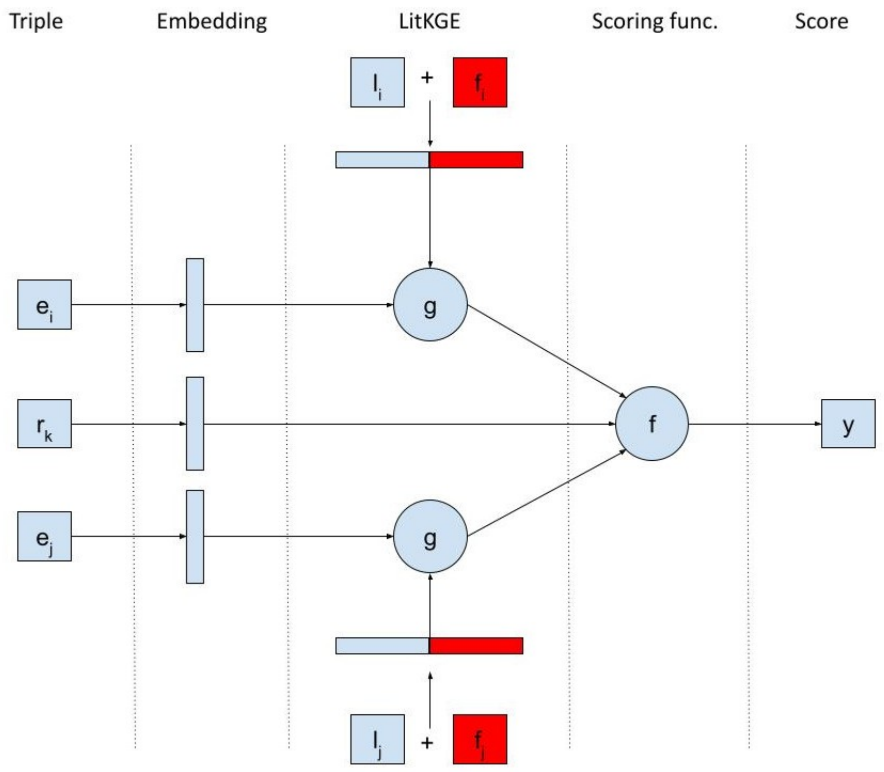


Figure 7.3.2: Overview of LitKGE as an improvement over LiteralE model. LitKGE takes as input the embedding of the entities and the concatenation of their corresponding literal vectors l_i and feature vector f_i as input and combines them via a learnable function g (i.e., g_{lf} in Equation 67). Then, it modifies the base scoring function f with the joint embedding obtained using g.

$$\begin{aligned} g_{lf} &: \mathbb{R}^{H} \times \mathbb{R}^{N_d+N_f} \rightarrow \mathbb{R}^{H}, \\ \mathbf{e}, \mathbf{l}_F &\mapsto \mathbf{z} \odot \mathbf{h} + (1-\mathbf{z}) \odot \mathbf{e}, \end{aligned} \tag{67}$$

where $\odot$ is the pointwise multiplication and

$$\mathbf{z} : \sigma(\mathbf{W}_{ze}^{T}\mathbf{e} + \mathbf{W}_{zl}^{T}\mathbf{l}_F + \mathbf{b}), \tag{68}$$

and

$$\mathbf{h} = h(\mathbf{W}_{h}^{T}[\mathbf{e}, \mathbf{l}_F]). \tag{69}$$

Note that $\mathbf{W}_{ze} \in \mathbb{R}^{H \times H}, \mathbf{W}_{zl} \in \mathbb{R}^{N_d+N_f \times H}, \mathbf{b} \in \mathbb{R}^{H}$, and $\mathbf{W}_h \in \mathbb{R}^{H+N_d+N_f \times H}$ are the parameters of g, σ is the sigmoid function, $\odot$ denotes the element-wise multiplication, and h is a component-wise nonlinearity. The scoring function $f(\mathbf{e}_i, \mathbf{e}_j, \mathbf{r}_k)$ would be replaced with $f(g(\mathbf{e}_i, \mathbf{l}_{Fi}), g(\mathbf{e}_j, \mathbf{l}_{Fj}), \mathbf{r}_k)$.

7.3.3 *Computational Complexity*

Note that given a KG, the numerical features of entities, as discussed in Section 7.3.1, are pre-computed, and hence, the major part of the computational cost of LitKGE comes from training the model depicted in the architecture in Figure 7.3.2. The cost can be computed in terms of the number of parameters in the model. LitKGE, similar to LiteraLE, introduces some overhead in the number of parameters as compared to the base model. The overhead is the number of parameters in the function g_{lf} in Equation 67. Specifically, there are $2H^2 + 2H(N_d + N_f) + H$ additional parameters corresponding to the dimensionality of W_h, W_{ze}, W_{zl}, and b in Equation 68 and 69. Hence, given g_{lf} and H, the number of additional parameters of LitKGE grows in $O(N_d + N_f)$, that is, linear to the number of attributes (a.k.a data relations) and features in the KG.

7.4 EXPERIMENTS

In this section, the details of experimentation including the datasets, the experimentation settings, and the results are discussed. Our implementation and the datasets are online[1] for the reviewing period and they will be made publicly available through github upon acceptance.

7.4.1 *Datasets*

Three standard datasets, namely, FB15K-237, YAGO3-10, and LitWD48K have been used to evaluate the performance of the proposed LitKGE LP approach. FB15K-237 is created by

1 shorturl.at/eklN7

Table 7.4.1: The statistics of the datasets used in the experiments in this chapter.

	FB15K-237	**YAGO3-10**	**LitWD48K**
#Entities	14,541	123,182	47,998
#Relations	237	37	257
#Attributes	121	5	297
#Relational Triples	310,116	1,089,040	336,745
#Numerical Attributive Triples	**70,257**	111,406	324,418
#Train	272,115	1,079,040	303,117
#Test	17,535	5,000	16,838
#Valid	20,466	5,000	16,838

removing the inverse relations from FB15K[49] which is a subset of Freebase[3] KG. YAGO3-10 is extracted from the YAGO3 knowledge graph, which mostly consists of triples related to people. For YAGO3-10, the numerical literals published in YAGO3-10-plus [74] and for FB15K-237 the numerical literals provided by LiteralE [73] are used for the experiments in this work. Differently from FB15K-237 and YAGO3-10, LitWD48K is a recent benchmark dataset extracted from Wikidata by explicitly designing it to contain literals so as to evaluate the performances of LP models. The statistics of these datasets are given in Table 7.4.1.

7.4.2 *Experiment Setting*

Note that the experiments are conducted using the scoring functions from two different base models DistMult and ComplEx and the LitKGE models corresponding to these functions are referred to as DistMult-LitKGE and ComplEx-LitKGE. In order to train the models, the strategy from LiteralE[73] is adopted. The hyperparameters used in our experiments across all datasets are: learning rate 0.001, batch size 128, embedding size 100, embedding dropout probability 0.2, and label smoothing 0.1. DistMult-LitKGE is trained for a maximum of 200 epochs on FB15K-23 and YAG03-10 datasets and 300 epochs on LitWD48K whereas ComplEX-LitKGE is trained for a maximum of 100 epochs on all datasets. More details on the hyper-parameters for the feature generation pipeline are provided in the resources made available publicly.

7.4.3 *Training*

The same training procedure used in LiteralE which is known as 1-N scoring approach is adopted to train the LitKGE models. For every triple (e_i, e_j, e_k) in the KG, the score for $(e_i, e_j^{'}, e_k)$, $\forall e^{'} \in E$ is computed by applying the extended scoring function f. Then, the sigmoid function is applied to the resulting score (i.e., $p = \sigma \circ f$), in order to interpret the result as the probability of the existence of a given triple. Finally, the probability vector $P \in [0,1]^{N_e}$ collects the probabilities computed w.r.t. all $e^{'} \in E$. Finally, the training is performed by minimizing the binary cross-entropy loss between p and the ground truth labels $y \in 0,1^{N_e}$ which indicates the existence of triples $(e_i, e_j^{'}, e_k)$, $\forall e^{'} \in E$. Given p_x and y_x as the predicted probability and the given truth value for the x-th element in the set $\{(e_i, e_j^{'}, e_k), e^{'} \in E\}$ respectively, the loss function is defined as shown in Equation 70 and optimized using the Adam [162] optimizer.

$$L(p,y) = -\frac{1}{N_e} \sum_{x=1}^{N_e} (y_x \log(p_x) + (1 - y_x) \log(1 - p_x)) \tag{70}$$

7.4.4 *Evaluation*

In order to evaluate the performance of the proposed approach on the LP task, the standard setup in other similar studies including LiteralE is followed. For each triple (e_i, e_j, r_k) appearing in the test set, a set of corrupted/negative triples by either replacing the head entity e_i or the tail entity e_j with any other entity $e^{'} \in \mathcal{E}$. The scores are computed for the corrupted triples as well as the true triple. Then, all triples with respect to their scores are ranked and evaluated using the standard evaluation metrics: Mean Reciprocal Rank (MRR), Hits@1, Hits@3, and Hits@10.

7.4.5 *Results and Discussion*

The statistics of the generated features for the datasets FB15K-237, YAG03-10, and LitWD48K with the proposed pipeline in this work, are presented in Table 7.4.2. Moreover, the results of the LP experiments conducted on these datasets are provided in Table 7.4.3. Overall, as these results indicate, the model LitKGE outperforms all the other models across all datasets with respect to all metrics using the DistMult scoring function. The results are discussed in detail as follows:

Table 7.4.2: Analysis of the features generated for the entities in the three datasets. #feat denotes the number of unique features and #feat-entries is the number of entries with these features for the entities in the corresponding dataset. #feat-max, #feat-min, and #feat-median represent the maximum, minimum, and median of the occurrences of the features.

	FB15K-237	YAGO3-10	LitWD48K
#feat	11	5	828
#feat-entries	2,337	10,151	192,035
#feat-max	1,127	5,072	15,289
#feat-min	8	347	2
#feat-median	170	1,403	12

7.4.5.1 *LitKGE vs. Base models (DistMult and ComplEx)*

DistMult-LitKGE outperforms DistMult on all three datasets with respect to all metrics. Specifically, applying LitKGE on top of DistMult improves the MRR score by 1.18%, 11.87%, and 10.38% on LitWD48K, FB15K-237, and YAGO3-10 datasets, respectively. Similarly, applying LitKGE on top of ComplEx improves the MRR score with 1.87% and 3.97%, respectively on LitWD48K and FB15K-237. It can be noted that applying ComplEx-LitKGE does not outperform ComplEx on YAGO3-10. This might be attributed to the fact that the ComplEx base model already achieves higher performance than DistMult. Note that this is also the case with ComplEx-LiteralE, i.e., ComplEx-LiteralE does not improve ComplEx on YAGO3-10.

7.4.5.2 *LitKGE vs. LiteralE*

When comparing LitKGE with LiteralE, it is seen that LitKGE outperforms LiteralE across all three datasets with the DistMult scoring function with respect to all metrics. DistMult-LitKGE improves DistMult-LiteralE's MRR score by 2.56%, 0.94%, and 7.88% on LitWD48K, FB15K-237, and YAGO3-10 datasets, respectively. Similarly, applying LitKGE on top of ComplEx improves the MRR score by 1.25% on LitWD48K dataset.

7.4.5.3 *Impact of the feature matrix*

As shown in Table 7.4.2, the LitWD48K dataset has many features (i.e., 828) as compared to the other datasets, and also it has as large as 192,035 entries which makes the resulting feature matrix less sparse. This contributes to the fact that LitKGE outperforms all the SOTA models on this dataset. Note that, unlike LitWD48K, FB15K-237 and YAGO3-10 are not created specifically for the evaluation of KGE models with literals, i.e., not all of their entities have literals. However, these datasets could benefit from the feature generation approach in LitKGE since LitKGE also generates features for those entities which do not have

Table 7.4.3: LP results on FB15K-237, YAGO3-10, and LitWD48K. The best values are highlighted in bold text.

	LitWD48K			
	mrr	hits@1	hits@3	hits@10
DistMult	0.336	0.264	0.360	0.480
ComplEx	0.315	0.248	0.341	0.442
DistMult-LiteralE	0.331	0.258	0.352	0.480
ComplEx-LiteralE	0.317	0.238	0.343	0.475
DistMult-LitKGE	**0.340**	**0.266**	**0.363**	**0.491**
ComplEx-LitKGE	0.321	0.243	0.344	0.480
	FB15K-237			
DistMult	0.282	0.203	0.309	0.438
ComplEx	0.290	0.212	0.317	0.445
DistMult-LiteralE	0.317	0.232	0.348	0.483
ComplEx-LiteralE	0.305	0.222	0.336	0.466
DistMult-LitKGE	**0.320**	**0.234**	**0.354**	**0.488**
ComplEx-LitKGE	0.302	0.219	0.333	0.465
	YAGO3-10			
DistMult	0.466	0.377	0.514	0.653
ComplEx	0.493	0.411	0.536	0.649
DistMult-LiteralE	0.479	0.4	0.525	0.627
ComplEx-LiteralE	0.485	0.412	0.527	0.618
DistMult-LitKGE	**0.520**	**0.446**	**0.563**	**0.653**
ComplEx-LitKGE	0.481	0.406	0.522	0.615

any literals associated with them. This gain is observed in the LP results obtained on these datasets with LitKGE where LitKGE outperforms the SOTA models.

7.4.5.4 *Impact of filtering while generating features*

As discussed in Section 7.3.1, those features which have values for only one entity are filtered out. This is performed in order to make the feature matrix less sparse. An ablation study is conducted on the LitWD48K dataset with the DistMult-LitKGE model in order to show the impact of performing the filtering step in the feature generation pipeline in Figure 7.3.1. As the results in Table 7.4.4 indicate, DistMult-LitKGE outperforms $\mathrm{DistMult-LitKGE}_{\mathrm{unfiltered}}$ with respect to all metrics, mainly with hits@10 met-

Table 7.4.4: Comparison of the LP results on LitWD48K dataset with and without applying the filtering step in the feature generation pipeline. DistMult-LitKGE is with filtering whereas DistMult-LitKGE$_{\text{unfiltered}}$ is when filtering is not used.

	mrr	hits@1	hits@3	hits@10
DistMult-LitKGE	**0.340**	**0.266**	**0.363**	**0.491**
DistMult-LitKGE$_{\text{unfiltered}}$	0.334	0.263	0.357	0.474

rics which is improved by 3.46%. This improvement is attributed to the filtering step in the pipeline during which 269 property_paths/features occurring only once are removed. This proves that including filtering in the pipeline plays its role in reducing sparsity.

7.4.5.5 *Experiment with a different scoring function*

In this work, as shown above, DistMult and ComplEx are chosen as scoring functions for LitKGE based on the following considerations. Firstly, DistMult and ComplEx are widely adopted and have been demonstrated to be effective latent feature methods, as reported in Chapter 3 and in [73]. By utilizing these scoring functions, it is possible to compare the results obtained with LitKGE against the SoTA models such as LiteralE-DistMult and LiteralE-ComplEX. Secondly, it is convenient to analyze the results obtained using datasets with different characteristics, such as comparing the symmetric and inverse handling capabilities of these models on various datasets. Thirdly, as discussed in Chapter 6 where a comparison of datasets for KGE with literals is provided, FB15K-237 is better than YAGO3-10 and FB15K to evaluate KGE models with literals. Besides, according to the results presented in [73], the DistMult and ComplEx scoring functions perform better than ConvE on the FB15K-237 dataset. Therefore, DistMult and ComplEx are chosen in this work to evaluate LitKGE.

Table 7.4.5: Comparison of LitKGE with ConvE scoring function against the SOTA models on the Yago3-10 dataset

	mrr	hits@1	hits@3	hits@10
ConvE-LiteralE	0.525	0.448	0.572	0.659
ConvE-LitKGE	**0.540**	**0.467**	**0.583**	**0.671**

However, it is very important to note that the methods DistMult and ComplEx are a representative selection and LitKGE can also be adapted to alternative latent feature methods, such as ConvE. In order to support this argument, an additional experiment is conducted with the ConvE scoring function on YAGO3-10 and the results are presented in Table 7.4.5. Note that YAGO3-10 dataset is selected due to the fact that ConvE performs best only on this dataset as compared to the other datasets with the current approaches [73]. The results presented in Table 7.4.5 indicate that LitKGE with the ConvE scoring function outperforms the current best-performing model ConvE-LiteralE proposed in [73].

7.5 CONCLUSION AND OUTLOOK

In this chapter, a novel approach named LitKGE that generates features for entities in order to enhance the standard KGE models such as LiteralE for the task of transductive LP is introduced. LitKGE is a universal method that can be integrated with any KGE model which utilizes literals. Its main focus is on making implicit information explicit so that KGE models could leverage it. To do so, it introduces an algorithm named WeiDNeR_Extended to generate a relation-relation/attribute network which is in turn used to generate property paths. The resulting property paths are further processed to get numerical features for the entities in the KG. Finally, the obtained features are incorporated into the LP task. LitKGE is tested on LitWD48K, FB15K-237, and YAGO3-10 datasets and it outperforms all the SoTA models across all these datasets. These datasets are extended with the generated features and the extensions are made publicly available to facilitate further research. The answer to the major research question tackled in this chapter is provided as follows.

- **C_2-RQ_3**: *Does generating entity features based on property paths and incorporating these features into existing KGE models result in improving LP tasks?*
  - Following the proposed approach LitKGE, as discussed in Section 7.3, it is possible to generate numerical entity features using property paths. Incorporating these features into the LP task results in making the implicit information explicit leading to enabling LitKGE to outperform the SoTA models, as presented in Table 7.4.3, Table 7.4.5, and Table 7.4.4.

The following are possible future works in order to further improve the proposed model:

- Exploring schema definitions and constraints of properties to deal with those KGs with possible duplicate paths and to handle sparsity in the feature matrix.
- Extending LitKGE by fusing the relational triples and the numeric literals with short textual literals such as labels and aliases and also long textual literals such as the description of entities.
- Studying explainability of the LitKGE model.

In this chapter, the benefits of numerical literals for the task of LP are demonstrated whereas, in the next chapter, the capability of multilingual LMs in capturing semantics present in the entity descriptions available in multiple natural languages is investigated for the task of LP.

This chapter showcases the advantages of using numerical literals in LP, while the following chapter explores how multilingual LMs can capture semantics from entity descriptions in various natural languages for LP purposes.

8 LEVERAGING MULTILINGUAL ENTITY DESCRIPTIONS

Most KGs contain textual descriptions of entities in various natural languages. These descriptions of entities provide valuable information that may not be explicitly represented in the structured part of the KG. Based on this fact, some LP methods which make use of the information presented in the textual descriptions of entities have been proposed to learn representations of KGs. However, these methods use entity descriptions in only one language and ignore the fact that descriptions given in different languages may provide complementary information and thereby also additional semantics. In this chapter, the benefits of multilingual embeddings for incorporating multilingual entity descriptions into the task of LP in KGs are investigated and initial results are presented.

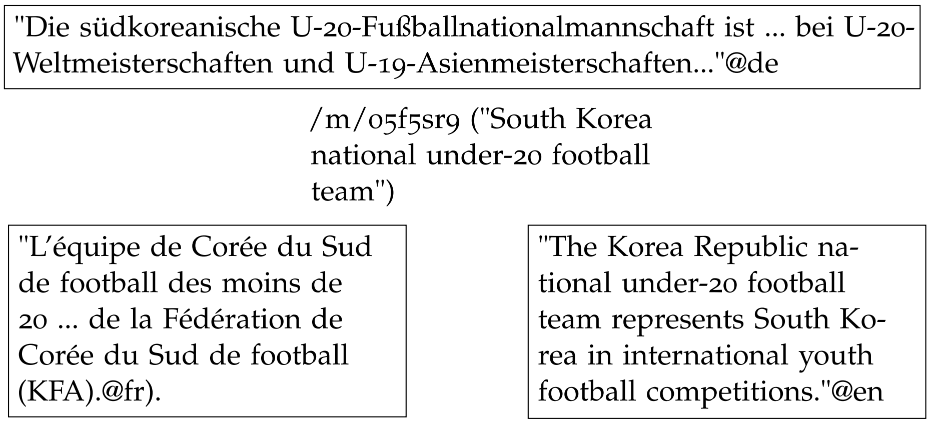
The rest of the chapter is organized as follows. To begin with, the motivation behind the work is provided in Section 8.1 followed by a discussion on the proposed approach in Section 8.2. The experiments conducted on the LP task are presented in Section 8.3. Finally, concluding remarks with directions for future work are given in Section 8.4.

8.1 INTRODUCTION

In recent years, there has been extensive research on KGE with a focus on predicting missing links in KGs. Most of these models use only relational triples (i.e., triples with object properties) such as TransE [49] and ConvE [16] whereas a few others include textual entity descriptions such as DKRL [60], MKBE[74] , and Jointly[63]. Furthermore, most popular KGs contain descriptions in two or more languages for a single entity due to the multilingual community working on these KGs (as in Wikidata) or the multilingual nature of its sources (as in DBpedia). The cultural context and bias associated with each of these descriptions induce a difference with regard to content. However, despite the fact that entity descriptions are available in multiple natural languages, all the existing models including DKRL consider only one language. Figure 8.1.1 presents an example scenario showing the differences in contents of multilingual descriptions of a single entity. In this example, the description in German contains information which does not appear in the English or French descriptions. For instance, the fact that the team is the record winner of the U-19 Asian Cup with twelve titles is only mentioned in the German description.

This chapter addresses the challenge in reference to the following research question C2-RQ4 defined in Section 1.2 of Chapter 1.

- **C_2-RQ_4**: How beneficial is to leverage multilingual embeddings to incorporate multilingual entity descriptions into the task of LP in KGs?



"Die südkoreanische U-20-Fußballnationalmannschaft ist ... bei U-20-Weltmeisterschaften und U-19-Asienmeisterschaften..."@de

/m/05f5sr9 ("South Korea national under-20 football team")

"L'équipe de Corée du Sud de football des moins de 20 ... de la Fédération de Corée du Sud de football (KFA).@fr).

"The Korea Republic national under-20 football team represents South Korea in international youth football competitions."@en

Figure 8.1.1: An entity from Freebase with descriptions from its corresponding English, German, and French Wikipedia pages. For instance, the description in German provides more content that is not in the descriptions of either the English or the French Pages.

In order to tackle this question, the performance of the existing model DKRL in leveraging multilingual descriptions using multilingual embeddings has been analyzed and the results of the initial experiments are discussed. The contributions of this work are summarized as follows.

- Empirical evaluation of the performance of the existing DKRL, i.e., CNN-based KGE model, in encoding descriptions based on multilingual embeddings.
- Analysis of the capability of the MUSE[1] embedding model in capturing semantics from descriptions in multiple languages.

8.2 METHODOLOGY

As discussed in [163], an entity alignment model named KDCoE [65] has demonstrated the advantage of multilingual word embeddings by using a cross-lingual Bilbowa word embedding [164] to encode multilingual descriptions for the task of cross-lingual learning. In this work, the same approach is adopted to encode multilingual entity descriptions for a LP task on a monolingual dataset. In particular, the experiments have been performed with one of the existing models DKRL [60] using pretrained multilingual word embeddings by MUSE. DKRL is an extension of TransE [49], which learns two kinds of vector representations for an entity, i.e., structure-based and description-based representations. DKRL adopts TransE for the structure-based representation and uses CNN to encode entity descriptions for the description-based representations. These two kinds of entity representations are learned simultaneously into the same vector space without forcing them to be unified. For our experiments, in order to effectively utilize multilingual descriptions, the embeddings of the

1 https://github.com/facebookresearch/MUSE

words in the descriptions obtained by MUSE are passed as inputs to the encoder. MUSE has been chosen because this work deals with multilingual descriptions and MUSE aligns embeddings (specifically, FastText embeddings) of words in different languages into the same vector space. Figure 8.2.1 shows the CNN encoder part of DKRL with pretrained word embeddings from multilingual descriptions as inputs.

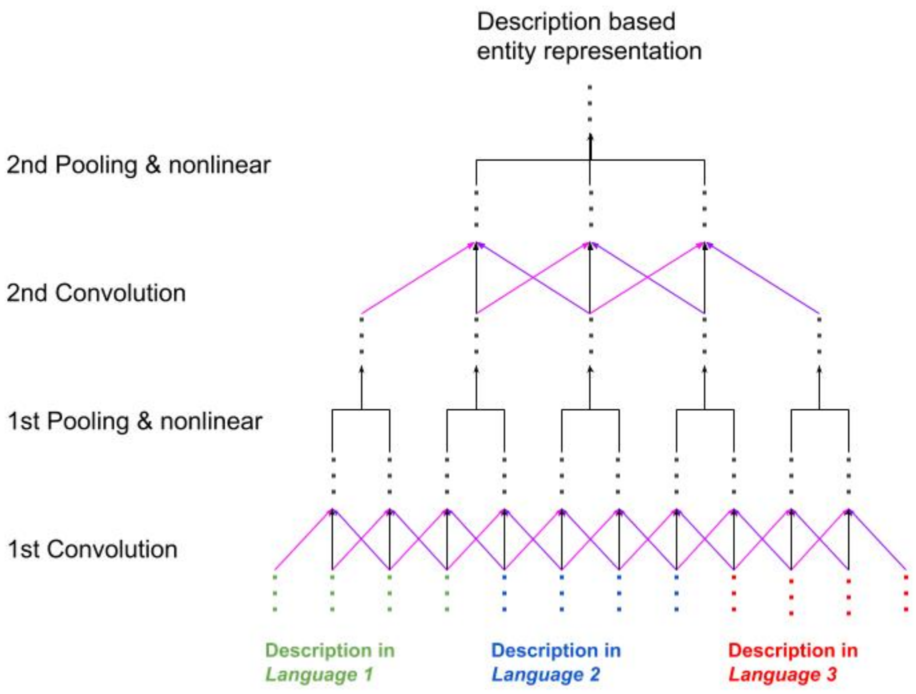

Description based entity representation
2nd Pooling & nonlinear
2nd Convolution
1st Pooling & nonlinear
1st Convolution
Description in Language 1
Description in Language 2
Description in Language 3

Figure 8.2.1: Passing pretrained multilingual word embeddings to a CNN encoder which is adopted from DKRL [60] and shown in [163], in order to encode multilingual entity descriptions.

8.3 EXPERIMENTAL EVALUATION

In this section, the experiments conducted to incorporate textual descriptions in English, French, and German into DKRL (for the task of transductive LP) are presented.

DATASETS Table 8.3.1 shows the dataset created out of FB15K-237 [15] for the experiments by removing those triples for which either the head or tail entity does not have descriptions in at least one of the three languages mentioned above or have less than 3 words after preprocessing. Since FB15K-237 is a dataset generated from Freebase and the entity descriptions in Freebase are old, the descriptions in all the three languages have been constructed by taking the information from the summary part of their respective Wikipedia pages. During preprocessing, stop words are removed and all phrases are marked using entity names and also by applying Spacy's[2] named entity recognizer. The modified dataset is made publicly available[3].

EXPERIMENTAL SETTING For the experiments with DKRL and TransE, the code[4] published by the authors of the DKRL paper and the code[5] made available by OpenKE [165] has been used respectively. The DKRL model has been trained on three varieties of the dataset, given the names DKRL$_e$, DKRL$_{eg}$, and DKRL$_{egf}$. For DKRL$_e$, only the descriptions in English are used whereas for DKRL$_{eg}$ the combination of descriptions in German and English are used. On the other hand, descriptions in all three languages are used to train DKRL$_{egf}$. The minimum, maximum, and average number of words are 3, 615, and 107.351 for the descriptions in DKRL$_e$, 9, 970, and 140.591 in DKRL$_{eg}$, and 18, 1460, and 192.091 in DKRL$_{egf}$ respectively.

In DKRL$_e$, the words are initialized using FastText pretrained embeddings and for the other two models, MUSE pre-trained embeddings are used. As shown in Table 8.3.2, TransE [49] has also been trained on the new dataset for a fair comparison with DKRL. The hyperparameters are chosen from embedding size {50, 100, 150}, margin {1.0, 2.0, 3.0, 4.0, 5.0}, learning rate {0.01, 0.1, 1.0} (following the same procedure as in the paper of TransE). The optimal parameters for TransE on this dataset is embedding size: 100, margin: 4.0, learning rate: 0.1, and epoch: 1000. For all the other models DKRL$_e$, DKRL$_{eg}$, and DKRL$_{egf}$, the same procedure as in the original study DKRL has been adopted. The optimal parameters are entity and relation embedding size: 100, learning rate 0.001, margin: 1.0, window-size: 2, dimension of feature map: 100, and word embedding size: 300, for all the three varieties. DKRL$_e$ is trained for 1000 epochs where as DKRL$_{eg}$ and DKRL$_{egf}$ are trained for 1200 epochs.

RESULTS As shown in Table 8.3.2, incorporating descriptions into the LP task brings improvement over TransE. However, when comparing the different varieties of DKRL, it is seen that combining multiple descriptions has only a slight improvement. For instance, hits@10 is the same for DKRL$_{eg}$ and DKRL$_{egf}$. One potential reason for such results could be the out-of-vocabulary words in the pre-trained word embeddings by MUSE. There are 18.4% and 20% out of vocabulary words for DKRL$_{eg}$ and DKRL$_{egf}$ respectively and they are randomly initialized.

2 https://spacy.io/
3 https://github.com/ISE-FIZKarlsruhe/Link-Prediction-with-Multilingual-Entity-Descriptions
4 https://github.com/xrb92/DKRL
5 https://github.com/thunlp/OpenKE

Table 8.3.1: The statistics of the dataset used for the experiments.

	FB15K-237
#Entities	12729
#Relations	234
#Train triples	219573
#Valid triples	13919
#Test triples	16084

Table 8.3.2: Experiment results using transE and DKRL models on the different varieties of the FB15K-237 dataset.

	MR	MRR	Hits@1	Hits@3	Hits@10
TransE	213	0.266	0.175	0.297	0.448
DKRL$_e$	201	0.275	0.189	0.304	0.449
DKRL$_{eg}$	185	0.280	0.191	0.310	0.457
DKRL$_{egf}$	180	0.285	0.196	0.311	0.457

8.4 CONCLUSION AND OUTLOOK

In this chapter, preliminary results from an ongoing work to discuss the benefits of leveraging multilingual embeddings for the task of LP are presented. Specifically, DKRL which is one of the existing KGE models has been evaluated on making use of the information present in multilingual descriptions. For the experiments, an extension of the FB15K-237 dataset with textual descriptions of entities extracted from their corresponding English, German, and French Wikipedia pages is created. The experiments conducted show promising results that encourage the use of entity descriptions available in multiple languages. The answer to the research question that is addressed in this chapter is given as follows.

- **C$_2$-RQ$_4$**: How beneficial is to leverage multilingual embeddings to incorporate multilingual entity descriptions into the task of LP in KGs?
  - Multilingual embeddings play a vital role in capturing the semantics present in entity descriptions in multiple natural languages. A promising result is obtained with the MUSE multilingual embedding model to capture descriptions in English, German, and French to improve the task of transductive LP, as shown in Table 8.3.2.

As future work, more experiments will be conducted by aligning pre-trained FastText embeddings which have a bigger vocabulary size, using MUSE, to avoid the problem which rises due to out-of-vocabulary words. Moreover, exploring other multilingual LMs such as Multilingual BERT will be considered. Another alternative to investigate is to learn description-based embeddings of an entity separately for each language and then fuse the vectors using different approaches.

Part V

CONCLUSION AND OUTLOOK

9 CONCLUSION AND OUTLOOK

This dissertation explores the use of literals for representation learning on KGs through the task of LP. It aims to address two primary challenges in LP: transductive LP and inductive LP. The literature reviews presented in Chapters 3 and Chapter 4 show that there exist some methods that make use of literals to learn embeddings of entities and relations into a low-dimensional vector space for LP. However, the full potential of different types of literals in a KG is still underutilized. In this thesis, different KGE-based LP models leveraging literals are proposed to predict the missing links in the KG in transductive and inductive settings. Moreover, several benchmark datasets are also introduced for both LP settings to address the lack of high-quality evaluation datasets for KGC. The contributions of this thesis are summarized in this chapter, along with areas for future research.

9.1 CONCLUSIONS

The importance of different types of literals for KGE, particularly in the context of LP, is studied in this dissertation. One of the contributions of the thesis is an extensive survey of SoTA KGE methods, along with an empirical evaluation on the task of LP. Furthermore, based on the findings of the survey, different novel KGE models are introduced along with new benchmark datasets for KGC.

Chapter 3 and Chapter 4 provide a literature review on the SoTA KGE models, for transductive and inductive LP respectively. The research question *C2-RQ1* from Section 1.2 of Chapter 1 is answered in Chapter 3 by conducting a detailed survey on the existing KGE models with literals focusing on transductive LP models. The survey covers the following parts: a thorough description of the architectures of the models, an analysis of the applications or tasks that the models are trained or evaluated on, an analysis of the various KGC evaluation datasets, an empirical evaluation of the models on transductive LP task, and a discussion on the shortcomings of the models and the datasets. In Chapter 4, a comprehensive review of the methods that are proposed for inductive LP is provided, along with a discussion of the existing evaluation datasets for inductive LP.

In Chaper 5, based on the findings of the literature review on inductive LP, a novel embedding model named *RAILD* is proposed to tackle the research question *C1-RQ1* from Section 1.2 of Chapter 1. This model is designed to enable inductive LP involving relations that are not observed during training, i.e., unseen relations, by using textual literals and the contextual information available in the KG. The textual descriptions of entities and relations are used to generate features with a neural LM for the corresponding entities and relations. In order to capture the contextual information for relations, the model generates a directed and weighted relation-relation network from triples in a KG using a novel algorithm called *WeiDNeR* and learns embeddings of the relations by applying a node embedding approach. The two components, i.e., the text-based and graph-based (contextual information) are combined during fine-tuning of a pre-trained LM with a LP objective. Furthermore, in the results obtained from the experiments, it is observed that combining the textual description and contextual information about relations significantly improves the prediction results. One of the key benefits of the model is that it is able to learn latent representations for unseen relations while performing LP. It is to be noted that, similar to the existing models which fine-tune neural LMs, the proposed model is computationally expensive as compared to simple KGE models such as TransE. However, the main focus is not on the runtime, but rather on the model's effectiveness, which is easily demonstrated by the results obtained. Moreover, in Chapter 5, a new benchmark dataset named Wilikdata68K is introduced to address the research question *C1-RQ2* from Section 1.2 of Chapter 1. The release of this dataset is expected to facilitate more research on inductive LP with unseen relations, as there is currently no dataset of this kind.

Taking into account the limitations of the existing KGC datasets for the evaluation of transductive LP models with literals, as highlighted in the literature review provided in this thesis, a set of novel KGC benchmark datastes named *LiterallyWikidata* is introduced in Chapter 6. These datasets are created to tackle the research question *C2-RQ2* from Section 1.2 of Chapter 1. The main focus is to create high-quality benchmark datasets containing different types of literals. Hence, the pipeline used to generate the datasets applies different steps to ensure that the datasets are not skewed, do not contain any duplicates, and are not easy for LP but rather pose a significant challenge. The datasets contain numeric literals, and short and long textual literals such as labels, aliases, and descriptions for relations and entities from Wikidata and Wikipedia.

In Chapter 7, the benefits of leveraging property paths that lead to numerical literals in a KG are investigated for the task of transductive LP. The aim is to find solutions to the research question *C2-RQ3* from Section 1.2 of Chapter 1 by proposing a novel KGE model named LitKGE that makes implicit information explicit for entities using property paths and numeric literals. LitKGE provides a universal approach to integrate numerical entity features into any KGE model which utilizes literals. This is accomplished by introducing an algorithm named WeiDNeR_Extended, which generates a relation-relation/attribute network used to produce property paths through graph traversal. The resulting property paths are then processed to extract numerical features for entities in the KG, which are subsequently integrated into the LP task. Leveraging literals in along with property paths enabled LitKGE to outperform all the existing SoTA models on multiple datasets. The datasets have been extended with the generated features and made publicly available for further research.

Finally, in Chapter 8, the answer to the research question *C2-RQ4* from Section 1.2 of Chapter 1 is provided. The main objective is to investigate the advantages of utilizing multilingual embeddings to incorporate multilingual entity descriptions into the task of

Table 9.1.1: Resources that are published as part of this thesis

Approach	Resources
RAILD	Github: `https://github.com/GenetAsefa/RAILD`
	Zenodo: `https://doi.org/10.5281/zenodo.7066504`
LiterallyWikidata	Github: `https://github.com/GenetAsefa/LiterallyWikidata`
	Zenodo: `https://doi.org/10.5281/zenodo.4701190`
Multilingual-KGE	Github: `https://github.com/ISE-FIZKarlsruhe/Link-Prediction-with-Multilingual-Entity-Descriptions`

LP in KGs. To achieve this, DKRL, one of the existing KGE models, has been assessed on making use of the information present in multilingual descriptions, and the obtained initial results are presented. To conduct the experiments, an extension of the FB15K-237 dataset with textual descriptions of entities in English, German, and French natural languages is used collected from Wikipedia. The results of the experiments are encouraging and suggest that incorporating entity descriptions available in multiple languages can be beneficial.

A summary of the various resources that are published as part of the different works presented in this thesis is given in Table 9.1.1. These resources include links to source codes and datasets which are made available on Github and Zenodo. Adhering to the FAIR principles (Findable, Accessible, Interoperable, and Reusable), these resources are openly shared, promoting their availability and usability by the wider research community.

9.2 OPEN ISSUES AND OUTLOOK

This dissertation is anticipated to inspire the exploration of novel solutions for open challenges that have yet to be tackled. In this section, the open issues of this thesis are presented, along with potential directions for future research.

The focus of this thesis lies in predicting missing links within a KG in transductive and inductive settings. However, LP can also be carried out across different KGs to predict the missing links between the two same entities across KGs, which is referred to as Entity Alignment. One way to accomplish this is to learn the embedding of the entities and the relations of the source and the target KGs separately and align them using a supervised model. Another way is to perform joint representation learning of the two KGs into a unified space. The LitKGE approach proposed in this thesis can be extended to perform a KG alignment task. Furthermore, the feature generation technique introduced in LitKGE could also be used to perform other machine-learning tasks on KGs.

In this thesis, numerical literals are utilized by LitKGE for transductive LP. However, there can be additional semantics about numerical literals such as their datatypes which should also be incorporated into the representation learning. Moreover, other sources of information like literals containing URIs leading to files such as audio, PDFs, and videos are not considered. Therefore, another future research directions would be to leverage the semantics present in these additional sources of information for the task of LP.

As discussed in the literature reviews of this thesis, there are some LP methods that are based on logical rules. Specifically, to the best of our knowledge, none of the existing rule-based inductive LP methods utilize literals. This indicates that there is a gap in leveraging literals with rule-based methods to perform LP tasks. Hence, future research could explore the advantages of combining literals and logical rules for LP to leverage both the explanatory qualities of rule-based systems and the semantics present in the literals.

One of the challenges that remain open in this thesis is the prediction of literals, also known as attribute values. The methods introduced in this thesis concentrate on utilizing literals to enhance the embeddings of entities and relations, and do not directly predict attribute values. As a result, a potential future task to enhance the proposed approaches would be to predict attribute values. Moreover, as the surveys conducted in this thesis indicate, there has been limited exploration into learning representations for literals. Therefore, this presents an interesting challenge for future research.

Another challenge related to KGE is the utilization of entity embeddings learned on LP tasks in real-world use-cases. For instance, the work in [166] presents a KGE-based approach applied to scholarly data for the task of author name disambiguation using literals. Another study investigates the challenges of enriching a KG which is generated from a real-world relational database (RDB) about companies, with information from external sources such as Wikidata and learning representations for the KG. Similarly, a possible future direction would be to apply the embeddings learned using the KGE methods proposed in this thesis i.e., RAILD and LitKGE, for downstream tasks in different real-world use-cases.

To sum up, it is anticipated that the findings and the contributions made in this thesis will play a vital role in advancing the progress of techniques employed for KGE and its utilization in various fields.

Part VI

APPENDIX

A

APPENDIX

A.1 SUMMARY OF APPLICATIONS

Table A.1.1: Summary of different applications on which the KG embedding techniques with literals, in their original papers, have been trained and/or evaluated

	Link Prediction	Triple Classification	Entity Classification	Entity Alignment	Attribute value prediction	Nearest neighbour analysis	Data linking	Document classification	Relational Fact Extraction
Extended RESCAL	✓		✓						
LiteralE	✓					✓			
TransEA	✓								
KBLRN	✓								
DKRL	✓		✓						
KDCoE	✓			✓					
KGlove with literals								✓	
MADLINK	✓	✓							
Transforming literals into entities [112]	✓								
IKRL	✓	✓							
EAKGE	✓			✓					
MKBE	✓				✓				
MT-KGNN		✓			✓				
LiteralE with blocking							✓		
Jointly(Desp)	✓	✓							✓
Jointly	✓	✓							
SSP	✓		✓						
MTKGRL	✓	✓							

BIBLIOGRAPHY

[1] Genet Asefa Gesese, Russa Biswas, and Harald Sack. "A Comprehensive Survey of Knowledge Graph Embeddings with Literals: Techniques and Applications." In: *Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG2019) Co-located with the 16th Extended Semantic Web Conference 2019 (ESWC 2019), Portoroz, Slovenia, June 2, 2019*. 2019, pp. 31–40. URL: http://ceur-ws.org/Vol-2377/paper_4.pdf.

[2] Denny Vrandečić and Markus Krötzsch. "Wikidata: A Free Collaborative Knowledge Base." In: (2014).

[3] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. "Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge." In: *ACM SIGMOD international conference on Management of data*. 2008.

[4] Thomas Pellissier Tanon, Gerhard Weikum, and Fabian M. Suchanek. "YAGO 4: A Reason-able Knowledge Base." In: *The Semantic Web - 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31-June 4, 2020, Proceedings*. Vol. 12123. Lecture Notes in Computer Science. Springer, 2020, pp. 583–596. DOI: 10.1007/978-3-030-49461-2_34. URL: https://doi.org/10.1007/978-3-030-49461-2_34.

[5] Antoine Bordes, Sumit Chopra, and Jason Weston. "Question Answering with Subgraph Embeddings." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 615–620. DOI: 10.3115/v1/D14-1067. URL: https://www.aclweb.org/anthology/D14-1067.

[6] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. "Collaborative Knowledge Base Embedding for Recommender Systems." In: *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: ACM, 2016, pp. 353–362. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939673. URL: http://doi.acm.org/10.1145/2939672.2939673.

[7] Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. "Connecting Language and Knowledge Bases with Embedding Models for Relation Extraction." In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 2013.

[8] Q. Wang, Z. Mao, B. Wang, and L. Guo. "Knowledge Graph Embedding: A Survey of Approaches and Applications." In: *TKDE* (2017).

[9] Jiahang Cao, Jinyuan Fang, Zaiqiao Meng, and Shangsong Liang. "Knowledge Graph Embedding: A Survey from the Perspective of Representation Spaces." In: *arXiv preprint arXiv:2211.03536* (2022).

[10] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. "Knowledge graphs." In: *ACM Computing Surveys (CSUR)* 54.4 (2021), pp. 1–37.

[11] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. "Learning Structured Embeddings of Knowledge Bases." In: *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*. AAAI'11. San Francisco, California: AAAI Press, 2011, pp. 301–306. URL: http://dl.acm.org/citation.cfm?id=2900423.2900470.

[12] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990. ISBN: 0716710455.

[13] HongYun Cai, Vincent Wenchen Zheng, and Kevin Chen-Chuan Chang. "A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications." In: *TKDE* (2018).

[14] Genet Asefa Gesese, Russa Biswas, Mehwish Alam, and Harald Sack. "A Survey on Knowledge Graph Embeddings with Literals: Which Model Links Better Literal-Ly?" In: *Semantic Web* 12.4 (2021), 617–647. ISSN: 1570-0844. DOI: 10.3233/SW-200404. URL: https://doi.org/10.3233/SW-200404.

[15] Kristina Toutanova and Danqi Chen. "Observed versus latent features for knowledge base and text inference." In: *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*. Beijing, China: Association for Computational Linguistics, July 2015, pp. 57–66. DOI: 10.18653/v1/W15-4007. URL: https://www.aclweb.org/anthology/W15-4007.

[16] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. "Convolutional 2d knowledge graph embeddings." In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[17] Genet Asefa Gesese, Mehwish Alam, and Harald Sack. "LiterallyWikidata - A Benchmark for Knowledge Graph Completion Using Literals." In: *The Semantic Web – ISWC 2021*. Cham: Springer International Publishing, 2021, pp. 511–527. ISBN: 978-3-030-88361-4.

[18] Daniel Daza, Michael Cochez, and Paul T. Groth. "Inductive Entity Representations from Text via Link Prediction." In: *Proceedings of the Web Conference 2021* (2021).

[19] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhiyuan Liu, Juan-Zi Li, and Jian Tang. "KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation." In: *Transactions of the Association for Computational Linguistics* 9 (2021), pp. 176–194.

[20] Aditya Grover and Jure Leskovec. "Node2vec: Scalable Feature Learning for Networks." In: KDD '16. San Francisco, California, USA: Association for Computing Machinery, 2016, 855–864. ISBN: 9781450342322. DOI: 10.1145/2939672.2939754. URL: https://doi.org/10.1145/2939672.2939754.

[21] Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. "Unsupervised Machine Translation Using Monolingual Corpora Only." In: *arXiv preprint arXiv:1711.00043* (2017).

[22] Leonhard Euler. "Solutio problematis ad geometriam situs pertinentis." In: *Commentarii academiae scientiarum Petropolitanae* (1741), pp. 128–140.

[23] Edward W Schneider. "Course Modularization Applied: The Interface System and Its Implications For Sequence Control and Data Analysis." In: (1973).

[24] Heiko Paulheim. "Knowledge graph refinement: A survey of approaches and evaluation methods." In: *Semantic Web* 8.3 (2017), pp. 489–508. DOI: 10.3233/SW-160218. URL: https://doi.org/10.3233/SW-160218.

[25] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. "DBpedia–A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia." In: *Semantic Web* (2015).

[26] Farzaneh Mahdisoltani, Joanna Biega, and Fabian M Suchanek. "Yago3: A Knowledge Base from Multilingual Wikipedias." In: *CIDR*. 2013.

[27] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. "Yago: A Core of Semantic Knowledge." In: *Proceedings of the 16th International Conference on World Wide Web*. WWW '07. Banff, Alberta, Canada: ACM, 2007, pp. 697–706. ISBN: 978-1-59593-654-7. DOI: 10.1145/1242572.1242667. URL: http://doi.acm.org/10.1145/1242572.1242667.

[28] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. "YAGO: A Large Ontology from Wikipedia and WordNet." In: *Journal of Web Semantics* 6.3 (2008). World Wide Web Conference 2007Semantic Web Track, pp. 203–217. ISSN: 1570-8268. DOI: https://doi.org/10.1016/j.websem.2008.06.001. URL: https://www.sciencedirect.com/science/article/pii/S1570826808000437.

[29] George A Miller. "WordNet: a lexical database for English." In: *Communications of the ACM* 38.11 (1995), pp. 39–41.

[30] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity." In: *The bulletin of mathematical biophysics* 5 (1943), pp. 115–133.

[31] Kunihiko Fukushima. "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position." In: *Biological cybernetics* 36.4 (1980), pp. 193–202.

[32] David H. Hubel and Torsten N. Wiesel. "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex." In: *The Journal of Physiology* 160 (1962).

[33] Wei Wang and Jianxun Gang. "Application of convolutional neural network in natural language processing." In: *2018 international conference on information Systems and computer aided education (ICISCAE)*. IEEE. 2018, pp. 64–70.

[34] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches." In: *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 103–111. DOI: 10.3115/v1/W14-4012. URL: https://aclanthology.org/W14-4012.

[35] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. "Pre-trained models for natural language processing: A survey." In: *Science China Technological Sciences* 63.10 (2020), pp. 1872–1897.

[36] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. "Distributed Representations of Words and Phrases and Their Compositionality." In: NIPS'13. Lake Tahoe, Nevada: Curran Associates Inc., 2013, 3111–3119.

[37] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Gobal Vectors for Word Representation." In: *EMNLP*. 2014.

[38] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. "Enriching Word Vectors with Subword Information." In: *Transactions of the Association for Computational Linguistics* 5 (2017), pp. 135–146. ISSN: 2307-387X.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In: *Advances in neural information processing systems* 30 (2017).

[40] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. "Network representation learning: A survey." In: *IEEE transactions on Big Data* 6.1 (2018), pp. 3–28.

[41] Chaozhuo Li, Zhoujun Li, Senzhang Wang, Yang Yang, Xiaoming Zhang, and Jianshe Zhou. "Semi-Supervised Network Embedding." In: *Database Systems for Advanced Applications*. Ed. by Selçuk Candan, Lei Chen, Torben Bach Pedersen, Lijun Chang, and Wen Hua. Cham: Springer International Publishing, 2017, pp. 131–147.

[42] Suhang Wang, Jiliang Tang, Charu Aggarwal, and Huan Liu. "Linked Document Embedding for Classification." In: CIKM '16. Indianapolis, Indiana, USA: Association for Computing Machinery, 2016, 115–124. ISBN: 9781450340731. DOI: 10.1145/2983323.2983755. URL: https://doi.org/10.1145/2983323.2983755.

[43] Xiao Huang, Jundong Li, and Xia Hu. "Label Informed Attributed Network Embedding." In: WSDM '17. Cambridge, United Kingdom: Association for Computing Machinery, 2017, 731–739. ISBN: 9781450346757. DOI: 10.1145/3018661.3018667. URL: https://doi.org/10.1145/3018661.3018667.

[44] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. "DeepWalk: Online Learning of Social Representations." In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '14. New York, New York, USA: Association for Computing Machinery, 2014, 701–710. ISBN: 9781450329569. DOI: 10.1145/2623330.2623732. URL: https://doi.org/10.1145/2623330.2623732.

[45] Joshua Tenenbaum, Vin Silva, and John Langford. "A Global Geometric Framework for Nonlinear Dimensionality Reduction." In: *Science (New York, N.Y.)* 290 (Jan. 2001), pp. 2319–23. DOI: 10.1126/science.290.5500.2319.

[46] Tomas Mikolov, Kai Chen, G.s Corrado, and Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space." In: *Proceedings of Workshop at ICLR* 2013 (Jan. 2013).

[47] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. "A Survey on Knowledge Graphs: Representation, Acquisition, and Applications." In: *IEEE Transactions on Neural Networks and Learning Systems* 33.2 (2022), pp. 494–514. DOI: 10.1109/TNNLS.2021.3070843.

[48] Takuma Ebisu and Ryutaro Ichise. "TorusE: Knowledge Graph Embedding on a Lie Group." In: AAAI'18/IAAI'18/EAAI'18. New Orleans, Louisiana, USA: AAAI Press, 2018. ISBN: 978-1-57735-800-8.

[49] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. "Translating Embeddings for Modeling Multi-Relational Data." In: *NIPS*. 2013.

[50] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. "Embedding Entities and Relations for Learning and Inference in Knowledge Bases." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015. URL: http://arxiv.org/abs/1412.6575.

[51] Ni Lao and William Cohen. "Relational Retrieval Using a Combination of Path-Constrained Random Walks." In: *Machine Learning* 81 (Oct. 2010), pp. 53–67. DOI: 10.1007/s10994-010-5205-8.

[52] Christian Meilicke, Manuel Fink, Yanjie Wang, Daniel Ruffinelli, Rainer Gemulla, and Heiner Stuckenschmidt. "Fine-Grained Evaluation of Rule- and Embedding-Based Systems for Knowledge Graph Completion." In: *International Workshop on the Semantic Web*. 2018.

[53] Mehdi Ali, Max Berrendorf, Charles Hoyt, Laurent Vermue, Mikhail Galkin, Sahand Sharifzadeh, Asja Fischer, Volker Tresp, and Jens Lehmann. *Bringing Light Into the Dark: A Large-scale Evaluation of Knowledge Graph Embedding Models Under a Unified Framework*. June 2020.

[54] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. *Knowledge Graph Embedding by Translating on Hyperplanes*. 2014. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8531.

[55] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. *Learning Entity and Relation Embeddings for Knowledge Graph Completion*. 2015. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9571.

[56] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. "Knowledge Graph Embedding via Dynamic Mapping Matrix." In: Jan. 2015, pp. 687–696. DOI: 10.3115/v1/P15-1067.

[57] Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. *Knowledge Graph Completion with Adaptive Sparse Transfer Matrix*. 2016. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11982.

[58] Yantao Jia, Yuanzhuo Wang, Hailun Lin, Xiaolong Jin, and Xueqi Cheng. "Locally Adaptive Translation for Knowledge Graph Embedding." In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI'16. Phoenix, Arizona: AAAI Press, 2016, pp. 992–998. URL: http://dl.acm.org/citation.cfm?id=3015812.3015960.

[59] Yanrong Wu and Zhichun Wang. "Knowledge Graph Embedding with Numeric Attributes of Entities." In: *Rep4NLP@ACL*. 2018.

[60] Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. "Representation Learning of Knowledge Graphs with Entity Descriptions." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 30.1 (2016). URL: https://ojs.aaai.org/index.php/AAAI/article/view/10329.

[61] Ruobing Xie, Zhiyuan Liu, Tat-Seng Chua, Huan-Bo Luan, and Maosong Sun. "Image-embodied Knowledge Representation Learning." In: *IJCAI*. 2017.

[62] Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan, and Zheng Chen. "Aligning knowledge and text embeddings by entity descriptions." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2015, pp. 267–272.

[63] Jiacheng Xu, Xipeng Qiu, Kan Chen, and Xuanjing Huang. "Knowledge Graph Representation with Jointly Structural and Textual Encoding." In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. 2017, pp. 1318–1324. DOI: 10.24963/ijcai.2017/183. URL: https://doi.org/10.24963/ijcai.2017/183.

[64] Han Xiao, Minlie Huang, Lian Meng, and Xiaoyan Zhu. "SSP: semantic space projection for knowledge graph embedding with text descriptions." In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[65] Muhao Chen, Yingtao Tian, Kai-Wei Chang, Steven Skiena, and Carlo Zaniolo. "Co-training Embeddings of Knowledge Graphs and Entity Descriptions for Cross-Lingual Entity Alignment." In: *arXiv preprint arXiv:1806.06478* (2018).

[66] Bayu Distiawan Trsedya, Jianzhong Qi, and Rui Zhang. "Entity Alignment between Knowledge Graphs Using Attribute Embeddings." In: *AAAI*. 2019.

[67] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. "A Three-Way Model for Collective Learning on Multi-Relational Data." In: *ICML*. 2011.

[68] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. "Holographic Embeddings of Knowledge Graphs." In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI'16. Phoenix, Arizona: AAAI Press, 2016, pp. 1955–1961. URL: http://dl.acm.org/citation.cfm?id=3016100.3016172.

[69] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. "Complex Embeddings for Simple Link Prediction." In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML'16. New York, NY, USA: JMLR.org, 2016, pp. 2071–2080. URL: http://dl.acm.org/citation.cfm?id=3045390.3045609.

[70] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. "A semantic matching energy function for learning with multi-relational data." In: *Machine Learning* 94.2 (2014), pp. 233–259. ISSN: 1573-0565. DOI: 10.1007/s10994-013-5363-6. URL: https://doi.org/10.1007/s10994-013-5363-6.

[71] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. "Reasoning With Neural Tensor Networks for Knowledge Base Completion." In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger. Curran Associates, Inc., 2013, pp. 926–934.

[72] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. "Knowledge Vault: A Web-scale Approach to Probabilistic Knowledge Fusion." In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '14. New York, New York, USA: ACM, 2014, pp. 601–610. ISBN: 978-1-4503-2956-9. DOI: 10.1145/2623330.2623623. URL: http://doi.acm.org/10.1145/2623330.2623623.

[73] Agustinus Kristiadi, Mohammad Asif Khan, Denis Lukovnikov, Jens Lehmann, and Asja Fischer. “Incorporating Literals into Knowledge Graph Embeddings.” In: *ISWC2019*. 2019.

[74] Pouya Pezeshkpour, Liyan Chen, and Sameer Singh. “Embedding Multimodal Relational Data for Knowledge Base Completion.” In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2018, pp. 3208–3218.

[75] Yi Tay, Anh Tuan Luu, Minh C. Phan, and Siu Cheung Hui. “Multi-task Neural Network for Non-discrete Attribute Prediction in Knowledge Graphs.” In: *CoRR* (2017).

[76] Michael Cochez, Martina Garofalo, Jérôme Lenßen, and Maria Angela Pellegrino. “A First Experiment on Including Text Literals in KGloVe.” In: *arXiv preprint arXiv:1807.11761* (2018).

[77] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. “Factorizing Yago: Scalable Machine Learning for Linked Data.” In: *Proceedings of the 21st international conference on World Wide Web*. ACM. 2012.

[78] G. de Assis Costa and J. M. P. de Oliveira. “Towards Exploring Literals to Enrich Data Linking in Knowledge Graphs.” In: *AIKE*. 2018.

[79] Shu Guo, Quan Wang, Bin Wang, Lihong Wang, and Li Guo. “SSE: Semantically Smooth Embedding for Knowledge Graphs.” In: *IEEE Transactions on Knowledge and Data Engineering* PP (Dec. 2016), pp. 1–1. DOI: 10.1109/TKDE.2016.2638425.

[80] Ruobing Xie, Zhiyuan Liu, and Maosong Sun. “Representation Learning of Knowledge Graphs with Hierarchical Types.” In: *IJCAI*. 2016.

[81] Denis Krompaβ, Stephan Baier, and Volker Tresp. “Type-Constrained Representation Learning in Knowledge Graphs.” In: *Proceedings of the 14th International Conference on The Semantic Web - ISWC 2015 - Volume 9366*. Berlin, Heidelberg: Springer-Verlag, 2015, pp. 640–655. ISBN: 978-3-319-25006-9.

[82] Quan Wang, Bin Wang, and Li Guo. “Knowledge Base Completion Using Embeddings and Rules.” In: *Proceedings of the 24th International Conference on Artificial Intelligence*. IJCAI'15. Buenos Aires, Argentina: AAAI Press, 2015, pp. 1859–1865. ISBN: 978-1-57735-738-4. URL: http://dl.acm.org/citation.cfm?id=2832415.2832507.

[83] Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. “Typed Tensor Decomposition of Knowledge Bases for Relation Extraction.” In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1568–1579. DOI: 10.3115/v1/D14-1165. URL: https://www.aclweb.org/anthology/D14-1165.

[84] Zhiting Hu, Poyao Huang, Yuntian Deng, Yingkai Gao, and Eric Xing. "Entity Hierarchy Embedding." 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)*. Beijing, China: Association for Computational Linguistics, July 2015, pp. 1292–1300. DOI: 10.3115/v1/P15-1125. URL: https://www.aclweb.org/anthology/P15-1125.

[85] Yankai Lin, Zhiyuan Liu, and Maosong Sun. "Modeling Relation Paths for Representation Learning of Knowledge Bases." In: *EMNLP*. 2015.

[86] Kelvin Guu, John Miller, and Percy Liang. "Traversing Knowledge Graphs in Vector Space." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 318–327. DOI: 10.18653/v1/D15-1038. URL: https://www.aclweb.org/anthology/D15-1038.

[87] Alberto García-Durán, Antoine Bordes, and Nicolas Usunier. "Composing Relationships with Translations." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 286–290. DOI: 10.18653/v1/D15-1034. URL: https://www.aclweb.org/anthology/D15-1034.

[88] Arvind Neelakantan, Benjamin Roth, and Andrew Mccallum. "Compositional Vector Space Models for Knowledge Base Completion." In: 1 (Apr. 2015). DOI: 10.3115/v1/P15-1016.

[89] Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew Mccallum. "Chains of Reasoning over Entities, Relations, and Text using Recurrent Neural Networks." In: Jan. 2017, pp. 132–141. DOI: 10.18653/v1/E17-1013.

[90] Yuanfei Luo, Quan Wang, Bin Wang, and Li Guo. "Context-Dependent Knowledge Graph Embedding." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 1656–1661. DOI: 10.18653/v1/D15-1191. URL: https://www.aclweb.org/anthology/D15-1191.

[91] Alberto García-Durán and Mathias Niepert. "KBlrn: End-to-End Learning of Knowledge Base Representations with Latent, Relational, and Numerical Features." In: *UAI*. 2018.

[92] Zhuoyu Wei, Jun Zhao, Kang Liu, Zhenyu Qi, Zhengya Sun, and Guanhua Tian. "Large-scale Knowledge Base Completion: Inferring via Grounding Network Sampling over Selected Instances." In: *CIKM*. 2015.

[93] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. "Jointly Embedding Knowledge Graphs and Logical Rules." In: Jan. 2016, pp. 192–202. DOI: 10.18653/v1/D16-1019.

[94] Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. "Injecting Logical Background Knowledge into Embeddings for Relation Extraction." In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado: Association for Computational Linguistics, 2015, pp. 1119–1129. DOI: 10.3115/v1/N15-1118. URL: https://www.aclweb.org/anthology/N15-1118.

[95] Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Sujian Li, Baobao Chang, and Zhifang Sui. "Encoding Temporal Information for Time-Aware Link Prediction." In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2350–2354. DOI: 10.18653/v1/D16-1260. URL: https://www.aclweb.org/anthology/D16-1260.

[96] C. Esteban, V. Tresp, Y. Yang, S. Baier, and D. Krompaß. "Predicting the co-evolution of event and Knowledge Graphs." In: *2016 19th International Conference on Information Fusion (FUSION)*. 2016, pp. 98–105.

[97] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. "Know-evolve: Deep Temporal Reasoning for Dynamic Knowledge Graphs." In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 3462–3471. URL: http://dl.acm.org/citation.cfm?id=3305890.3306039.

[98] Jun Feng, Minlie Huang, Yang Yang, and Xiaoyan Zhu. "GAKE: Graph Aware Knowledge Embedding." In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, Dec. 2016, pp. 641–651. URL: https://www.aclweb.org/anthology/C16-1062.

[99] Xueyan Jiang, Volker Tresp, Yi Huang, and Maximilian Nickel. "Link Prediction in Multi-relational Graphs Using Additive Models." In: *Proceedings of the 2012 International Conference on Semantic Technologies Meet Recommender Systems &#38; Big Data - Volume 919*. SeRSy'12. Boston: CEUR-WS.org, 2012, pp. 1–12. URL: http://dl.acm.org/citation.cfm?id=2887638.2887639.

[100] Hatem Mousselly-Sergieh, Teresa Botschen, Iryna Gurevych, and Stefan Roth. "A multimodal translation-based approach for knowledge graph representation learning." In: *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*. 2018, pp. 225–234.

[101] Russa Biswas, Harald Sack, and Mehwish Alam. "MADLINK: Attentive multihop and entity descriptions for link prediction in knowledge graphs." In: *Semantic Web* (2022).

[102] Palash Goyal and Emilio Ferrara. "Graph Embedding Techniques, Applications, and Performance: A Survey." In: *Knowl.-Based Syst.* (2018).

[103] Yankai Lin, Zhiyuan Liu, and Maosong Sun. "Knowledge Representation Learning with Entities, Attributes and Relations." In: *ethnicity* (2016).

[104] Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. "You CAN Teach an Old Dog New Tricks! On Training Knowledge Graph Embeddings." In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=BkxSmlBFvr.

[105] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. "Knowledge graph and text jointly embedding." In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1591–1601.

[106] Michael Cochez, Petar Ristoski, Simone Paolo Ponzetto, and Heiko Paulheim. "Global RDF Vector Space Embeddings." In: *International Semantic Web Conference*. Springer. 2017.

[107] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. "Sequence to Sequence Learning with Neural Networks." In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. Montreal, Canada: MIT Press, 2014, 3104–3112.

[108] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." In: Jan. 2019, pp. 3973–3983. DOI: 10.18653/v1/D19-1410.

[109] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M Suchanek. "Fast Rule Mining in Ontological Knowledge Bases with AMIE+." In: *VLDB* (2015).

[110] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[111] G. de Assis Costa and J. M. P. de Oliveira. "A Blocking Scheme for Entity Resolution in the Semantic Web." In: *AINA*. 2016.

[112] Moritz Blum, Basil Ell, and Philipp Cimiano. "Exploring the impact of literal transformations within Knowledge Graphs for Link Prediction." In: *The 11th International Joint Conference on Knowledge Graphs (IJCKG)*. 2022.

[113] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." In: *International Conference on Learning Representations (ICLR)*. 2015.

[114] Matthew Francis-Landau, Greg Durrett, and Dan Klein. "Capturing semantic similarity for entity linking with convolutional neural networks." In: *arXiv preprint arXiv:1604.00734* (2016).

[115] Jake Zhao, Yoon Kim, Kelly Zhang, Alexander M Rush, and Yann LeCun. "Adversarially regularized autoencoders." In: *arXiv preprint arXiv:1706.04223* (2017).

[116] David Berthelot, Thomas Schumm, and Luke Metz. "Began: Boundary equilibrium generative adversarial networks." In: *arXiv preprint arXiv:1703.10717* (2017).

[117] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. "Image-to-image translation with conditional adversarial networks." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1125–1134.

[118] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. "YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia." In: *Artificial Intelligence* 194 (2013). Artificial Intelligence, Wikipedia and Semi-Structured Resources, pp. 28–61. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2012.06.001. URL: https://www.sciencedirect.com/science/article/pii/S0004370212000719.

[119] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, and Tom M. Mitchell. "Toward an Architecture for Never-Ending Language Learning." In: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*. AAAI Press, 2010.

[120] Yanrong Wu and Zhichun Wang. "Knowledge Graph Embedding with Numeric Attributes of Entities." In: *Proceedings of The Third Workshop on Representation Learning for NLP*. Association for Computational Linguistics, 2018, pp. 132–136.

[121] Tara Safavi and Danai Koutra. "CoDEx: A Comprehensive Knowledge Graph Completion Benchmark." In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Nov. 2020.

[122] 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 ACM SIGMOD International Conference on Management of Data*. 2020. ISBN: 9781450367356.

[123] Tara Safavi, Danai Koutra, and Edgar Meij. "Improving the Utility of Knowledge Graph Embeddings with Calibration." In: *arXiv preprint arXiv:2004.01168* (2020).

[124] Wenhan Xiong, Thien Hoang, and William Yang Wang. "DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning." In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2017.

[125] F. Maxwell Harper and Joseph A. Konstan. "The MovieLens Datasets: History and Context." In: *ACM Trans. Interact. Intell. Syst.* 5.4 (Dec. 2015). ISSN: 2160-6455.

[126] Lucas van Berkel and Victor de Boer. "kgbench: A Collection of Knowledge Graph Datasets for Evaluating Relational and Multimodal Machine Learning." In: *ESWC*. 2021.

[127] Haseeb Shah, Johannes Villmow, A. Ulges, U. Schwanecke, and F. Shafait. "An Open-World Extension to Knowledge Graph Completion Models." In: *AAAI*. 2019.

[128] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. "Reasoning with Neural Tensor Networks for Knowledge Base Completion." In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*. 2013.

[129] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. "Knowledge Graph Embedding by Translating on Hyperplanes." In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. AAAI'14. Québec City, Québec, Canada: AAAI Press, 2014, 1112–1119.

[130] Yankai Lin, Zhiyuan Liu, and Maosong Sun. "Knowledge Representation Learning with Entities, Attributes and Relations." In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. IJCAI'16. New York, New York, USA: AAAI Press, 2016, 2866–2872. ISBN: 9781577357704.

[131] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. "Knowledge Graph Embedding with Iterative Guidance from Soft Rules." In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[132] T. Mitchell et al. "Never-Ending Learning." In: *Commun. ACM* 61.5 (Apr. 2018), 103–115. ISSN: 0001-0782.

[133] Stanley Kok and Pedro Domingos. "Statistical Predicate Invention." In: *Proceedings of the 24th International Conference on Machine Learning*. Association for Computing Machinery, 2007.

[134] A. McCray. "An Upper-Level Ontology for the Biomedical Domain." In: *Comparative and Functional Genomics* 4 (2003), pp. 80 –84.

[135] Charles Kemp, Joshua B. Tenenbaum, Thomas L. Griffiths, Takeshi Yamada, and Naonori Ueda. "Learning Systems of Concepts with an Infinite Relational Model." In: *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*. AAAI'06. Boston, Massachusetts: AAAI Press, 2006, 381–388. ISBN: 9781577352815.

[136] Rudolph J. Rummel. "Dimensionality of Nations Project: Attributes of Nations and Behavior of Nation Dyads, 1950-1965. [distributor], 1992-02-16." In: ().

[137] Guillaume Bouchard, Sameer Singh, and T. Trouillon. "On Approximate Reasoning Capabilities of Low-Rank Vector Spaces." In: *AAAI Spring Symposia*. 2015.

[138] Alberto García-Durán, Antoine Bordes, and Nicolas Usunier. "Effective Blending of Two and Three-Way Interactions for Modeling Multi-Relational Data." In: *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part I*. ECMLPKDD'14. Nancy, France, 2014, 434–449.

[139] Geoffrey E Hinton et al. "Learning distributed representations of concepts." In: *Proceedings of the eighth annual conference of the cognitive science society*. Vol. 1. Amherst, MA. 1986, p. 12.

[140] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. "RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space." In: *International Conference on Learning Representations*. 2019.

[141] 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*. WWW '13. Rio de Janeiro, Brazil: Association for Computing Machinery, 2013, 413–422. ISBN: 9781450320351. DOI: 10.1145/2488388.2488425. URL: https://doi.org/10.1145/2488388.2488425.

[142] Fan Yang, Zhilin Yang, and William W. Cohen. "Differentiable Learning of Logical Rules for Knowledge Base Reasoning." In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, 2316–2325. ISBN: 9781510860964.

[143] Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. "DRUM: End-to-End Differentiable Rule Mining on Knowledge Graphs." In: Red Hook, NY, USA: Curran Associates Inc., 2019.

[144] William L. Hamilton, Rex Ying, and Jure Leskovec. "Inductive Representation Learning on Large Graphs." In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, 1025–1035. ISBN: 9781510860964.

[145] Aleksandar Bojchevski and Stephan Günnemann. "Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking." In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL: https://openreview.net/forum?id=r1ZdKJ-0W.

[146] Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. "Knowledge Transfer for Out-of-Knowledge-Base Entities : A Graph Neural Network Approach." In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. 2017, pp. 1802–1808. DOI: 10.24963/ijcai.2017/250. URL: https://doi.org/10.24963/ijcai.2017/250.

[147] Peifeng Wang, Jialong Han, Chenliang Li, and Rong Pan. "Logic Attention Based Neighborhood Aggregation for Inductive Knowledge Graph Embedding." In: *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI'19/IAAI'19/EAAI'19. Honolulu, Hawaii, USA: AAAI Press, 2019. ISBN: 978-1-57735-809-1. DOI: 10.1609/aaai.v33i01.33017152. URL: https://doi.org/10.1609/aaai.v33i01.33017152.

[148] Komal Teru, Etienne Denis, and Will Hamilton. “Inductive Relation Prediction by Subgraph Reasoning.” In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 9448–9457. URL: https://proceedings.mlr.press/v119/teru20a.html.

[149] Mehdi Ali, Max Berrendorf, Mikhail Galkin, Veronika Thost, Tengfei Ma, Volker Tresp, and Jens Lehmann. “Improving Inductive Link Prediction Using Hyper-Relational Facts.” In: *SEMWEB*. 2021.

[150] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. “A Comprehensive Survey on Graph Neural Networks.” In: *IEEE Transactions on Neural Networks and Learning Systems* 32 (2019), pp. 4–24.

[151] Liang Yao, Chengsheng Mao, and Yuan Luo. “KG-BERT: BERT for knowledge graph completion.” In: *arXiv preprint arXiv:1909.03193* (2019).

[152] Louis Clouâtre, Philippe Trempe, Amal Zouaq, and Sarath Chandar. “MLMLM: Link Prediction with Mean Likelihood Masked Language Model.” In: *Findings of ACL/IJCNLP*. Vol. ACL/IJCNLP 2021. Findings of ACL. Association for Computational Linguistics, 2021, pp. 4321–4331.

[153] 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.

[154] Genet Asefa Gesese, Mehwish Alam, and Harald Sack. “RAILD: Towards Leveraging Relation Features for Inductive Link Prediction In Knowledge Graphs.” In: *IJCKG*. 2022.

[155] Xin Luna Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. “Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion.” In: *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’14, New York, NY, USA - August 24 - 27, 2014*. Evgeniy Gabrilovich Wilko Horn Ni Lao Kevin Murphy Thomas Strohmann Shaohua Sun Wei Zhang Geremy Heitz. 2014, pp. 601–610. URL: http://www.cs.cmu.edu/~nlao/publication/2014.kdd.pdf.

[156] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. “Attention is All you Need.” In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[157] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.” In: *CoRR* abs/1910.01108 (2019). arXiv: 1910.01108. URL: http://arxiv.org/abs/1910.01108.

[158] Genet Asefa Gesese, Mehwish Alam, and Harald Sack. *LiterallyWikidata - A Benchmark for Knowledge Graph Completion using Literals*. Apr. 2021. DOI: 10.5281/zenodo.4701190. URL: https://doi.org/10.5281/zenodo.4701190.

[159] Vladimir Batagelj and Matjaž Zaveršnik. "Fast algorithms for determining (generalized) core groups in social networks." In: *Advances in Data Analysis and Classification* 5.2 (2011), pp. 129–145.

[160] Mingyang Li, Neng Gao, Chenyang Tu, Jia Peng, and Min Li. "Incorporating Attributes Semantics into Knowledge Graph Embeddings." In: *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. 2021, pp. 620–625. DOI: 10.1109/CSCWD49262.2021.9437876.

[161] Farshad Bakhshandegan Moghaddam, Carsten Draschner, Jens Lehmann, and Hajira Jabeen. "Literal2feature: An automatic scalable rdf graph feature extractor." In: *Further with Knowledge Graphs*. IOS Press, 2021, pp. 74–88.

[162] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[163] Genet Asefa Gesese, Mehwish Alam, and Harald Sack. "Semantic Entity Enrichment by Leveraging Multilingual Descriptions for Link Prediction." In: *DL4KG workshop co-located with ESWC*. 2020.

[164] Stephan Gouws, Yoshua Bengio, and Greg Corrado. "Bilbowa: Fast bilingual distributed representations without word alignments." In: *ICML*. 2015.

[165] Xu Han, Shulin Cao, Lv Xin, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. "OpenKE: An Open Toolkit for Knowledge Embedding." In: *Proceedings of EMNLP*. 2018.

[166] Cristian Santini, Genet Asefa Gesese, Silvio Peroni, Aldo Gangemi, Harald Sack, and Mehwish Alam. "A Knowledge Graph Embeddings Based Approach for Author Name Disambiguation Using Literals." In: *Scientometrics* 127.8 (2022), 4887–4912. ISSN: 0138-9130. DOI: 10.1007/s11192-022-04426-2. URL: https://doi.org/10.1007/s11192-022-04426-2.

DECLARATION

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, 02.08.2023

M.Sc. GENET ASEFA GESESE