DEPARTMENT OF INFORMATICS

UNIVERSITY OF FRIBOURG (SWITZERLAND)

Extending Knowledge Graph Embeddings for Data Imputation

THESIS

Presented to the Faculty of Science and Medicine of the University of Fribourg (Switzerland)
in consideration for the award of the academic grade of
Doctor of Philosophy in Computer Science

by

PAOLO ROSSO

from

ITALY

Thesis No: 2259
UniPrint
2021

Accepted by the Faculty of Science and Medicine of the University of Fribourg (Switzerland) upon the recommendation of:

Prof. Dr. Denis Lalanne, University of Fribourg (Switzerland), president of the jury.
Prof. Dr. Philippe Cudré-Mauroux, University of Fribourg (Switzerland), thesis supervisor.
Prof. Dr. Dingqi Yang, University of Macau (Macau), thesis co-supervisor.
Dr. Bryan Perozzi, Google (NYC, United States of America), external examiner.
Prof. Dr. Jie Tang, Tsinghua University (Beijing, China), external examiner.

Fribourg, May 28, 2021

Thesis supervisor	*Dean*
Prof. Dr. Philippe Cudré-Mauroux	Prof. Dr. Gregor Rainer

Title: Extending Knowledge Graph Embeddings for Data Imputation

I, Paolo Rosso, declare that I have authored this thesis independently, without illicit help, that I have not used any other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Signed: ______________________________

Date: ______________________________

Technology is just a tool. In terms of getting the kids working together and motivating them, the teacher is the most important.

Bill Gates

To sing opera, one needs two things: the voice and the passion - and above all, the passion.

Andrea Bocelli

Persistence and resilience only come from having been given the chance to work through difficult problems.

Gever Tulley

To my father…

Acknowledgements

I would first like to thank my family for the continuous support they have given me on this journey; I could not have done it without them.

Thanks to my supervisor Prof. Dr. Philippe Cudré-Mauroux for his incredible guidance, support and insightful feedback. You provided me with the tools that I needed to choose the right direction and successfully complete my dissertation. Without you, this would have not been possible.

Thanks also to my former colleague Prof. Dr. Dingqi Yang. I am extremely grateful for our friendly meetings and your personal support. Without your help and wise guidance this doctoral thesis would have not been the same.

I would also like to thank all the members in the eXascale Infolab group who provided stimulating discussions as well as many lunches, wonderful coffee breaks and fun activities.

From the bottom of my heart I would like to express my gratitude to my friends in Cuneo and Torino. You have provided me happy distraction to rest my mind outside of my research.

Last but not least, to Thesi. Thanks for your love and support. You are always there in time of need.

Fribourg, 14 Mars 2021

Abstract

With the advancement of Big Data and Natural Language Processing (NLP) technologies, extensive research into Knowledge Graphs (KGs) has been conducted. In a typical KG, such as Wikidata, entities are connected via relations. A popular approach to represent facts in KGs is to define them as triplets $(head, relation, tail)$. For example, the fact *Bern*, *capitalOf*, *Switzerland*, is composed of two entities, *Bern* and *Switzerland*, connected by the relation *capitalOf*. Although KGs are effective in representing structured data, they cannot be used to train modern Machine Learning models which often require numerical input. To tackle this issue, Knowledge Graph embeddings have been proposed. In our context, KG embeddings aim to project entities and relations from a KG into a low-dimensional and continuous vector space. The main benefit of such a representation is that the resulting vectors can be subsequently used as input to Machine Learning pipelines. In this thesis, we first introduce popular Knowledge Graphs, as well as typical KG embedding models. After providing an overview of the applications and problems that can be tackled with KG embeddings, we present our own contributions to this research field. Specifically, we first propose a novel approach, called JOINER, to jointly learn KG embeddings from text and a Knowledge Graph by taking advantage of both large-scale unstructured content (text) and high-quality structured data (the Knowledge Graph). JOINER not only preserves co-occurrences between words in a text corpus and relations between entities in a Knowledge Graph, it also provides the flexibility to control the amount of information shared between the two data sources via regularization. We conduct a thorough evaluation of JOINER on three evaluation tasks (analogical reasoning, link prediction and relation extraction) using three different corpora, showing significant improvement on most tasks. Next, we present a new KG embeddings model, called HINGE, able to learn hyper-relational facts from KGs, which are facts containing not only a base triplet $(head, relation, tail)$ but also associated key-value pairs. HINGE captures not only the primary structural information of the KG encoded in the triplets, but also the correlation between each triplet and its associated key-value pairs. Our extensive evaluation shows the superiority of HINGE on various link prediction tasks over KGs, outperforming not only the KG embedding methods learning from triplets only (by 0.81-41.45%), but also the methods learning from hyper-relational facts using an n-ary representation (by 13.2-84.1%). Additionally, we propose an end-to-end solution called RETA in order to tackle instance completion problems by suggesting relation-tail pairs given a head entity. RETA consists of two components: RETA-Filter and RETA-Grader. More precisely, RETA-Filter first generates a filtered list of candidates by extracting and leveraging the schema of a KG; RETA-Grader then evaluates and ranks the candidate pairs considering the plausibility of both the candidate triplet and its corresponding schema using a newly-designed KG embedding model. We evaluate our methods against a sizable collection of state-of-the-art techniques on three real-world KG datasets. Results show that our RETA-Filter generates of high-quality candidate r-t pairs, outperforming the best baseline techniques while reducing by 10.61%-84.75% the candidate pool size under the same candidate quality guarantees. Moreover, our RETA-Grader also significantly outperforms state-of-the-art link prediction techniques on the instance completion task by 16.25%-65.92% across different datasets. Finally, we address research questions raised in this thesis related to a number of Knowledge Graph embedding methods presented in the next sections. Additionally, we summarize the series of contributions we made in some of the core tasks tackled by Knowledge Graph embedding methods and conclude our thesis by discussing how to extend our proposed works.

Key words: Knowledge Graph embeddings, Word embeddings, regularization, Hyper-relation, Link prediction, Entity Types, Instance completion

Résumé

Avec l'avancement des technologies de Big Data et de traitement du langage naturel (NLP), des recherches approfondies sur les graphes de connaissances (ou *Knowledge Graphs* KGs) ont été menées. Dans un KG typique, tel que Wikidata, les entités sont reliées par des relations. Une approche populaire pour représenter les faits dans les KGs consiste à les définir comme des triplets $(head, relation, tail)$. Par exemple, le fait *Bern, capitalOf, Switzerland*, est composé de deux entités, *Bern* et *Switzerland*, reliées par la relation *capitalOf.* Bien que les graphes de connaissances soient efficaces pour représenter des données structurées, ils ne peuvent pas être utilisés pour former des modèles modernes d'apprentissage automatique qui nécessitent souvent des données numériques. Pour résoudre ce problème, des incorporations (ou *embeddings*) de graphes de connaissances ont été proposées. Dans notre contexte, les incorporations de KG visent à projeter les entités et les relations d'un KG dans un espace vectoriel continu et de faible dimension. Le principal avantage d'une telle représentation est que les vecteurs résultants peuvent être utilisés ultérieurement comme entrée dans des pipelines d'apprentissage automatique. Dans cette thèse, nous présentons d'abord les graphes de connaissances les plus populaires, ainsi que les modèles typiques d'intégration de KG. Après avoir donné une vue d'ensemble des applications et des problèmes qui peuvent être abordés avec les encastrements de KG, nous présentons nos propres contributions à ce domaine de recherche. Plus précisément, nous proposons d'abord une nouvelle approche, appelée JOINER, pour apprendre conjointement les encastrements de KG à partir d'un texte et d'un graphe de connaissances en tirant parti à la fois du contenu non structuré à grande échelle (le texte) et des données structurées de haute qualité (le graphe de connaissances). JOINER préserve non seulement les cooccurrences entre les mots d'un corpus textuel et les relations entre les entités d'un graphe de connaissances, mais il offre également la possibilité de contrôler la quantité d'informations partagées entre les deux sources de données par le biais de la régularisation. Nous effectuons une évaluation approfondie de JOINER sur trois tâches d'évaluation (raisonnement analogique, prédiction de liens et extraction de relations) en utilisant trois corpus différents, montrant une amélioration significative pour la plupart des tâches. Ensuite, nous présentons un nouveau modèle d'intégration de KG, appelé HINGE, capable d'apprendre des faits hyper-relationnels à partir de KGs, qui sont des faits contenant non seulement un triplet de base $(head, relation, tail)$ mais aussi des paires clé-valeur associées. HINGE capture non seulement l'information structurelle primaire du KG encodée dans les triplets, mais aussi la corrélation entre chaque triplet et ses paires clé-valeur associées. Notre évaluation approfondie montre la supériorité de HINGE sur diverses tâches de prédiction de liens sur les KGs, surpassant non seulement les méthodes d'intégration de KG apprenant uniquement à partir de triplets (de 0,81 à 41,45%), mais aussi les méthodes apprenant à partir de faits hyperrelationnels utilisant une représentation n-aire (de 13,2 à 84,1%). De plus, nous proposons une solution de bout en bout appelée RETA afin de résoudre les problèmes de complétion d'instances en suggérant des paires relation-queue pour une entité de tête. RETA se compose de deux éléments : RETA-Filter et RETA-Grader. Plus précisément, RETA-Filter génère d'abord une liste filtrée de candidats en extrayant et en exploitant le schéma d'une KG; RETA-Grader évalue et classe ensuite les paires de candidats en tenant compte de la plausibilité du triplet candidat et de son schéma correspondant à l'aide d'un nouveau modèle d'intégration de KG. Nous évaluons nos méthodes par rapport à un ensemble important de techniques de pointe sur trois ensembles de données de KGs réels. Les résultats montrent que notre RETA-Filter génère des paires r-t de haute qualité, surpassant les meilleures techniques de base tout en réduisant de 10,61% à 84,75% la taille du pool de candidats avec les mêmes garanties de qualité. De plus, notre RETA-Grader surpasse également de manière significative les techniques de prédiction de liens de l'état de l'art pour des tâches de complétion d'instances de 16,25% à 65,92% suivant les jeux de données utilisés. Enfin, nous abordons les questions de recherche soulevées dans cette thèse en relation avec les méthodes d'intégration des graphes de connaissances présentées dans les sections suivantes. En outre, nous résumons la série de contributions que nous avons faites concernant les diverses tâches abordées et concluons cette thèse en discutant de la manière d'étendre les travaux proposés.

Mots clefs : Knowledge Graph embeddings, Word embeddings, regularization, Hyper-relation, Link prediction, Entity Types, Instance completion

Contents

List of Figures

List of Tables

1 Introduction

Artificial Intelligence (AI) and more specifically Machine Learning (ML) are two of today's most rapidly growing technical fields due to recent progress including new algorithms, the use of powerful hardware to power them, and the collection of big datasets [62]. AI was founded as an academic discipline in 1955 [81] and it has been defined as the science and engineering of making intelligent machines, especially intelligent computer programs [80]. With recent developments of AI, considerable research into Knowledge Graphs has been carried out [100, 41]. Knowledge Graphs are essentially multi-relational graphs composed of entities (nodes) and relations (edges) and encoding knowledge. Figure 1.1 shows a simple instance of a KG where entities in the real world are represented as nodes, which are connected by edges representing the relations between those entities. A fact in a KG, also known as a *triplet*, is represented by two entities connected by a relation. Often, the two entities composing a fact are also referred to as the *head* and *tail* entities, and a fact represented as a triplet h, r, t. Moreover, a triplet may also contain additionally information, stored as key-value pairs, for instance. Figure 1.2 shows an example of a fact in a KG. In this case, *Marie Curie* and *University of Paris* are the head and tail entities connected by the relation *educated at*. Moreover, the fact also contains extra information stored as key-value pairs, where *academic major* and *academic degree* are the keys and their values are *Physics* and *Master of Science*, respectively. Unlike relational databases where the information is bound under a rigid schema, the schema-less nature of KGs creates flexibility by not enforcing the representation of the data.

Knowledge Graphs have become a key asset powering a large range of applications, mainly in the area of Artificial Intelligence. For instance, KGs can be used to support decision making and to improve machine learning applications. Example of applications include question-answering, recommender systems, spam detection, etc. Although KGs are effective in representing structured data, their underlying symbolic nature usually makes them hard to manipulate because many Machine Learning algorithms expect numerical input data. To avoid this problem, a new research direction known as Knowledge Graph embeddings has

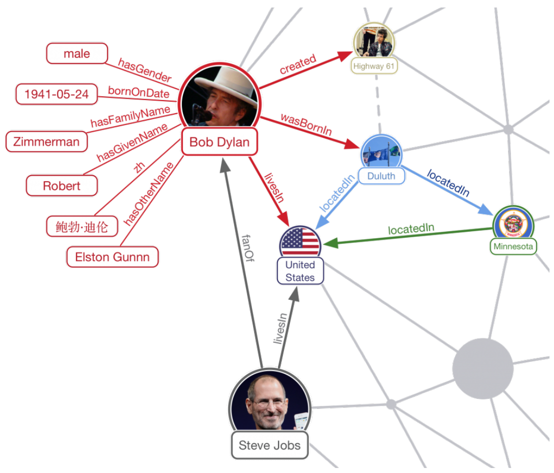


Figure 1.1 – A simple instance of a KG in which entities are connected via relations.

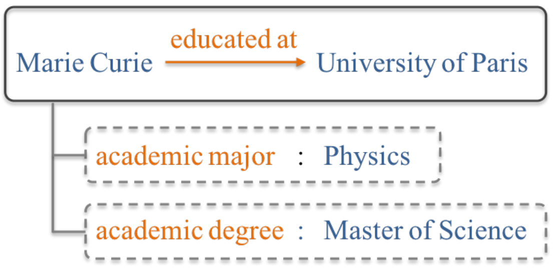


Figure 1.2 – Example of a fact in a KG. The fact is composed by the head and tail entities *Marie Curie* and *University of Paris*, respectively, connected by the relation *educated at*. Additionally, the fact contains also two sets of key-value pairs. The first key-value pair is represented by *academic major* and *Physics*, while the second set of key-value pairs is represented by *academic degree* and *Master of Science*.

been proposed, where the idea is to represent entities and relations in continuous vector spaces. The generated vectors can then be used to train Neural Networks while preserving the inherent structure of the KG. Additionally, the vectors of entities and relations can be used to tackle other problems, such as link prediction, entity resolution, or triple classification. Most of the existing techniques perform embedding tasks on the basis of observed facts in the KGs. More precisely, existing techniques first represent entities and relations in a continuous vector space, and define a scoring function to measure the plausibility of a fact. Often, these techniques represent the entities and relations in a KG in a low-dimensional vector space, and define an evaluation function to measure the plausibility of each fact triplet. Neural networks are used to learn the embeddings. The embeddings form the weights of the network which are adjusted to minimize loss on a task to be tackled. Additionally, further approaches also leverage other types of information, such as text, entity types, etc., in order to enhance the vectorial representation. Thanks to the low and dense feature space, embeddings are capable of preserving the structure and property information of the graph and can be used as a proxy when analyzing the entities and relations. In the last few years, KG embeddings have become a research hotspot and models have been proposed in that context. These models differ in three main ways: (i) how they define the representation form (triplet, property graphs, etc.), (ii) how they define the scoring function, (iii) how they optimize the ranking criterion that maximizes the global plausibility of the existing triplets. Additionally, Knowledge Graphs can be used as an applied technology to support specific applications in the industrial domain. For example, in the field of medicine, KGs are used to diagnose diseases and to analyze drugs [112, 4]. In the field of e-commerce, KGs containing information about products are built in order to accurately match the user's purchase intention and product candidate set [72, 139], and in public security KGs are employed to analyze the relations between entities and obtain clues [140]. A distributed representation technology can be utilized by KG embeddings in order to alleviate the problems of data sparsity and computational inefficiency [27], having three main advantages:

- By embedding entities and relations into a continuous low-dimensional vector space, the data sparsity problem can be effectively mitigated, which makes the calculation of semantic or inferential relations of entities extremely inaccurate.
- Compared to traditional one-hot representation, which is a representation of categorical variables as binary vectors, KG embeddings use a distributed representation method to improve the efficiency of semantic computing.
- The calculation between different types of information can be performed by defining a unified feature space to connect heterogeneous objects.

Despite recent advances, KG embeddings are still in their infancy and many problems on how to best learn such embeddings are still unsolved. In the following section, we look at some of these issues and define the research questions that we focus on in this thesis.

1.1 Research Questions

This thesis focuses on Knowledge Graph embeddings, which project entities and relations of a KG into a low-dimensional and continuous vector space while preserving the key structural properties of the KG. Such KG embeddings effectively retain the reasoning ability of the KG, and can thus efficiently support various downstream applications by facilitating the applications of classical vector-based machine learning techniques on KGs. A large number of successful KG embeddings models have been proposed [19, 131, 29], but most of them train the models only on the observed triplets. Thereupon, in these thesis work we focus on learning broader KG embedding models by leveraging additional information in addition to the Knowledge Graphs. Moreover, we also study the problem of learning more expressive KG embeddings by leveraging richer data structures like hyper-relational facts. Specifically, we aim at answering the following research questions in this thesis:

- **Q1:** How can we jointly learn embeddings from some text and a Knowledge Graph? How can we best benefit from both data sources simultaneously?
- **Q2:** How can we effectively learn Knowledge Graph embeddings from more expressive structures such as hyper-relational facts?
- **Q3:** How can we tackle instance completion tasks by leveraging expressive KG embeddings?

In the following chapters, we answer those questions in detail.

1.2 Summary of Contributions

Knowledge Graphs gained great popularity in the past years, and a number of research works proposed Machine Learning models able to represent entities and relations of a KG as vectors. Those vectors, also known as Embeddings, can be used as input to train Neural Networks and preserve the inherent structure of the Knowledge Graph as well as the reasoning ability over the graph. This thesis proposes a set of new algorithms to learn embeddings representing entities and relations from a Knowledge Graph. In the following, we describe the main contributions of this thesis and list the conference papers we have published along our research work.

- **Background:** in Chapter 2, we give an overview of the state of the art in Knowledge Graph embeddings. We group existing works in three categories: Translational distance models, Semantic matching models and Deep learning models. In addition, we describe the learning processes of the embeddings and the two main strategies to generate negative samples based on the Open World Assumption and on the Closed World Assumption. Finally, we describe typical applications of KG embedding and also applications going beyond KGs. Part of the work presented in Chapter 2 has been published as a survey in the following paper:

Paolo Rosso, Dingqi Yang, Philippe Cudré-Mauroux: Knowledge Graph Embeddings. Encyclopedia of Big Data Technologies 2019

- **Jointly learning embeddings from both text and Knowledge Graphs:** in Chapter 3, we aim at answering research question *Q1*, and in particular how to benefit from jointly learning embeddings from text and a KG. We present JOINER, a new method to jointly learn embeddings taking advantage of both large-scale unstructured content (text) and high-quality structured data (the Knowledge Graph). JOINER not only preserves the co-occurrence between words in a text corpus and relations between entities in a Knowledge Graph, but it also provides the flexibility to control the amount of information shared between the two data sources via regularization. Our extensive evaluation shows the superiority of JOINER across different evaluation tasks, including analogical reasoning, link prediction, and relation extraction, yielding up to 4.3% absolute improvement and 76% savings in learning time overhead. The work presented in Chapter 3 has been published in the following research paper:

  Paolo Rosso, Dingqi Yang, Philippe Cudré-Mauroux: Revisiting Text and Knowledge Graph Joint Embeddings: The Amount of Shared Information Matters! BigData 2019: 2465-2473

- **Learning KG embeddings for hyper-relational facts:** in Chapter 4, we aim at answering the second research question (*Q2*), and in particular how to learn KG embeddings from hyper-relation facts. We propose HINGE, a hyper-relational KG embedding model, which directly learns from hyper-relational facts in a KG. HINGE captures not only the primary structural information of the KG encoded in the triplets, but also the correlation between each triplet and its associated key-value pairs. Our extensive evaluation shows the superiority of HINGE on various link prediction tasks over KGs. In particular, HINGE consistently outperforms not only the KG embedding methods learning from triplets only (by 0.81-41.45% depending on the link prediction tasks and settings), but also the methods learning from hyper-relational facts using the n-ary representation (by 13.2-84.1%). The work presented in Chapter 4 has been published in the following paper:

  Paolo Rosso, Dingqi Yang, Philippe Cudré-Mauroux: Beyond Triplets: Hyper-Relational Knowledge Graph Embedding for Link Prediction. WWW 2020: 1885-1896

- **An end-to-end solution to tackle instance completion:** in Chapter 5, we aim at answering our last research question *Q3*, and in particular how to leverage the schema of a KG to tackle instance completion. We propose an end-to-end solution called RETA consisting of two components: RETA-Filter and RETA-Grader. More precisely, our RETA-Filter first generates candidate r-t pairs for a given h by extracting and leveraging the schema of a KG; our RETA-Grader then evaluates and ranks the candidate r-t pairs considering the plausibility of both the candidate triplet and its corresponding schema using a newly-designed KG embedding model. We evaluate our methods against a sizable collection of state-of-the-art techniques on three real-world KG datasets. Results show that our RETA-Filter generates of high-quality candidate r-t pairs, outperforming the best baseline techniques while reducing by 10.61%-84.75% the candidate size under the same candidate quality guarantees. Moreover, our RETA-Grader also significantly outperforms state-of-the-art link prediction techniques on the instance completion task by 16.25%-65.92% across different datasets. The work presented in Chapter 5 has been published in the following paper:

Paolo Rosso, Dingqi Yang, Philippe Cudré-Mauroux: RETA: A Schema-Aware, End-to-End Solution for Instance Completion in Knowledge Graphs. WWW 2021.

1.3 Additional Contributions

In addition to the main contributions of this thesis mentioned above, we also published further works related to embeddings and graphs.

- **Highly-efficient graph embedding technique preserving high-order node proximity:** in this work, we propose NodeSketch, a highly-efficient graph embedding technique preserving high-order node proximity via recursive sketching. Specifically, built on top of an efficient data independent hashing/sketching technique, NodeSketch generates node embeddings in Hamming space. For an input graph, it starts by sketching the self-loop-augmented adjacency matrix of the graph to output low-order node embeddings, and then recursively generates k-order node embeddings based on the self-loop-augmented adjacency matrix and (k-1)-order node embeddings. This work has been published in the following paper:

  Dingqi Yang, Paolo Rosso, Bin Li, Philippe Cudré-Mauroux: NodeSketch: Highly-Efficient Graph Embeddings via Recursive Sketching. KDD 2019: 1162-1172

- **Location prediction:** in this work, we aim at predicting a user's next location based on historical user mobility traces. This problem has been studied using Recurrent Neural Networks by incorporating spatiotemporal factors into the recurrent hidden state. However, such a scheme oversimplifies the temporal periodicity and spatial regularity of user mobility. In order to solve this problem, we propose Flashback, a general RNN architecture designed for modeling sparse user mobility traces by doing flashbacks on hidden states in RNNs. Specifically, Flashback explicitly uses some spatiotemporal context to search past hidden states with a high predictive power for location prediction, which can then directly benefit from rich spatiotemporal contexts. This work has been published in the following paper:

Dingqi Yang, Benjamin Fankhauser, Paolo Rosso, Philippe Cudré-Mauroux: Location Prediction over Sparse User Mobility Traces Using RNNs: Flashback in Hidden States! IJCAI 2020: 2184-2190

- **Entity disambiguation on tables:** in this work, we present two novel and unsupervised Web table annotation methods, which leverage the context of the tables to better capture their semantics. Our first method is lookup based and exploits text similarity to find reference entities in the knowledge base. The second method uses distributional vector representations – a.k.a. embeddings – of the Web tables to elicit their context and disambiguate their semantics. This work has been published in the following paper:

  Yasamin Eslahi, Akansha Bhardwaj, Paolo Rosso, Kurt Stockinger, Philippe Cudré-Mauroux: Annotating Web Tables through Knowledge Bases: A Context-Based Approach. SDS 2020: 29-34

- **Entity disambiguation on microposts:** in this work we propose an approach that extends the context of a micropost by leveraging graph-based algorithms to further disambiguate the entities present in it. Our approach, GraphEDM, is divided into two phases. First, we use unsupervised clustering approaches to regroup tweets in semantic neighborhoods using embedding approaches. Next, each ambiguous entity in a cluster is iteratively disambiguated by leveraging a graph-based algorithm. This work is submitted for publication:

  Prathyusha Nerella, Akansha Bhardwaj, Paolo Rosso, Philippe Cudré-Mauroux. GraphEDM: A Graph-Based Approach to Disambiguate Entities in Microposts.

- **Training the embeddings of tabular data:** in this work, we propose an effective approach based on Convolutional Neural Networks called ConvTab to train the embeddings of tabular data. Our approach is divided into two phases. First, we leverage the discriminating power of CNNs to train a table classifier. Next, the representations learned from this model are used to generate semantic features for query-table similarity. These query-table similarity features are used as input to the learning algorithm. This work is submitted for publication:

  Akansha Bhardwaj, Vibhav Agarwal, Paolo Rosso, Philippe Cudré-Mauroux. ConvTab: A Context-Preserving, Convolutional Model for Ad-Hoc Table Retrieval.

1.4 Outline

The rest of the thesis is organized as follows. Chapter 2 reviews existing work in training Knowledge Graph embeddings and describes their learning process. Additionally, we describe typical KG embedding applications, such as link prediction, triple classification, entity classification and entity resolution, and other application domains beyond KGs such as relation extraction, question answering and recommendation systems. In Chapter 3, we revisit text and Knowledge Graph joint embeddings methods and we show that the existing techniques yield suboptimal results, as they fail to control the amount of shared information between the two data sources during the joint learning process and generate additional learning samples. Aiming at releasing the power of such joint embeddings, we propose JOINER, a new joint text and Knowledge Graph embedding method using regularization, able to control the amount of information shared between the two data sources via regularization without generating additional learning samples. In this chapter we also try to answer research question *Q1*, in particular how to benefit from the joint learning. Next, in Chapter 4, we propose a new method to learn hyper-relational facts, where each fact of the KG contains not only a triplet, but also the associated key-value pairs and tackle link prediction. In this chapter, we show that the triplet serves as the fundamental data structure in modern KGs and preserves essential information for link prediction. Specifically, we propose HINGE, a hyper-relational KG embedding model, which directly learns from hyper-relational facts in a KG. By doing so, we answer research question *Q2* and show how to effectively learn KG embeddings from hyper-relational facts. In Chapter 5 we propose an end-to-end solution for instance completion in Knowledge Graphs called RETA, consisting of two components: RETA-Filter, which generates candidate relation-tail pairs for a given head by extracting and leveraging the schema of a KG, and RETA-Grader, which evaluates and ranks the candidate relation-tail pairs considering the plausibility of both the candidate triplet and its corresponding schema. Through the description of RETA, we also answer research question *Q3*, and in particular how to tackle instance completion task by leveraging the schema of the KG. Finally, we conclude in Chapter 6 by summarizing our work and discussing future work, as well sketching future research avenues for Knowledge Graph embeddings.

1.5 What this Thesis is Not About

In the past few years, several transformer-based architecture were proposed such as BERT [30], GPT-3 [21] and ELMo [103], showing great success in Natural Language Processing. Specifically, these architectures can learn contextualized embeddings with large amounts of data and achieve state-of-the-art performance in many NLP tasks. Although these methods can achieve strong performance in several NLP tasks, this thesis proposes a set of KG embeddings trained using shallow neural networks or well-studied architectures, such as Convolutional Neural

Networks. In particular, our goal is not to propose the best performing KG embeddings method, but a set of architectures that aim at achieving a good trade off between architecture complexity and performance. We leave the study of more advanced and complex models, such as transformers, for future work.

In the following chapter, we provide an overview on currently available Knowledge Graphs and investigate state-of-the-art methods for Knowledge Graph embeddings. In addition, we provide an overview on the training process of KG embeddings and how to accelerate it through a technique called negative sampling. Finally, we describe a number of KG embeddings applications.

2 Background

With the growing popularity of multi-relational data on the Web, knowledge graphs have become a key data source in various application domains, such as Web search, question answering, and natural language understanding. In a typical KG such as Freebase ([16]) or Google's Knowledge Graph ([45]), entities are connected via relations. For example, *Bern* is capital of *Switzerland.* Formally, a popular approach to represent such relational data is to use the Resource Description Framework. It defines a fact as a triple (subject, predicate and object), which is also known as head, relation, and tail or (h, r, t) for short. Following the above example, the head, relation and tail are *Bern*, *capitalOf* and *Switzerland*, respectively. With a considerable number of entities and relations (e.g., Google's Knowledge Graph has more than 18 billion of triples with 570 million of entities and 35,000 of relations by the end of 2014), KGs now become a valuable information source that can empower many semantic Web applications.

Despite the importance of building large-scale KGs, their symbolic and logical frameworks are not flexible enough to be compatible with modern statistical and machine-learning techniques which require often numerical inputs. In this context, Knowledge Graph embeddings that project entities and relations in a KG into a low-dimensional continuous vector space have attracted much attention. One of the key benefits of such numeric representation is that they can easily serve as input to classical statistical and machine learning approaches. The learnt entity and relation embeddings can thus be used in different tasks, such as KG completion [19], relation extraction [135], entity classification and entity resolution [95].

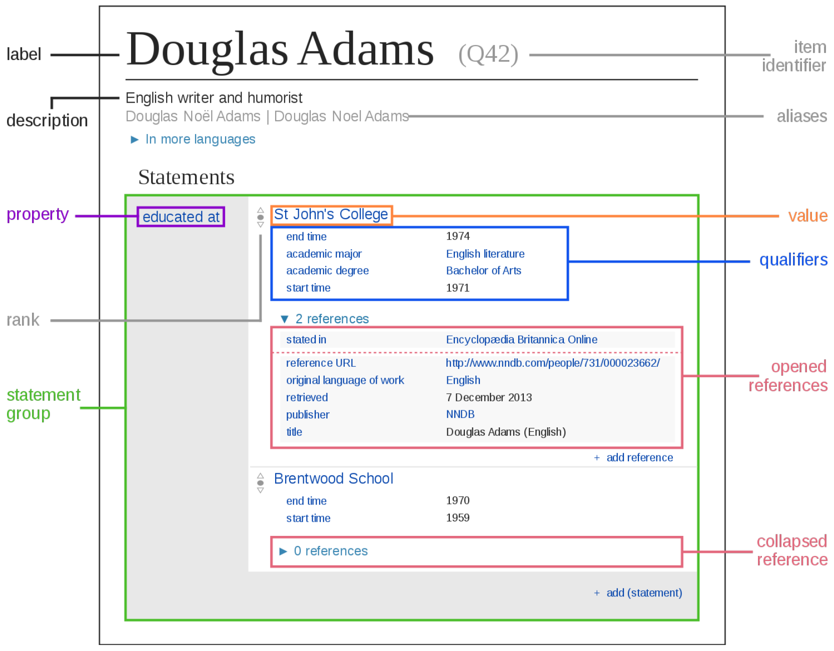
In the following section, we provide a review of currently available Knowledge Graphs. Although the term "Knowledge Graph" has been used in literature since 1972 [114], its modern version comes from the 2012, when Google announces the Google Knowledge Graph [1]. We have grouped these Knowledge Graphs into two categories: publicly available and non-publicly available.

2.1 Publicly and Non-Publicly Available Knowledge Graphs

Publicly available Knowledge Graphs are those KGs that can be freely used by anyone. Examples of publicly available KGs are Wikidata, YAGO and Freebase, which we describe next.

1. **Wikidata** [127] is a collaboratively edited multilingual Knowledge Graph curated and maintained by a large community of thousands of volunteers. Figure 2.1 shows how a Wikidata entry is structured. An *Item* is related to a unique identifier, known as a *QID*. An identifier is linked to a label-description pair to avoid any ambiguity. Optionally, multiple aliases and a number of statements (and their properties and values) are associated with the item. Statements record the information about the item. They consist of key-value pairs, which match a property with one or more entity values. For example, the statement *carrot is orange* would be encoded by pairing a key *color* (P462) with the value *orange* (Q39338) under the item *carrot* (Q81). Statements may also map a key to multiple values. For example, the property *occupation* for *Steve Jobs* is linked with the values *entrepreneur* and *inventor*. Values may have types, such as other Wikidata items, strings, numbers, etc. Following the previous example, the values *entrepreneur* and *inventor* have types *Wikidata items*. A property describes the data value of a statement and can be thought of as a relation connecting the item with a value. Properties have their own pages on Wikidata and may have rules about their usage. For example, the property *capital* can only have a single value, reflecting the case that territories have only one capital city. In our context, items are the heads and tails of a fact, while the properties are the relations. Optionally, qualifiers can provide additional information by refining the meaning of a statement. For example, the property *population* could be linked with the qualifier *as of 2020*. With currently more than 55M data items and over 5.4K distinct properties that help describe these data items, Wikidata is also used by end-user applications such as Google Search and Apple Siri. Wikidata's data is highly dynamic. Editors can create items and add new facts about any topic that satisfies the criteria defined by the community [3]. Such a process brings benefits in terms of data diversity and freshness but does not guarantee the completeness of the data [78].

2. **YAGO** [119] is a KG on general knowledge about people, cities, countries, movies, and organizations. YAGO was one of the first academic projects building a knowledge base automatically. The main idea behind YAGO was to extract information from Wikipedia and unify it with WordNet [87], using a combination of rule-based and heuristic methods. YAGO also includes a pipeline in order to check and filter facts by reaching a manually verified accuracy of 95%. Despite the fact that YAGO2 [54] and YAGO3 [79] increased the size of the original release of YAGO, they could not rival Wikidata in terms of size. Still, YAGO became very popular also thanks to more than 40,000 people who contribute to the project at least once a month. Finally, in 2020 YAGO4 [101] was proposed to



Douglas Adams (Q42)

English writer and humorist
Douglas Noël Adams | Douglas Noel Adams
► In more languages

Statements

educated at

St John's College

end time	1974
academic major	English literature
academic degree	Bachelor of Arts
start time	1971

▼ 2 references

stated in	Encyclopædia Britannica Online
reference URL	http://www.nndb.com/people/731/000023662/
original language of work	English
retrieved	7 December 2013
publisher	NNDB
title	Douglas Adams (English)

+ add reference

Brentwood School

end time	1970
start time	1959

► 0 references

+ add (statement)

Figure 2.1 – The most important terms used in Wikidata. Each item has a unique item identifier, known as a *QID*. Items may also have labels and descriptions that must be unique. Multiple aliases and a number of statements (and their properties and values) are associated with the item.

solve two main problems faced by Wikidata: fact correctness and schema complexity. Since the content in Wikidata is partially added by users, it may contain contradictory statements since Wikidata typically does not enforce semantic constraints (e.g., "each person has exactly one father"). Additionally, Wikidata uses a big set of relations and classes, totaling 6.7k relations, of which only 2.6K have more than 1000 facts, and it comprises around 2.4M classes of which 80% have less than 10 instances; YAGO4 collects facts from Wikidata and constraints them into a simpler and cleaner taxonomy from schema.org. Facts in YAGO 4 are composed by triplets, each of which consists of a head, a relation and a tail [1].

3. **Freebase** [16] is a database used to structure general human knowledge. Traditionally, structured databases have been managed by trusted administrators who created and modified the schema, often with poor support for structural diversity. More precisely, relational databases store data in a fixed and pre-defined structure, leading to a rigid data model where each schema change comes at a cost. At the other end of the spectrum, popular online encyclopedias like Wikipedia define semi-structured document repositories and provide tools with structured query capabilities. Freebase tries to merge the advantages of both scalable structured databases and the diversity of collaborative wikis. Essentially, Freebase is a scalable tuple store containing a large and diverse dataset harvested from sources such as Wikipedia, providing an HTTP/JSON-Based API. However, in 2015 Freebase was shut down and its data move to Wikidata. Nevertheless, the Freebase dump is still utilized in the Knowledge Graph embedding community. Facts in Freebase are also composed by triplets, each of which consists of a head, a relation and a tail [2].

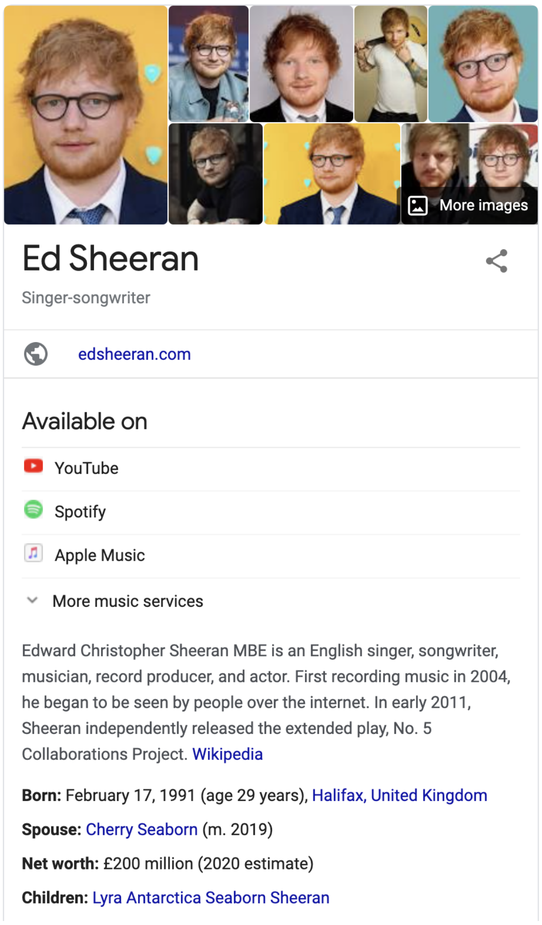
Non-publicly available Knowledge Graphs are KGs where the access to the data is limited. Companies like Google, Amazon, Facebook, Airbnb, Linkedin, IBM, Microsoft, Uber and eBay have developed their own KGs.

1. **Google** integrated their own Knowledge Graph in their search engine to search for things, people or places and instantly get information that is relevant to the input search query. For more than four decades, search engines ranked web pages by the number and quality of links to a page itself. However, Google found out that many query terms were ambiguous or polysemous. For example, *Taj Mahal* can represent a monument, or a Grammy Award-winning musician, or a casino in Atlantic City, NJ, or even an Indian restaurant. Since a given term may refer to different entities, Google came up with a graph that is used to better contextualize real-world entities with their relationships.

[1]https://yago-knowledge.org/downloads/yago-4
[2]https://developers.google.com/freebase

This is a critical first step towards building the next generation of search, where the engine is able to query the Knowledge Graph and provide more relevant results. The Google Knowledge Graph also has the ability to provide a short summary of the retrieved entity, including key facts. For example, Figure 2.2 shows a summary resulting from the query *Ed Sheeran*. As can be seen, Google's Knowledge Graph provides information about the British singer and also key facts such as his date of birth, place of birth, source, etc. The Knowledge Graph also helps to understand the relationships between things. For example, Marie Curie is a person in the Knowledge Graph. She had two children, one of whom also won a Nobel Prize. Additionally, she had a husband, Pierre Curie, who claimed a third Nobel Prize for the family. All these facts are linked in the graph as well, showing that it is not just a catalog of objects but also a graph that models all these inter-relationships. Google began to gradually roll out the Knowledge Graph view to U.S. users, and given its success it has been also rolled out on mobile phones and tablets [2]. Additionally, Google also explored automatic methods for constructing knowledge bases. Specifically, Luna Dong. *et al.* proposed the Knowledge Vault [31], a Web-scale probabilistic knowledge base that combined extraction from Web content (obtained via analysis of text, tabular data, page structure, and human annotations) with prior knowledge derived from existing knowledge repositories. In order to build such a knowledge base, Google employed supervised machine learning methods for merging different information sources [1].

2. **Amazon** developed the Product Graph to help customers find products more easily, suggesting products that fit their need. Amazon's Product Graph describes every item using product and non-product concepts, and form links between different entities/items. However, Amazon has a catalog comprising millions of items, and manually attaching attributes to every product can be an expensive process. In order to solve this issue, the Product Graph team uses several machine learning techniques to get product-related information from Amazon detail pages and the Internet. The challenge in building such a graph is that product information across the Internet is mostly unstructured, making the process of information extraction very difficult. To overcome this problem, Amazon engineers use distantly supervised learning techniques to identify items from a smaller and more structured dataset. The team responsible for the Product Graph also applies knowledge linkage and cleaning to make sure that the data is reliable. One technique used is to make a judgment on the validity of information depending on the source. For example, if a source lists the release of a movie as 2010, while IMDb lists the movie release date as 2011, then the algorithm would use the information from IMDb which is the more trusted source. Lastly, the team applies graph mining techniques to identify interesting hidden patterns, useful to serve recommendations such as "People who bought item A also bought item B" [7]. Figure 2.3 shows the results related to the query *puzzle*. All the suggested items were retrieved using the Amazon Product Graph.



Ed Sheeran

Singer-songwriter

edsheeran.com

Available on

YouTube

Spotify

Apple Music

More music services

Edward Christopher Sheeran MBE is an English singer, songwriter, musician, record producer, and actor. First recording music in 2004, he began to be seen by people over the internet. In early 2011, Sheeran independently released the extended play, No. 5 Collaborations Project. Wikipedia

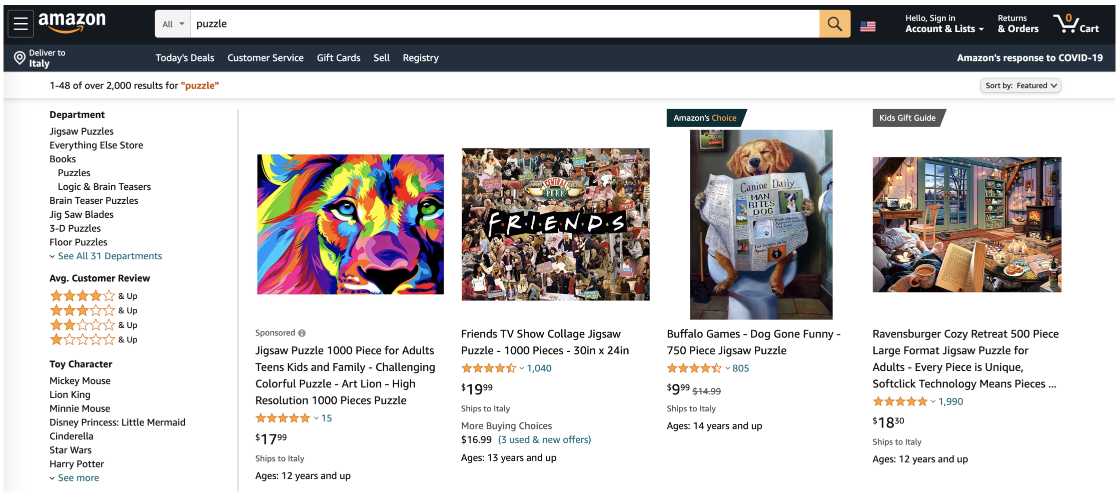
Born: February 17, 1991 (age 29 years), Halifax, United Kingdom

Spouse: Cherry Seaborn (m. 2019)

Net worth: £200 million (2020 estimate)

Children: Lyra Antarctica Seaborn Sheeran

Figure 2.2 – Summary of the user query *Ed Sheeran*. The Knowledge Graph does not only provide a summary of the British singer, but also shows key facts such as date of birth, place of birth, source, etc.



1-48 of over 2,000 results for "puzzle"

Department
Jigsaw Puzzles
Everything Else Store
Books
Puzzles
Logic & Brain Teasers
Brain Teaser Puzzles
Jig Saw Blades
3-D Puzzles
Floor Puzzles
See All 31 Departments

Avg. Customer Review
& Up
& Up
& Up
& Up

Toy Character
Mickey Mouse
Lion King
Minnie Mouse
Disney Princess: Little Mermaid
Cinderella
Star Wars
Harry Potter
See more

Sponsored
Jigsaw Puzzle 1000 Piece for Adults Teens Kids and Family - Challenging Colorful Puzzle - Art Lion - High Resolution 1000 Pieces Puzzle
15
$17.99
Ships to Italy
Ages: 12 years and up

Friends TV Show Collage Jigsaw Puzzle - 1000 Pieces - 30in x 24in
1,040
$19.99
Ships to Italy
More Buying Choices
$16.99 (3 used & new offers)
Ages: 13 years and up

Buffalo Games - Dog Gone Funny - 750 Piece Jigsaw Puzzle
805
$9.99 ~~$14.99~~
Ships to Italy
Ages: 14 years and up

Ravensburger Cozy Retreat 500 Piece Large Format Jigsaw Puzzle for Adults - Every Piece is Unique, Softclick Technology Means Pieces ...
1,990
$18.30
Ships to Italy
Ages: 12 years and up

Figure 2.3 – Example of retrieved results on Amazon typing the query “puzzle”. The most relevant results are fetched using the Amazon Product Graph.

3. **Facebook** has the world’s largest social graph, which also includes information about music, movies, places, etc. that its users care about. Many experiences in Facebook’s products today are powered by their Knowledge Graph, for example when helping people plan movie outings on Messenger. There are four factors that drive the development of Facebook’s Knowledge Graph: coverage, correctness, structure, and constant change. Coverage means being exhaustive in a particular domain, correctness means being able to explain why a certain assertion is made, structure means that the Knowledge Graph must be self-describing (for example an item should have an expected type), and constant change means that the graph has multiple representations (a new graph is build periodically from scratch). Facebook starts by building its KG from the contents of Facebook pages and combines it with further sources of structured data. One of the biggest challenges that Facebook has faced in building its KG is to leverage the data found on its pages and extract relevant information to achieve the goals of a clean and structured KG [99].

4. **Airbnb**, the American vacation rental online marketplace, also relies on Knowledge Graphs to discover what the user wants to know about a destination. In order to provide relevant information to people, Airbnb needs to have some way of representing relationships between distinct but related entities such as cities, activities, cuisines, etc. These types of information became important as Airbnb is an end-to-end travel platform that provides not only places for staying in homes, but also entire experiences such as classes, cultural immersions, etc. Figure 2.4 shows an example of a simplified version of the Airbnb Knowledge Graph. The entities are represented by users, homes, experiences, events, etc. Each entity in the graph has a type, and each type is defined by a unique

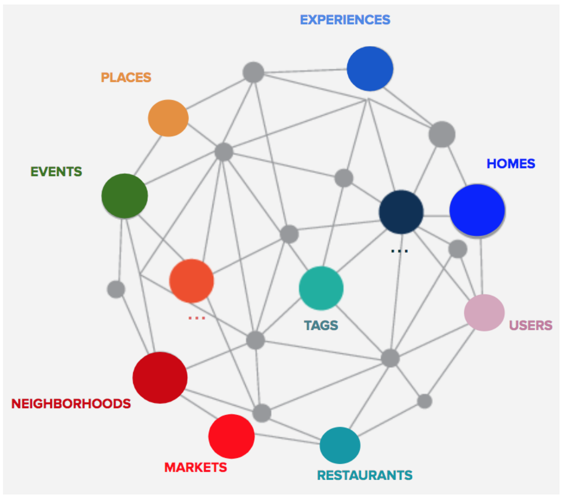
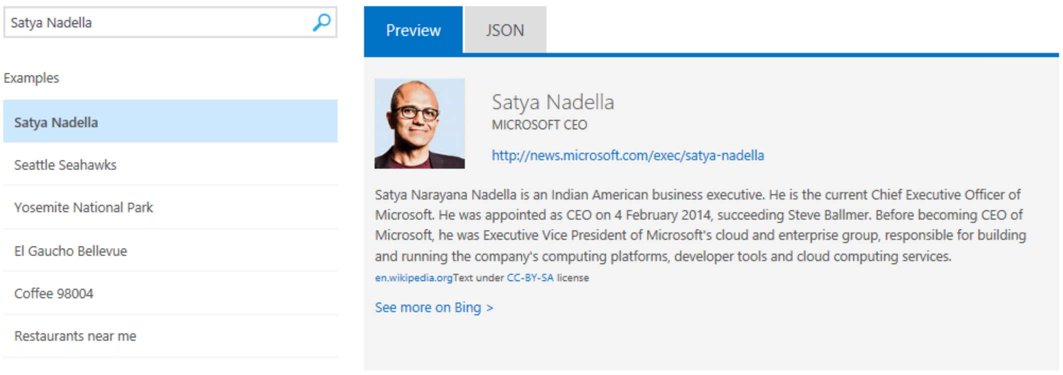


Figure 2.4 – Example of a simplified version of the Airbnb Knowledge Graph. Entities such as users, homes, experiences, events, etc., are interconnected via relations. The KG is used to scale the ability to answer travel related queries.

schema. For example, an entity of type place is defined by its name and GPS coordinate while an entity of type event is defined by its name, date, and venue. Also, relations connecting nodes have specific types to reflect different kinds of relationships among entities, for example language-spoken-in-country [6].

5. **LinkedIn** extensively uses machine learning technologies to optimize products, update news feed, and recommend jobs, people, or articles. LinkedIn's Knowledge Graph defines different entities such as users, skills, jobs, companies etc. The Knowledge Graph is used to enhance recommender systems, monetization, search, consumer products and consumer analytics. The approach used by LinkedIn to build the Knowledge Graph is very different from projects like Wikidata that rely on the contributions from human volunteers, or from other projects like Google's Knowledge Vault that automatically extracts facts from the Internet for constructing knowledge bases. Specifically, LinkedIn builds its KG primarily from user-generated content from members, recruiters, advertisers, and company administrators, and augment it with data extracted from the Internet. Moreover, the Knowledge Graph must be able to scale with new members, companies and skills registered, new jobs posted, etc. Because of the fast-evolving data on LinkedIn, the KG must be updated in real time, for example when a user changes job or when a new company is added. To solve this challenge, LinkedIn applies machine learning techniques on user-generated data and external sources to infer new relations between entities, and then interactively acquire data from users to validate inference results [75].

6. **IBM** developed Watson Discovery, a search and text-analytics service that uses Natural



Satya Nadella
MICROSOFT CEO
http://news.microsoft.com/exec/satya-nadella

Satya Narayana Nadella is an Indian American business executive. He is the current Chief Executive Officer of Microsoft. He was appointed as CEO on 4 February 2014, succeeding Steve Ballmer. Before becoming CEO of Microsoft, he was Executive Vice President of Microsoft's cloud and enterprise group, responsible for building and running the company's computing platforms, developer tools and cloud computing services.
en.wikipedia.orgText under CC-BY-SA license

See more on Bing >

Figure 2.5 – Result of the query *Satya Nadella* using Bing's Entity Search API. The API provides core details and information sources about the entity *Satya Nadella* to augment the user experience.

Language Processing technologies to understand natural language. Watson Discovery also integrates a feature to create Knowledge Graphs. Specifically, given as input an unstructured dataset, Discovery automates the creation of a Knowledge Graph at scale, extracting and disambiguating entities and relationships using sophisticated relevancy ranking techniques [56].

7. **Microsoft** made Bing's Entity Search API available, a service that allows to enrich applications with knowledge from, the web. This API lets users access the Bing Knowledge Graph, which in turn consists of billions of real world entities such as people, places, things etc. Through this service, users can retrieve most of the relevant entities within a document, as well as details and information sources about them. For example, by searching for "Satya Nadella", users get a single dominant entity corresponding to Microsoft's CEO. Additionally, users can also perform searches like "restaurants near me" to get a ranked list of the most relevant local businesses that match the user's interest. Figure 2.5 shows an example of how the information about the entity *Satya Nadella* is extracted from the Knowledge Graph and provided to the users using Bing's Entity Search API. The API not only provides a description of the entity, but also further information such as his official website [82].

8. **Uber** believes that restaurant recommendations is key to the Uber Eats experience. When a user picks food from the Uber platform, at any given time and location, there could be hundreds of options to choose from. There are many factors that can influence this choice such as time, cuisine preference and current mood. Uber tackles this problem by combining recommendation technologies and machine learning. When a user looks for food and types a query, Uber's goal is to understand his/her intent based on knowledge on food organized as a graph in order to expand on the query. Sometimes,

users have clear intents but this is unfortunately not always the case. A user might have a certain type of food in mind, but chooses something else while browsing the app. For example, when a user searches for udon but ends up ordering soba. As humans, this choice may sound natural, because udon and soba are somewhat similar, but for a machine this is not trivial knoledge since users specify their intent through textual keywords in the form of a search query. Consequently, a possible solution is to use query understanding to figure out the intent. A classic approach for query understanding is using Natural Language Processing but this does not work when the intent of the user is unclear. The alternative solution proposed by Uber is to build and use a food-focused Knowledge Graph to better understand food-related queries. Since entities such as restaurants, cuisines, and items have natural relationships, they can be modeled as a graph. One of the main challenges faced by Uber was to build an ontology to describe the graph, including properties of different entities and their relations. Once the graph is established, an offline phase annotates restaurants and menu items with very descriptive tags, while an online phase is used to rewrite the user's query in real time to optimize the quality of the returned results. Additionally, Uber uses the Knowledge Graph to solve other problems like the zero result problem. This problem occurs when a user searches for a restaurant that is not on the platform or that is not located nearby. Instead of returning no result, the platform recommends restaurants in the area with similar cuisines [126]. Figure 2.6 shows a subgraph of the overall ontology used for Uber Eats' Knowledge Graph. In the graph, we can see that dishes like Udon, Sushi, and Soba are linked to Japan, and that Noodle is a super category of Soba and Udon.

9. **eBay** released the eBay ShopBot, a smart personal shopper that helps consumers find items they wished to buy from a marketplace of more than one billion of listings. Users can type queries in order to ask the bot to show the most relevant items. For example, for the query "Can you show me brown leather Coach messenger bags under $100?" the bot is able to identify the object of interest as a handbag and the target price range to be between $0 and $100. subsequently, the bot executes a named-entity recognition program to identify brown as the color, leather as the material, and Coach as the brand. Once the characteristics of the object are recognized, the data is mapped onto eBay inventory using a Knowledge Graph, which is used to find the best results in the shortest amount of time [34]. Figure 2.7 shows a subgraph of ShopBot's Knowledge Graph. Purple entities represent "categories" (e.g., handbag, men's backpack, etc.), green "attributes" (e.g., color, type, etc.) while pink represent "values" for those attributes (e.g., red, toiletry bag, etc.).

In Table 2.1 we summarize the aforementioned Knowledge Graphs.

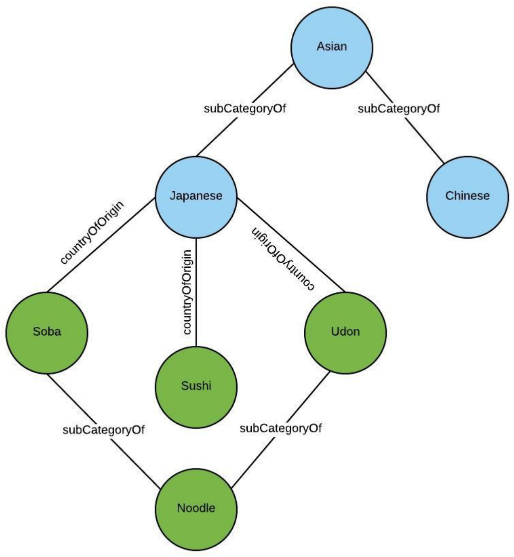


Figure 2.6 – Example of Uber's Knowledge Graph, showing the relationship between cuisines and dish types.

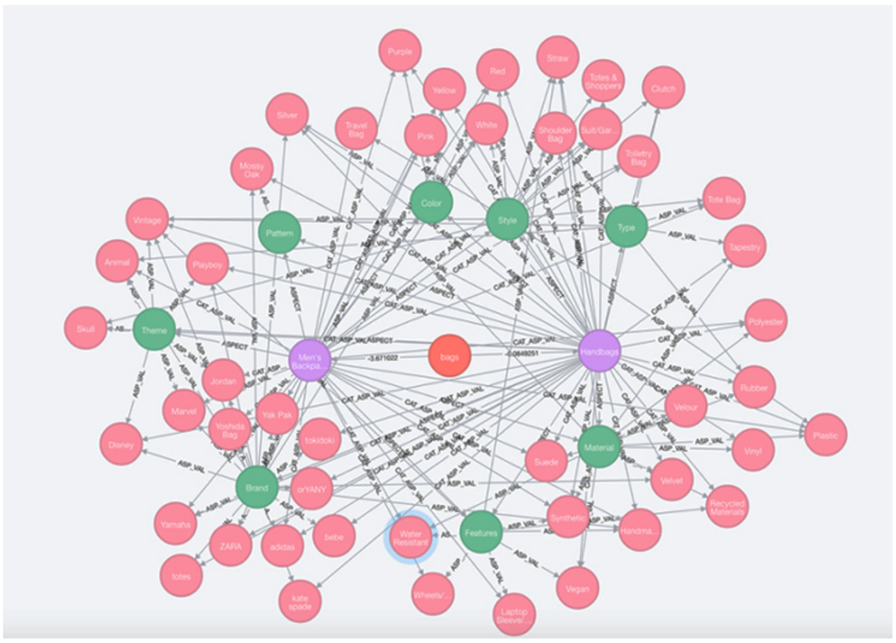


Figure 2.7 – Snapshot of ShopBot's Knowledge Graph used to understand user requests and retrieve the most relevant items. In a search query for "bags" for example, purple nodes represent "categories", green "attributes" and pink are "values" for those attributes.

Table 2.1 – Knowledge Graphs Summary

KG	**Domain**	**Publicly/Non-Publicly**
Wikidata	General	Publicly
YAGO	General	Publicly
Freebase	Human	Publicly
Google	General	Non-Publicly
Amazon	Products	Non-Publicly
Facebook	Social Networking	Non-Publicly
Airbnb	Lodging	Non-Publicly
LinkedIn	Social Networking	Non-Publicly
Microsoft	General	Non-Publicly
Uber	Food Ordering	Non-Publicly
eBay	Products	Non-Publicly

In the following, we first discuss typical KG embedding models and then their extensions by integrating additional data sources. We then summarize the applications of KG embeddings.

2.2 Learning Knowledge Graph embeddings

Learning KG embeddings consists in two key steps in general:

1. Defining a KG embedding model with a specific scoring function, which computes the probability that a given triple is true;

2. Initializing entity and relation embeddings (e.g. vectors) according to the KG embedding model, and learning those embeddings by maximizing the sum of the scoring function over all triples in the KG. Triples appearing in the KG will have higher scores than the triples that do not exist in the KG.

2.2.1 KG embedding models

Depending on the type of scoring function, there are three categories of embedding models: translational distance models, semantic matching models, and deep learning models. Translational distance models, such as *TransE* ([19]), use a scoring function that measures the distance between two entities, semantic matching models, such as *RESCAL* ([95]), use a scoring function that measures the similarity of the facts, while deep learning models, such as *ConvE* ([29]), use deep neural networks.

Translational distance model: TransE and its extensions

TransE is a representative translational distance model that projects entities and relations onto a unique vector space. In this model, the head h and the tail t of a triplet are connected by their relation r, holding the fact that the embedding of t should be similar to the embedding of h plus the embedding of r (i.e., $h + r \approx t$). The proposed idea is based on the vector-offset method for identifying linguistic regularities in continuous space word representations [86], for example, *USA - dollar* ≈ *Japan - yen.* In a KG, this analogy holds since through the *currencyOf* relation we get *dollar* + *currencyOf* ≈ *USA* and *yen* + *currencyOf* ≈ *Japan.* In this way, the scoring function is defined as the negative distance between the sum of the head and the relation, subtracted by the tail:

$$f_r(h,t) = -||\mathbf{h} + \mathbf{r} - \mathbf{t}||_{1/2} \tag{2.1}$$

Following this initial idea, several techniques improved TransE by designing sophisticated scoring functions that are able to capture complex KG structures, in particular multi-mapping relations (one-to-many, many-to-one, or many-to-many). TransH ([132]), for instance, suggests a new approach by projecting the relations on different hyperplanes in order to capture many-to-many mapping properties of some relations; TransR ([74]) defines a mapping matrix and a vector for every relation; TransD ([59]) introduces dynamic matrices for each entity-relation pair by considering the diversity of entities and relations simultaneously. Sophisticated scoring functions can indeed improve the KG embeddings in some downstream learning tasks, tough they also increase the complexity of the embedding models. RotatE [120] uses complex latent space and define each relation as a rotation from the head entity to the tail entity. TorusE [35] was motivated by the fact that TransE uses the regularization to force entity embeddings to lie on a hypersphere, thus limiting their capability to satisfy the translational constraint. To handle TransE's regularization problem, TorusE embeds entities and relations vectors in a torus space

Semantic matching model: RESCAL and its extensions

RESCAL is a tensor factorization model for KG embeddings, which decomposes a three-way tensor consisting of head, relation and tail dimensions. RESCAL generates for each entity a vector, and for each relation a matrix capturing the interaction between the entities. The proposed model allows for discovering the correlation between multiple interconnected entities. The model represents facts via a tensor product with a corresponding scoring function defined as follows:

$$f_r(h,t) = \mathbf{h}^\top \mathbf{M}_r \mathbf{t} = \sum_{i=1}^{d} \sum_{j=1}^{d} [\mathbf{M}_r]_{ij} \cdot [\mathbf{h}]_i \cdot [\mathbf{t}]_j \tag{2.2}$$

where h and t are the vectors of the head and tail, respectively, and M_r is the matrix that represents the relations. d refers to the dimension of the embeddings.

Several pieces of work extend RESCAL by designing customized tensor factorization models. DistMult ([143]) simplifies the RESCAL model by using a bilinear formulation, showing similar performance with less parameters (more efficient in the learning process). However, the model works with symmetric relations only. [124] propose a model called Complex Embeddings (ComplEx) that extends DistMult in order to model asymmetric relations. ComplEx introduces a complex space in which head, relation and tail embeddings are represented. In this model, the scoring function generates different scores from facts with asymmetric relations. Neural Tensor Network (NTN) ([117]) is another model with a neural network architecture. For each fact, the embedding vectors of its head and tail are fed into the input layer of a neural network, and then mapped onto its hidden layer combined with a relation-specific tensor. Finally, the output layer generates a score for each fact. ANALOGY [76] also extends RESCAL in order to model the analogical properties of entities and relations, for example, "AlfredHitchcock is to Psycho as JamesCameron is to Avatar". SimplE [63] extends DistMult by learning two embeddings of each entity, one in case the entity is the head of a fact and the other in case the entity is the tail. [58] observed that factorization models often fail when they make entity predictions that are incompatible with the type required by the relation. To solve this problems, [58] proposed TypeDM and TypeComplex, a modification of DistMult and ComplEx respectively by explicitly modeling type compatibility.

Deep learning model: ConvE and its extensions

ConvE [29] is a deep learning model for KG embeddings, which uses a Convolutional Neural Network and a fully-connected layers to capture the interactions between entity and relation embeddings. Specifically, ConvE uses a convolution layer directly over 2D reshaping of head and relation embeddings. The scoring function is defined as following:

$$f_r(h,t) = f(vec(f([\bar{\mathbf{h}};\bar{\mathbf{r}}] * \omega))\mathbf{W})\mathbf{t} \tag{2.3}$$

where $\bar{\mathbf{h}}$ and $\bar{\mathbf{r}}$ denote a 2D reshaping of $\mathbf{h}$ and $\mathbf{r}$, respectively. Specifically, the model concatenates $\mathbf{h}$ and $\mathbf{r}$ and uses it as an input for a 2D convolutional layer with filters ω. The convolutional layer returns a set of feature maps that are reshaped into a vector, which is then projected into a k-dimensional space using a linear transformation parametrised by the matrix

W, and matched with the tail embedding **t** via an inner product.

HypER [11] simplifies ConvE by using a hypernetwork to generate a set of 1D relation-specific filters, then extracts relation-specific features from the head embedding. ConvKB [91] extends ConvE by applying a convolution layer over the embedding triplets (h, r, t). In this model, each triplet is represented as a 3-column matrix where each column vector represents a triplet element. CapsE [128] extends ConvKB using capsule network to model facts in a KG. Similarly to ConvKB, it stacks the embedding triplets (h, r, t) in a 3-column matrix where each column vector represents the embedding of an element in the triplet. The 3-column matrix is the fed to a CNN in order to generate a set of future maps. These future maps are fed as input of a capsule neural network to produce continuous vectors. Another relevant work is NaLP [47]. Specifically, it learns n-ary relational facts (represented as a set of role-value pairs) by concatenating role-value pairs and fed them to a CNN in order to capture the correlation between pairs.

2.2.2 Learning process

The goal of the learning process is to maximize the sum of the scoring function over all triples in the KG. Typical examples of optimization algorithms used in this context include Stochastic Gradient Descent (SGD) ([107]), Broyden–Fletcher–Goldfarb–Shanno (BFGS) ([13]) and AdaGrad ([33]). In order to accelerate the training process, negative sampling techniques can be applied by replacing the head, the relation or the tail of a given fact. These generated triples are called negative samples. There are two main methods of generating negative samples, based on the *Open World Assumption (OWA)* and on the *Closed World Assumption (CWA)*.

Open World Assumption

The open world assumption assumes that the KG only contains true facts and the facts that do not appear can either be false or just missing [32]. Under this assumption, a negative fact is probabilistically generated given a positive fact by randomly corrupting its head, relation or tail. The entity and relation representations are learned by minimizing a loss function defined based on the scoring function, such as the logistic loss defined as follows:

$$\min_{h,r,t} \sum_{(h,r,t)\in\mathbb{D}^+\cup\mathbb{D}^-} log(1 + exp(-y_{hrt} \cdot f_r(h,t))) \tag{2.4}$$

where D^+ and D^- are positive and negative training samples, and y_{hrt} is equal to 1 if the label is positive, -1 otherwise. The logistic loss can be optimized using for instance stochastic gradient descent in mini-batch mode. In the training phase, a set of true facts are sampled and a set of negative fact get generated, and the embeddings can be iteratively updated with a fixed or adaptive learning rate.

Closed World Assumption

The closed world assumption assumes that all the facts that are not in the KG are negative samples (i.e., assuming the KG is complete). The entity and relation representations are learned by minimizing, for example, the squared loss:

$$\min_{h,r,t} \sum_{h,t \in \mathbb{E}, r \in \mathbb{R}} (y_{hrt} - f_r(h,t))^2 \tag{2.5}$$

where $\mathbb{E}$ and $\mathbb{R}$ are the set of entities and relations, respectively. y_{hrt} is equal to 1 if the triplet appears in the KG, 0 otherwise. In addition, [94] propose the logistic loss and [83] the absolute loss as alternatives to the squared loss.

In summary, the closed world assumption usually has more limitations than the open world assumption, as it penalizes the missing true facts from a KG. In practice, despite their tremendous size, modern KGs all suffer from incompleteness issues [134]. Consequently, the open world assumption is more realistic for most settings and thus performs better on average better than the closed world assumption [49].

2.3 KG embedding extension by integrating additional information

When learning KG embeddings, additional information, such as entity types, relation paths, textual descriptions or logical rules, can be added to the embedding model to improve the quality of the embeddings on certain tasks.

Entity types represent semantic categories the entity belongs to. For example, the entity *PresidentObama* can be annotated as a *PERSON* entity type. This piece of information can be incorporated in different ways. [96] use the entity type as a relation and its corresponding facts *(h,r,t)* as training example. [48] propose a method in which entities of the same type are close to each other in the vector space. The entity type can also be used to set constraints for different relations. For example, [137] use this constraint to generate correct negative samples by filtering out triples with incorrect entity types.

Relation paths refer to a sequence of relations between two entities. The multi-hop relationships contain useful information that can be used for KG completion. For example, [70] predict the relation between entities using a path ranking algorithm that connects two entities. More precisely, the relations can be represented as vectors or matrices, and their addition or multiplication can be used to compose a path as vector or matrix composition. [73] show a method to approximate the relation path via sampling and pruning. Along similar lines, [123] propose an algorithm that incorporates paths with specific lengths and intermediate entities in the model.

Most of the KGs contain entity descriptions that can be used to enrich the semantic information of the entities. External information sources, such as Wikipedia articles [131] or news releases [117], can be used to extend the entity description by providing richer textual information. A representative work by [131] introduce a model that combines text corpus and KG to align them in the same vector space and creates KG embeddings and text embeddings. The model includes three main parts: a KG model, a text model and an alignment model. Specifically, the KG model is used to generate embeddings of entities and relations in the KG while the text model is used to generate embeddings from the text corpus. Finally, the alignment model is used to align the KG embedding and text embedding in the same vector space using different alignment mechanisms, such as entity name and Wikipedia anchors. In this way, the model is able to predict out-of-KG entities (phrases not stored in the KG but that appear in the text). Similarly, [109] propose JOINER, a joint text and KG embedding method using regularization. JOINER not only preserves co-occurrence between words in a text corpus and relations between entities in a KG, it also provides the flexibility to control the amount of information shared between the two data sources via regularization. Compared to other models that combine text corpus and a KG, JOINER does not generate additional learning samples, which makes it computationally efficient.

Logical rules are another type of information that could be integrated into embedding models. For instance, if two entities are connected by the relation *HasWife*, then they should also be connected by the relation *HasSpouse*. There exist systems, such as *WARMR* ([28]), *ALEPH* ([90]), and *AMIE* ([43]), that can automatically extract such kinds of relations. [105] prove that the logical rules contain rich information and that they can be used to acquire and infer further knowledge. Following this idea, [130] propose an approach seamlessly incorporating logic rules into KG embedding models by reducing the solution space and thus improving the inference accuracy for knowledge base completion tasks. [50] propose a model that embeds KG and logical rules in a unified framework. Specifically, logical rules are first instantiated into ground rules, for example, *HasWife(x,y)* implies *HasSpouse(x,y)* and vector embeddings are introduced for entity pairs. For each fact of the logical rule, a score is computed in order to indicate whether the ground rule is satisfied or not. The embedding model is then learned based on the unified facts and rules. In this way, the model is more effective for knowledge acquisition and inference as the embeddings are compatible with both facts and rules.

Key-value pairs associated with triplets are another type of information that could be integrated into KG embedding models. Specifically, hyper-relational facts are facts that contain not only a base triplet *(h,r,t)*, but also some associated key-value pairs *(k,v)*, where a key-value pair denotes a relation-entity pair or a relation-literal pair associated with the base triplet. Existing works, [47, 133, 152] transform hyper-relational data (a.k.a. multi-fold or n-ary relational data) into an n-ary representation, i.e., a set of key-value (relation-entity) pairs while completely avoiding triplets. Using this representation, the aforementioned works learn the relatedness between entity/relation pairs. Specifically, m-TransH ([133]) models the interaction between entities involved in each fact, and RAE ([152]) further extends m-TransH by considering the relatedness between entities in each fact. As these two works capture only the relatedness between entities, NaLP ([47]) was subsequently proposed to model the relatedness between key-value (relation-entity) pairs contained in each fact. However, [111] show that transforming a hyper-relational fact into an n-ary representation is incompatible with the schema used by modern KGs, where triplets still serve as the fundamental data structure to discover missing facts based on the contents of the KG. This means that key-value pairs on a hyper-relational fact should not be treated equally to base triplets *(h,r,t)*, as the latter actually preserves essential information for discovering missing facts. To address this issue, [111] propose HINGE, a hyper-relational KG embedding model that captures not only the primary structural information of the KG encoded in the triplets, but also the correlation between each triplet and its associated key-value pairs.

In addition to the information described above, further types of information that can be added to the embedding models include entity attributes, temporal information and graph structures. [96] highlight the fact that entity attributes and relations must be separated. The authors propose a new algorithm to handle attributes efficiently. [61] show that the KG facts are often time-sensitive and that they develop a time-aware knowledge base embedding approach by taking advantage of the time at which facts have occurred. The proposed solution forces the embeddings to be temporally consistent by using temporal constraints to model the relations. [40] show a graph-aware approach that learns entitiy and relation embeddings by leveraging the relation paths and edge context (i.e., all the relations that connect an entity). The intuition behind this approach is that all the relations linking to and from an entity are representative of that entity.

2.4 KG embedding applications

Typical applications of KG embedding include link prediction, triple classification, entity classification and entity resolution.

- Link prediction (also called KG completion) attempts to discover missing facts based on the contents of the KG. Specifically, it predicts an entity given a relation and a second entity, i.e., given *(r, t)* it predicts *h*, denoted also as *(?, r, t)*, or given *(h, r)* it predicts *t*, denoted also as *(h, r, ?)*. [73] define this task as entity prediction while [18] define it as entity ranking. A similar approach predicts a relation given its head and tail entities, denoted also as *(h, ?, t)*, which is similar also to relation prediction [137]. In order to evaluate the results generated by this task, a common practice is to store in a list all the answers and see the rank of the correct answer. Several evaluation metrics can be used in this context, such as *Hits@n* that considers only the ranks smaller than *n*, or the *mean rank*, that is, the average of the predicted ranks.

- Triple classification aims at determining whether a triple appears in a KG. More precisely, triple classification can be performed based on the score of a candidate triple *(h, r, t)* that can be easily computed using the scoring function. In this way, an unseen fact can be either true if its score is higher than a threshold and false otherwise. Traditional evaluation metrics can be used in this task, for example, mean average precision [50] or micro and macro averaged accuracy [48].
- Entity classification classifies entities into different semantic categories. Concretely, the type of an entity is usually denoted using a *IsA* relation, and entity classification can thus be seen as a particular case of the link prediction task, in which only *(h, IsA, ?)* triples are predicted.
- Entity resolution verifies whether two entities are actually referring to the same object or not. [18] tackle this problem by considering a scenario in which the relation of two equivalent entities is explicit denoted as *EqualTo*. By learning the embedding for this type of relation, the problem of entity resolution becomes a triple classification problem. Fundamentally, the triple classification problem judges whether the fact *(h, EqualTo, t)* is true or not. Alternatively, [95] propose a different approach that computes the similarity between two entities and use the score to calculate the likelihood that two entities refer to the same object. This method works even if the relation *EqualTo* is not encoded in the KG.

KG embeddings can also be applied to other application domains beyond KGs. Three most popular out-of-KG applications are relation extraction, question answering and recommendation systems.

- Relation extraction tries to discover relations from text where entities have already been identified. [135] propose a method to extract relations by combining TransE and text, showing that the integration of TransE and a traditional text-based extractor can actually improve the performance of relation extraction.
- Question answering refers to the task of answering questions over KGs. Given a question in plain text, a fact or a set of facts containing the correct answer is extracted as an answer.

This task is challenging because of the extended variability of natural language text used to formulate the question and of the extensive size of the KGs. A successful solution that involves KG embedding is proposed by [17], which learns embeddings in order to put questions and corresponding answers closer in the vector space. Given a question and an answer, the model generates a high score if the answer is correct, low score otherwise. The results show that, by involving the KG, the task is successfully performed without using any rules or additional tagging step as most traditional question answering applications do.

- Recommendation systems suggest users a list of items according to the users' preferences. Collaborative filtering techniques are often used to perform recommendations based on the historical interaction between users and items. However, user-item interactions are often sparse, leading to unsatisfactory performance. To alleviate this issue, hybrid recommendation systems were developed by adding auxiliary information [148]. [151] propose a hybrid recommendation system that integrates a KG. More precisely, the hybrid recommendation system models structural knowledge by applying a KG embedding technique such as *TransR* in order to learn the representation of each item. Similarly, the users are represented by vectors, and each item is represented by its KG vector representation plus an offset. Finally, the preference of a user for a specific item is computed as a product of the user and item vectors. In this way, the hybrid recommendation system automatically extracts semantic representations from facts in the KG to improve the quality of the recommendation system.

2.5 Conclusion

With the booming of multi-relational data on the Web, Knowledge Graphs have become an important data source empowering many applications. However, the symbolic and logical representation of KGs make it difficult to take them as input to machine-learning or processing pipelines. To tackle this issue, knowledge graph embedding techniques were proposed to project entities and relations from a KG into a low-dimensional continuous vector space, while still preserving the inherent structure of the KG and reasoning capabilities over the KG. The learnt embeddings have been successfully used in both KG-reasoning applications, such as link prediction, triple classification, entity classification and entity resolution, and out-of-KG applications, such as relation extraction, question answering and recommendation systems.

In the next chapter we investigate about how to jointly learn embeddings from text and a Knowledge Graph in order to benefit from both large-scale unstructured content text and high-quality structured Knowledge Graph. Specifically, we propose a method that not only jointly learn embeddings from text and a Knowledge Graph, but also regularize the amount of shared information between the two data sources. We compare our proposed method against a collection of state-of-the-art KG embeddings across different evaluation tasks, such as analogical reasoning, link prediction, and relation extraction, showing that our method yields better results and significantly less computational overhead.

3 Revisiting Text and Knowledge Graph Joint Embeddings: The Amount of Shared Information Matters!

Jointly learning embeddings from text and a Knowledge Graph benefits both word and entity/relation embeddings by taking advantage of both large-scale unstructured content (text) and high-quality structured data (the Knowledge Graph). Current techniques leverage anchors to associate entities in the Knowledge Graph to corresponding words in the text corpus; these anchors are then used to generate additional learning samples during the embedding learning process. However, we show in this chapter that such techniques yield suboptimal results, as they fail to control the amount of shared information between the two data sources during the joint learning process. Moreover, the additional learning samples often incur significant computational overhead. Aiming at releasing the power of such joint embeddings, we propose JOINER, a new joint text and Knowledge Graph embedding method using regularization. JOINER not only preserves co-occurrence between words in a text corpus and relations between entities in a Knowledge Graph, it also provides the flexibility to control the amount of information shared between the two data sources via regularization. Our method does not generate additional learning samples, which makes it computationally efficient. Our extensive empirical evaluation on real datasets shows the superiority of JOINER across different evaluation tasks, including analogical reasoning, link prediction, and relation extraction. Compared to state-of-the-art techniques generating additional learning samples from a set of anchors, our method yields better results (with up to 4.3% absolute improvement) and significantly less computational overhead (76% less learning time overhead).

3.1 Introduction

Jointly learning embeddings from both text and Knowledge Graphs has been shown to combine the advantages from both *large-scale* unstructured data (text) and *high-quality* structured data (KGs). On one hand, word embeddings techniques, represent words in unstructured text as real-valued vectors, which can be efficiently used as features by various downstream applications such as document classification [115], named entity recognition [125], or sentiment analysis [118]. SkipGram [84, 85], for instance, is a well-known word embedding technique that can efficiently learn word embeddings by preserving the co-occurrence between words in a context window in a large text corpus, hence implicitly capturing semantic relations between words (e.g., $USA - dollar \approx Japan - yen$). Although word embedding techniques can efficiently capture the semantics of words from large-scale unstructured text, they typically cannot model the nature of the *relation* between pairs of words, e.g., they cannot infer the *currency* relation between *Japan* and *yen*.

Knowledge Graphs, on the other hand, contain high-quality structured data where entities are connected via relations. More specifically, a typical KG, such as Freebase [16], Google's Knowledge Graph [45] or Wikidata [136], can be represented as a set of facts; each fact is a triplet *head, relation, tail* or (h, r, t) for short, such as *Japan* (head) *hasCurrency* (relation) *yen* (tail). To efficiently exploit KGs in practice, KG embeddings [129] have been proposed to represent entities as vectors, and relations as operations, while still preserving the relations between entities. A typical KG embedding technique is TransE [19]. It models a relation as a vector-plus operation and hence preserves the relations between entities as $h + r \approx t$. By doing so, it preserves the reasoning ability of a KG, i.e., a new fact can be asserted by evaluating $||h + r - t||$. Although the high-quality and structured data present in KGs have been widely used to power various applications, ranging from question answering [146] to query expansion [46], they are known to suffer from incompleteness (i.e., they are missing many/most entities and facts) [19, 131]. For example, 93.8% of persons from Freebase have no *place of birth*, while 78.5% have no *nationality* [88].

In the current literature, the most popular scheme to jointly learn embeddings from text and KGs [131, 153, 39, 141] is to define one or several types of anchors that link entities in a KG and words in a text corpus, such as entity names (labels) appearing in text or entities associated to Wikipedia pages. Subsequently, these anchors are used to generate additional learning samples during the embedding learning process. For example, when an entity's name (label) appears in text, beyond the co-occurrence between words, a joint embedding model additionally learns from the co-occurrence of the entity and its surrounding words [131]. Although such a joint learning scheme has shown the advantage of combining text and KGs, we still raise the following question: *Does this scheme optimally combine the advantages from the two data sources?*

To answer this question, we started by reviewing the aforementioned joint learning process. More precisely, the heuristically-defined anchors used by this scheme uniquely determine the additionally-generated learning samples, and thus implicitly control the extend to which the word and KG embeddings are *jointly* learnt from the two data sources. Subsequently, it fails to provide the flexibility to control *how much information is actually shared between the two data* *sources in the embedding learning process*, which in many cases leads to suboptimal results (as we show below in our experiments). Even though one can combine different heuristics to generate different sets of anchors, it still gives very limited flexibility to control the *joint* learning process (see the baseline JointAS in our experiments for more details). Moreover, this joint learning scheme also suffers from an inefficiency issue, as the additionally generated learning samples require extra computation and learning time.

Aiming at releasing the power of joint text and KG embeddings, we propose *JOINER*, a novel Joint text and KG Embedding learning method via Regularization. JOINER not only preserves the co-occurrence between words in a text corpus and the relation between entities in a KG, but also provides the flexibility to smoothly control the amount of information shared between the two data sources in the joint embedding learning process using regularization. Specifically, we insert a regularization term in both the text and the KG embedding models to minimize the distance between a word (from a text corpus) and an entity (from a KG) when the two are linked by an anchor (e.g., entity names appearing in text). Subsequently, we are able to smoothly control via a regularization parameter the extent to which the word and the KG embeddings are *jointly* learnt from the two sources. Moreover, without additionally generated learning samples, JOINER incurs significantly less computation overhead compared to classical joint embedding learning methods. We conduct a thorough evaluation of JOINER on three evaluation tasks (analogical reasoning, link prediction and relation extraction) using Freebase, Wikipedia and New York Times corpora. Our results show the superiority of JOINER. In particular, compared to the state-of-the-art technique generating additional learning samples from a set of anchors [131], our method yields better performance (with a 1.4%-4.3% improvement on different evaluation tasks), and 76% less computational overhead. Moreover, our results also show that the amount of shared information matters across different evaluation tasks.

3.2 Related work

Word embedding techniques learn vector representations of words from a text corpus by preserving the semantic similarity between words [15, 26, 84, 102]. For example, the SkipGram model [84] learns word embeddings by maximizing the co-occurrence probability of words in a text corpus. The resulting embeddings are able to capture semantic and syntactic relations between words. By incorporating a negative sampling technique and using asychronized stochastic gradient descent to speedup the embedding learning process [85], the SkipGram model has been shown to be able to scale up to a large corpus in practice. Although word embedding models are able to efficiently capture semantic and syntactic relations between words from a large-scale corpus, they cannot extract explicit relations between pairs of words.

Knowledge Graph embedding techniques learn vector representations of entities and relations in a KG by preserving the relations between entities and thus maintaining the reasoning ability of the KG [108, 129]. The existing KG embedding techniques can be classified into two categories, i.e., translational distance models and semantic matching models. Translational distance models exploit distance-based scoring functions to create the embeddings. One representative model is TransE [19], which creates embeddings from triplet (h, r, t) such that the relation between the head and tail entities are preserved as $h + r \approx t$. Several works further improve TransE to capture complex KG structures, such as multi-mapping relations (one-to-many, many-to-one, or many-to-many), using more sophisticated scoring function involving relation-specific hyperplanes [132] or spaces [74, 59], for example. Ebisu et. al.[36] employ the same principle applied in TransE to a Lie group, which is a compact space representation. Semantic matching models, on the other hand, exploit similarity-based scoring functions. One typical model is RESCAL [95]. It represents each entity as a vector and each relation as a matrix, and uses a bilinear function to model the relation between two entities. Several works also extend RESCAL by putting a specific focus on reducing the model complexity [143], by capturing asymmetric relations [124], or by modeling non-linear relations using neural networks [117, 29, 113, 10, 92]. Another promising direction is the Data-to-Text movement [5, 51], whose goal is to decode a Knowledge Graph into sentences and then add them as training for a language model such as GPT-3 [21]. Despite achieving good performance, these non-linear models are computationally expensive, non-transparent and poorly understood, as opposed to translational distance and semantic matching models. Although KG embeddings are able to capture specific relations between entities, they suffer from incompleteness, i.e., many triplets are missing from the KG.

Joint text and Knowledge Graph embeddings learn embeddings by combining the two data sources. The most common scheme to jointly learn text and KG embeddings is to define one or several types of anchors that link entities in a KG and words in a text corpus, which are then used to generate additional learning samples in the embedding learning process. One representative method is proposed by [131]. Specifically, based on a SkipGram-like text model and a TransE-like KG model, it jointly learns word and entity/relation embeddings. It defines two types of anchors connecting words in a text corpus and entities in a KG: 1) entity names (labels) appearing in text and 2) entity-associated Wikipedia pages, and then generates additional learning sample from these anchors. For entity names appearing in text, additional learning samples are generated for the KG model by replacing the entity from each head-relation-tail triplet by the corresponding word (e.g., word-relation-tail if the head is an anchor from text). For entity-associated Wikipedia pages, additional learning samples are generated for the text model by replacing the word from each word pairs by the associated entity (e.g., word-entity). [153] extends this model by using another type of anchors, i.e., words appearing in the entity description from the KG. Following this joint learning scheme, many task-specific joint embedding methods have also been proposed. For example, [39] combines all the aforementioned types of anchors to jointly learn embeddings specifically for the entity disambiguation problem. [141] addresses the same problem considering not only entity names appearing in text as anchors, but also the relatedness between entities. [53] learns joint text and KG embeddings using entity name appearing in text as anchors, and also by inferring relations from text using distant supervision. However, all these methods do not provide the flexibility to control how much information is actually shared between the two data sources in the embedding learning process, and thus fail to optimally take advantage of both data sources. Moreover, the additional samples generated from the anchors create some significant overhead in terms of computations and learning time, in particular when a large number of additional samples being fed to a stochastic optimization process [144]. To address these issues, we propose JOINER to jointly learn text and KG embeddings via regularization. JOINER can flexibly control the amount of information shared between the two data sources in the embedding learning process with significantly less computational overhead.

In addition, there are some other embedding methods combining text and KGs focusing on improving one using the other, but they do not tackle the same problem as ours. On the one hand, KG embeddings can be improved using textual relations (i.e., full lexicalized dependency paths) [122] or sentence/paragraph embeddings [142]. On the other hand, word embeddings can be enhanced by considering explicit relations from semantic knowledge [147] or word concepts from KGs [25]. However, these methods output either word embeddings or entity/relation embeddings.

Finally, regularization techniques are widely used for joint learning from multiple data sources [145, 24]. To the best of our knowledge, for joint learning from text and KGs, only one recent work KADE [14] adopts regularization techniques to align entities with *documents* using heuristically defined relatedness between entities and documents, which differs from our goal (aligning entities with *words*). Meanwhile, it outputs document embeddings rather than word embeddings, and thus fails to support the analogy reasoning task.

3.3 *JOINER*

3.3.1 Model

Our proposed joint model is built on top of a text model for learning word embeddings and a KG model for learning entity/relation embeddings. In this chapter, we adopt the same individual text and KG models as in [131], which is a representative joint text and KG embedding method generating additional learning samples from a set of anchors. Subsequently, *by comparing with [131] in the experiments, the superiority of using regularization when jointly learning the embeddings can be clearly verified*, as we discount the effect of the individual text and KG models. However, we also note that our joint embedding learning method using regularization is not limited to any specific text and KG model; it can incorporate more sophisticated text or KG models (more discussion on this point later). In the following, we start by describing the individual text and KG models, followed by our joint model.

Text model. Our text model learns word embeddings by capturing the co-occurrence of words observed in a text corpus $\mathscr{D}$. We adopt the same text model as in [131]. Specifically, let $\mathscr{V}$ denote the word vocabulary of the text corpus[1]. The conditional probability of a target word w appearing close to a context word v (within a context window of a certain length) is defined as follows:

$$p(w|v) = \frac{\exp\{s(\mathbf{w},\mathbf{v})\}}{\sum_{v' \in V} \exp\{s(\mathbf{w},\mathbf{v}')\}} \tag{3.1}$$

where $s(\mathbf{w},\mathbf{v})$ is a scoring function evaluating the co-occurrence of two words w and v based on their embeddings. It is defined as $s(\mathbf{w},\mathbf{v}) = b - \frac{1}{2}||\mathbf{w}-\mathbf{v}||^2$, where b is a constant margin used for better numerical stability in the learning process. Subsequently, the objective of the text model is to maximize the likelihood of the co-occurrence of pairs of words in the whole text corpus:

$$\mathscr{L}_T = \sum_{(w,v)\in\mathscr{C}} \#(w,v) \log p(w|v) \tag{3.2}$$

where $\mathscr{C}$ is the set of unique word pairs co-occurring in a context window of a certain length, and $\#(w,v)$ is the number of times (w,v) appears in the corpus $\mathscr{D}$.

Knowledge Graph model. Our KG model learns entity/relation embeddings by preserving the relations between entities. Specifically, a KG Δ consists of a set of triplets (h,r,t), $h,t \in \mathscr{E}$ and $r \in \mathscr{R}$, where $\mathscr{E}$ and $\mathscr{R}$ refer to the entity and relation vocabularies, respectively. Similar to the text model, we define the conditional probability of observing h given (r,t) in the KG as follows:

$$p(h|r,t) = \frac{\exp\{s(\mathbf{h},\mathbf{r},\mathbf{t})\}}{\sum_{h'\in\mathscr{E}} \exp\{s(\mathbf{h}',\mathbf{r},\mathbf{t})\}} \tag{3.3}$$

where $s(\mathbf{h},\mathbf{r},\mathbf{t})$ is a scoring function evaluating the correctness of the triplet (h,r,t) based on their embeddings. It is defined as $s(\mathbf{h},\mathbf{r},\mathbf{t}) = b - \frac{1}{2}\cdot||\mathbf{h}+\mathbf{r}-\mathbf{t}||^2$. In addition, $p(r|h,t)$ and $p(t|h,r)$ are defined in the same way as $p(h|r,t)$ with the corresponding normalization terms, respectively. Subsequently, the KG model maximizes the likelihood of observing all triplets from the KG:

$$\mathscr{L}_{KG} = \sum_{(h,r,t)\in\Delta} [\log p(h|r,t) + \log p(r|h,t) + \log p(t|h,r)] \tag{3.4}$$

[1]We pre-process the text corpus in order to detect common phrases (e.g., *new york*) such that the vocabulary $\mathscr{V}$ also contains these phrases. For the sake of simplicity, we use the term "word(s)" to refer these words/phrases in $\mathscr{V}$, if not specified otherwise.

Joint Text and KG model using regularization. Our joint model combines the above text and KG models via regularization. Specifically, let $\mathscr{A}$ denote a set of anchors, where each anchor connects a word from the text corpus and an entity from the KG. In this chapter, we adopt two common types of anchors, 1) entity names appeared in text and 2) entity-associated Wikipedia pages, which are widely used in the current literature [131, 39]. First, entity names naturally link entities to words in the text corpus. For each entity $e \in \mathscr{E}$, if its name (label) w_e also appears in our word vocabulary, i.e., $w_e \in V$, we consider (e, w_e) as an anchor. Second, entity-associated Wikipedia pages also link entities to words. A Wikipedia (English) page is often associated with a unique entity in a KG (e.g., Freebase). Subsequently, a Wikipedia anchor (i.e., a word w with a hyperlink to the Wikipedia page) in the text actually links to the page's associated entity e_w; we also considers (e_w, w) as an anchor. Finally, both types of anchors are included in our set of anchors $\mathscr{A}$, and we do not distinguish them in $\mathscr{A}$. Note that we do not consider entity descriptions as anchors [153], as it suffers from a low coverage issue (entity descriptions are not always available in a KG).

Based on the set of anchors $\mathscr{A}$, we are then able to jointly learn the text and KG models not only by capturing the co-occurrence between words in a text corpus and the relation between entities in a KG, but also by minimizing the distance between a word and an entity when the two are linked by an anchor via regularization. To achieve this goal, we insert regularization terms in both the text and KG models to connect them by minimizing the Euclidean distance between a word and an entity if they are connected by an anchor in $\mathscr{A}$. Specifically, for the text model, we redefine the scoring function as follows:

$$s(\mathbf{w},\mathbf{v}) = b - \frac{1}{2}(||\mathbf{w}-\mathbf{v}||^2 + 1_{(v,e_v)\in\mathscr{A}} \cdot \beta \cdot ||\mathbf{v}-\mathbf{e}_v||^2) \tag{3.5}$$

where $1_{(v,e_v)\in\mathscr{A}}$ is an indicator function which is equal to 1 when (v, e_v) is an anchor in $\mathscr{A}$ and to 0 otherwise. β is a regularization parameter which defines the weight of the regularization term $||\mathbf{v}-\mathbf{e}_v||^2$ (more on β below). Using this scoring function for our text model, we are able to not only capture the word co-occurrence (i.e., $||\mathbf{w}-\mathbf{v}||^2$), but also to preserve the similarity between a word and its anchor entity (if any) via the regularization term. Subsequently, we redefine the scoring function for the KG model in a similar way:

$$\begin{aligned} s(\mathbf{h},\mathbf{r},\mathbf{t}) = b - \frac{1}{2}(&||\mathbf{h}+\mathbf{r}-\mathbf{t}||^2 + 1_{(h,w_h)\in\mathscr{A}} \cdot \beta \cdot ||\mathbf{h}-\mathbf{w}_h||^2 \\ &+ 1_{(t,w_t)\in\mathscr{A}} \cdot \beta \cdot ||\mathbf{t}-\mathbf{w}_t||^2) \end{aligned} \tag{3.6}$$

where we insert two regularization terms for head and tail entities, respectively. This scoring function is able to not only capture the relations between entities (i.e., $||\mathbf{h}+\mathbf{r}-\mathbf{t}||^2$), but also preserves the similarity between an entity and its word anchor via the two regularization terms. We use the same regularization parameter β for the KG model as for the text model, as our regularization terms share the same formulation in both the text and KG models.

In our joint learning model, *the regularization parameter β plays a key role to provide the flexibility to control the amount of information shared between the two data sources.* More precisely, β actually defines the weight of the regularization term in the whole scoring function. A smaller value of β implies a lower importance of the regularization term in the scoring function; the learning process will learn less from the set of anchors, meaning that less information is actually shared between the two models in the learning process. In the extreme case where we set $\beta = 0$, our joint learning model degrades to two independent text and KG models. In practice, β needs to be appropriately set to obtain high-quality word and entity/relation embeddings. On one hand, a small value of β may lead to insufficient information sharing between the models in the learning process. On the other hand, too big a value may lead to putting too much emphasis on the set of anchors while insufficiently learning from the two data sources (text and KGs). We investigate the impact of β across different evaluation tasks in our experiments. Below, we present our joint learning process.

3.3.2 Joint Embedding Learning Process

Learning with negative sampling. In practice, the considerable size of the vocabularies $\mathcal{V}$ and $\mathcal{E}$ makes it difficult to compute the normalizer in $p(w|v)$ (Eq. 3.1) and $p(h|r,t)$ (Eq. 3.3) (also for $p(r|h,t)$ and $p(t|h,r)$). To overcome this issue, negative sampling techniques [85] can be adopted to simplify the objective functions $\mathcal{L}_T$ and $\mathcal{L}_{KG}$. Specifically, for a pair of words (w, v), we not only maximize the probability of their co-occurrence, i.e., $p((w,v) \in \mathcal{D}|\mathbf{w},\mathbf{v}) = \sigma(s(\mathbf{w},\mathbf{v}))$, but also maximize the probability of the word w and a randomly sampled negative word v_N not appearing in the corpus D, i.e., $p((w,v_N) \notin \mathcal{D}|\mathbf{w},\mathbf{v}_N) = \sigma(-s(\mathbf{w},\mathbf{v}_N))$, where $\sigma(\cdot)$ is the sigmoid function. In summary, for each pair of words (w, v), we maximize the following objective function:

$$\Theta_w = \log\sigma(s(\mathbf{w},\mathbf{v})) + \gamma\mathbb{E}_{v_N}[\log\sigma(-s(\mathbf{w},\mathbf{v}_N))] \tag{3.7}$$

where $\gamma \in \mathbb{Z}^+$ is the number of negative samples. By applying the same negative sampling technique to the KG model, instead of maximizing $p(h|r,t)$, we maximize the following objective function:

$$\Theta_h = \log\sigma(s(\mathbf{h},\mathbf{r},\mathbf{t})) + \gamma\mathbb{E}_{h_N}[\log\sigma(-s(\mathbf{h}_N,\mathbf{r},\mathbf{t}))] \tag{3.8}$$

where h_N is a randomly sampled negative head entity. We also convert $p(r|h,t)$ and $p(t|h,r)$ in the same way:

$$\Theta_r = \log\sigma(s(\mathbf{h},\mathbf{r},\mathbf{t})) + \gamma\mathbb{E}_{r_N}[\log\sigma(-s(\mathbf{h},\mathbf{r}_N,\mathbf{t}))] \tag{3.9}$$

Algorithm 1 JOINER

```
Require: A text corpus $\mathscr{D}$, a KG $\Delta$, and the corresponding vocabulary of words, entities and
    relations ($\mathscr{V}$, $\mathscr{E}$ and $\mathscr{R}$, respectively)
 1: Initialize word, entity and relation embedding vectors $\mathbf{w}$ ($w \in \mathscr{V}$), $\mathbf{e}$ ($e \in \mathscr{E}$) and $\mathbf{r}$ ($r \in \mathscr{R}$)
 2: repeat
 3:     Sample a batch of word pairs $D_{batch}$ from $\mathscr{D}$
 4:     for $(w, v) \in D_{batch}$ do
 5:         Update $\mathbf{w}, \mathbf{v}$ with the gradients of $\Theta_w$ (Eq.3.7)
 6:     end for
 7:     Sample a batch of triples from $\Delta_{batch}$ from $\Delta$
 8:     for $(h, r, t) \in \Delta_{batch}$ do
 9:         Update $\mathbf{h}, \mathbf{r}, \mathbf{t}, \mathbf{h}_N$ with the gradients of $\Theta_h$(Eq.3.8)
10:         Update $\mathbf{h}, \mathbf{r}, \mathbf{t}, \mathbf{r}_N$ with the gradients of $\Theta_r$(Eq.3.9)
11:         Update $\mathbf{h}, \mathbf{r}, \mathbf{t}, \mathbf{t}_N$ with the gradients of $\Theta_t$(Eq.3.10)
12:     end for
13: until Convergence
```

$$\Theta_t = \log \sigma(s(\mathbf{h}, \mathbf{r}, \mathbf{t})) + \gamma \mathbb{E}_{t_N}[\log \sigma(-s(\mathbf{h}, \mathbf{r}, \mathbf{t}_N))] \tag{3.10}$$

Alternating joint learning process. Based on the above objective functions, our joint learning process alternates between the two data sources. As shown in Algorithm 1, after initializing word/entity/relation embeddings (line 1), we alternatively learn from a batch of word pairs sampled by scanning the text corpus (Line 3-6), and a batch of triplets randomly sampled from the KG (Line 7-12). The batch size is empirically set to 500; in practice, when setting it to a small value (much smaller than the size of the corpora), it has negligible impact on the results. Our learning process is performed using asynchronous stochastic gradient descent (ASGD) until convergence. In practice, we iterate both datasets multiple times to ensure the convergence (see experiment settings for more details).

Figure 3.1 shows the architecture of our proposed method HINGE. Specifically, HINGE is built on top of a text model and a KG model. The text model captures the co-occurrence of words observed in a text corpus, while the KG model learns entity and relation embeddings by preserving the relations between entities. In addition, HINGE defines a joint model that combines the above text and KG models via regularization without generating extra samples.

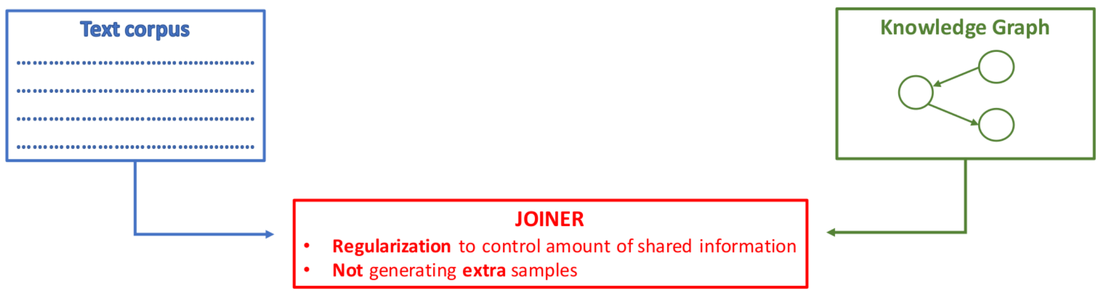

Text corpus
Knowledge Graph
JOINER
Regularization to control amount of shared information
Not generating extra samples

Figure 3.1 – Overview of our proposed method JOINER

3.3.3 Extensibility of JOINER

In this chapter, our JOINER model combines a text model and a KG model, whose scoring functions are both defined using the Euclidean distance. Our regularization term is also defined using the Euclidean distance. Our model is not limited to these two text and KG models, however. It can be modified to incorporate other text and KG models, under the condition that the scoring functions of the two models are defined using the same distance metric (e.g., cosine distance), with which our regularization term should also be defined. In this study, we instantiate JOINER using the Euclidean distance to distinctly demonstrate its advantages over [131] which uses the *same* text *and* KG model as ours but generates additional learning samples for the joint learning process.

3.4 Experimental Evaluation

We evaluate JOINER on three different tasks: analogical reasoning, link prediction in KGs, and relation extraction. Subsequently, we investigate the impact of the regularization parameter β, followed by a runtime evaluation. We start below by presenting our experiment settings.

3.4.1 Experiment Setting

Dataset. Similar to [131], we use the following public text corpora and KGs.

- *Text.* We use the English Wikipedia dump collected in July 2017. We filter out noisy pages (e.g., pages for disambiguation), and perform named entity recognition to detect common phrases, which are included in the vocabulary $\mathcal{V}$note1. After removing rare words, our text corpus $\mathcal{D}$ contains 1,110,804,425 words in total, with 489,861 unique words.

- *Knowledge Graph.* We adopt Freebase in our experiments, which has been widely used by previous work [19, 132, 131, 59]. We extract facts (triplets) from the Freebase dump[2], and for each entity we take its label in English as its entity name for anchor extraction. We consider the top 200K frequent entities and retain all related facts. In the end, our KG contains 199,355 unique entities, 3,442 unique relations and 2,459,553 triplets. For the link prediction task, we randomly sample 20% of the triplets as test data, and the rest as training data. We choose this KG rather than the commonly used FB15k or WN18, because its large scale ensures sufficient anchors linking entities to words.

Based on the above datasets, we extract the set of anchors $\mathscr{A}$, which sum up to 98,812 unique anchors. This represents 20.17% of the unique words in $\mathscr{V}$ and 49.57% of the unique entities in $\mathscr{E}$. Moreover, we find that these anchors actually involve 2,301,519 triplets (93.57% of all triplets) in the KG, and 398,551,199 words (35.88% of all words) in our text corpus. These statistics imply that anchors tend to connect frequent words and entities rather than infrequent ones.

Baselines. We compare our method against a sizable collection of nine state-of-the-art techniques from three categories.

- Word embedding techniques:
  - **SkipGram** [84] is a popular word embedding model that maximizes the co-occurrence probability of a word and its context in a text corpus, where the scoring function is defined using the dot product, i.e., $s(\mathbf{w},\mathbf{v}) = \mathbf{w}\cdot\mathbf{v}$. Our text model is equivalent to the SkipGram model when constraining the embedding norm to be $||\mathbf{w}|| = ||\mathbf{v}|| = 1$, as $\mathbf{w}\cdot\mathbf{v} = 1 - 0.5\cdot\|\mathbf{w}-\mathbf{v}\|^2$. Note that we adopt the Euclidean distance in the scoring function of our text model (rather than using the SkipGram model), in order to be consistent with our KG model (whose scoring function is also defined using the Euclidean distance).
  - **GloVe** [102] is another word embedding techniques that learns directly from aggregated global word-word co-occurrence statistics from a text corpus. The scoring function is defined using the dot product, i.e., $s(\mathbf{w},\mathbf{v}) = \mathbf{w}^\top\cdot\mathbf{v} + b_w + b_v$, where b_w and b_v are scalar biases for a word and its context.
- Knowledge graph embedding techniques:
  - **TransE** [19] is a popular KG embedding model that preserves the relation between two entities as $h + r \approx t$. We also use it as our KG model. However, its objective function is different from ours. Specifically, TransE minimizes a margin-based ranking objective function, i.e., $[d(\mathbf{h}+\mathbf{r},\mathbf{t}) + b - d(\mathbf{h}_N+\mathbf{r},\mathbf{t}_N)]_+$, where $d(\cdot,\cdot)$ can

[2]https://developers.google.com/freebase/

either be the L_1 or L_2-norm. In our experiments, we test different margins b with both L_1 and L_2-norm, the optimal setting being $b = 1$ with the L_1-norm for TransE.

- **TransH** [132] is a KG embedding technique that further extend TransE to better capture multi-mapping relations in a KG. Specifically, TransH introduces the idea of relation-specific hyperplanes, in which, for a given triplet (h, r, t), the embeddings of the entities $\mathbf{h}$ and $\mathbf{t}$ are projected into a relation hyperplane $\mathbf{w}_r$, and their projections are denoted as $\mathbf{h}_\perp = \mathbf{h} - \mathbf{w}_r^\top \mathbf{h}\mathbf{w}_r$ and $\mathbf{t}_\perp = \mathbf{t} - \mathbf{w}_r^\top \mathbf{t}\mathbf{w}_r$, respectively. Thus, $\mathbf{h}_\perp$ and $\mathbf{t}_\perp$, can be connected by a translation vector $\mathbf{v}_r$ on the hyperplane. Similar to TransE, the relation between two entities is preserved as $h_\perp + v_r \approx t_\perp$, and the objective function is defined as $[d(\mathbf{h}_\perp + \mathbf{v}_r, \mathbf{t}_\perp) + b - d(\mathbf{h}_{\perp N} + \mathbf{v}_r, \mathbf{t}_{\perp N})]_+$, where $d(\cdot,\cdot)$ can either be the L_1 or L_2-norm. In our experiments, we also test different margins b with both L_1 and L_2-norm, the optimal setting being $b = 0.25$ with the L_1-norm for TransH.
- **TransD** [59] is a KG embedding technique that decomposes the projection matrix into a product of two vectors. Specifically, additionally mapping vectors $\mathbf{w}_h, \mathbf{w}_t \in \mathscr{R}^d$ and $\mathbf{w}_r \in \mathscr{R}^k$ are introduced along with the entity and relation representations $\mathbf{h}, \mathbf{t} \in \mathscr{R}^d$ and $\mathbf{r} \in \mathscr{R}^k$. The two projection matrices are defined as $\mathbf{M}_r^1 = \mathbf{w}_r\mathbf{w}_h^\top + \mathbf{I}$, and $\mathbf{M}_r^2 = \mathbf{w}_r\mathbf{w}_t^\top + \mathbf{I}$. These two projection matrices are then used to project the head and tail entities with $\mathbf{h}_\perp = \mathbf{M}_r^1\mathbf{h}$ and $\mathbf{t}_\perp = \mathbf{M}_r^2\mathbf{t}$. With the projected entities, the objective function is defined in the same way as in TransR. In our experiments, we also test different margins b with both L_1 and L_2-norm, the optimal setting being $b = 1$ with the L_1-norm for TransD.
- **DistMult** [143] is an efficient semantic matching model using a bilinear scoring function that maps each entity to k-dimensional vector and each relation r to a diagonal matrix $\mathbf{R}_r^{k\times k}$. For a given triplet, the score is computed as $s(h, r, t) = \mathbf{h}^\top \mathbf{R}_r \mathbf{t}$. DistMult also minimizes a margin-based ranking objective function, i.e., $max\{s(h_N, r, t_N) - s(h, r, t) + b, 0\}$. In our experiments, we also test different margins b, the optimal setting being $b = 0.1$ for DistMult.

• Knowledge graph embedding techniques:

- **JointAS** [131] is a representative joint text and KG embedding model using Additional learning Samples generated from two types of anchors. It uses the same text and KG models as ours. It defines two types of anchors that link words and entities and then generate additional learning samples from these anchors. First, for entity names appeared in text, additional learning samples are generated for the KG model by replacing the entity from each triplet (h, r, t) by the corresponding word, e.g., (w_h, r, t) if (h, w_h) is an anchor in $\mathscr{A}$. This type of anchors is called “AN”. For entity-associated Wikipedia pages, additional learning samples are generated for the text model by replacing the word from each word pairs (w, v) by the associated entity, e.g., (w, e_v) if (e_v, v) is an anchor in $\mathscr{A}$. This type of anchors is called "AA". Its joint learning process alternates between the text, the KG and the two types of additionally generated learning samples. As suggested by the authors, we set the margin to its optimal value $b = 7$. Note that according to the selected types of anchors, JointAS can be configured in three settings, i.e., (AN), (AA) and (AN+AA).

We exclude task-specific joint embedding methods such as [39, 141] from our baselines, as they involve task-specific heuristics, and cannot support different evaluation tasks. We also exclude KADE [14], because it aligns entities with documents and thus outputs document embeddings rather than word embeddings, failing to support the analogy reasoning task.

For our JOINER, we search the optimal regularization parameter β from 0.0001 to 0.1 on a log scale, and set the margin $b = 8$. For all the methods, we set the embedding size to 100, the number of negative samples to 10, and the context window size for text models to 5. For all models involving the text corpus, we train embeddings with three epochs on the text corpus (for the joint models, the KG is simultaneously traversed multiple times using the alternating joint learning process). For a fair comparison, we also train TransE, TransH, TransD and DistMult by traversing the KG the same number of times as for the joint models. We train and test five models for each method and report the average results. The code for our model and datasets are made publicly available[3].

3.4.2 Task 1: Analogical Reasoning

The analogical reasoning task is widely used to evaluate word embeddings. It consists of a set of analogies, such as *USA→dollar: Japan→?* that require to predict the missing word (*yen* in this example). We use a popular analogy dataset provided by [85], which contains 19,544 word and 3,218 phrase analogies. To predict the missing word for each analogy (denoted as *A→B:C→?*), we compute the Euclidean distance between $\mathbf{C} + (\mathbf{B} - \mathbf{A})$ and all the word embeddings, and pick the word with the minimum Euclidean distance as the predicted word. We report the accuracy over the whole analogy dataset.

Table 3.1 shows the results. First, we observe that JOINER outperforms all baselines by achieving the highest accuracy (with a 2.8% improvement over the best performing baseline SkipGram). Here we also report the corresponding optimal regularization parameter $\beta = 0.001$ (we will later show that the optimal β varies across different tasks). Second, the best-performing word embedding technique, SkipGram, surprisingly beats JointAS on all configurations, which is the opposite of the results from [131]. This is probably due to the fact that our text and KG datasets are more tightly connected by anchors than the dataset used in [131], in particular for

[3]https://github.com/eXascaleInfolab/JOINER_code/

Table 3.1 – Analogical reasoning performance

Method	Accuracy (%)
SkipGram	56.3
GloVe	47.2
JointAS (AN)	38.9
JointAS (AA)	50.7
JointAS (AN+AA)	40.5
JOINER (β=0.001)	**57.9**

the text corpus. More precisely, 35.88% of all words in our text corpus and 93.57% of all triplets in our KG are connected by anchors, while these two percentages are 2.76% and 40% in the dataset used by [131], respectively. Subsequently, on our dataset, JointAS generates too many additional learning samples which dominate the learning process, leading to degraded results. In contrast, our JOINER approach allows to control the extent to which the embeddings are jointly learnt from the two data sources, and can hence avoid learning too much from the anchors by setting the regularization parameter β accordingly. Finally, we observe that JointAS results are sensitive to the anchor configuration; AA is the best anchor type for the analogical reasoning task, while AN actually pollutes the results. Similar findings are also reported in [131]. However, choosing appropriate anchors is not straightforward for JointAS; we will see later that different types of anchors may have varying impact on different tasks.

3.4.3 Task 2: Link Prediction

The link prediction task is widely used for KG completion [19]. It suggests new triplet (h, r, t) with h or t missing. In other words, it predicts t given (h, r) or predicts h given (r, t). To implement this task, we follow the same evaluation protocol as in TransE [19]. More precisely, when predicting t given (h, r), we compute the scores for $||\mathbf{h} + \mathbf{r} - \mathbf{e}||$, where $e \in \mathscr{E}$; by ranking the scores in ascending order, we generate a predicted ranking list of entities. We then report $Hits$@10, which measures the percentage of the predictions whose top-10 entities contain the ground truth entity t, over all triplets in the test dataset. We call this setting "raw". Moreover, when predicting t given (h, r), other tail entities may co-exist in the KG, i.e., $T' = \{t' | t' \neq t, (h, r, t') \in \Delta\}$. In this case, ranking t' in front of t should not be counted as an error. To avoid this case, [19] suggests filtering out these entities (T') before generating the ranked list of entities. We call this setting "filtered". The above evaluation protocol also applies to predict h given (r, t).

Table 3.2 shows the results. First, we observe that JOINER achieves the best performance in most cases (except when predicting tail with the "raw" setting, JointAS (AN) is slightly better). In the "filtered" setting (which is more reasonable for this task), JOINER outperforms the

Table 3.2 – Link prediction performance

Method	Hits@10 Raw (%)		Hits@10 Filtered (%)	
	Head	Tail	Head	Tail
TransE	37.1	44.6	41.7	49.2
TransH	38.7	45.4	46.1	52.6
TransD	38.9	43.1	48.6	52.7
DistMult	35.7	36.8	44.5	45.4
JoinAS (AN)	41.4	**49.1**	53.9	60.8
JoinAS (AA)	40.1	46.3	52.4	58.0
JoinAS (AN+AA)	40.9	48.7	52.5	59.7
JOINER (β=0.001)	**42.2**	48.9	**56.2**	**62.1**

best-performing baseline JointAS (AN) with 2.1% and 4.3% improvement in predicting tail and head, respectively. Second, we find that JointAS with all configurations achieves better results than TransE, TransH, TransD and DistMult, showing the effectiveness of joint embeddings for this task. Moreover, for JointAS, we find that the anchor configuration AN yields higher performance than AA on this task, while we observe the opposite (i.e., AA shows better results than AN) on the analogical reasoning task. This implies that JointAS is less robust to different tasks, as the selection of anchors has a strong impact on the results. In contrast, our JOINER is less sensitive to anchor selection, as optimal performance can be achieved by tuning the regularization parameter.

3.4.4 Task 3: Relation Extraction

Relation extraction finds relations between two entities from text [135]. Given two detected entities in a sequence of text, it assigns a specific relation (if any) to the pair of entities based on their contexts (features) in text. We adopt the same evaluation protocol as used by JointAS [131]. Specifically, we first use a basic relation extractor [89] to generate a set of candidate relations r_i for each pair of entities (h, t) with their estimated probability $Pr_{mintz}(r_i)$ (according to text features). Then, we compute the Euclidean norms for the corresponding candidate triplets based on the embeddings, and convert the Euclidean norms into probabilities using a softmax function, i.e., $Pr_{JOINER}(r_i) = 1 - softmax(||\mathbf{h} + \mathbf{r}_i - \mathbf{e}||)$. Finally, for each candidate relation, we linearly combine the two probabilities as $\alpha \cdot Pr_{mintz}(r_i) + (1-\alpha) \cdot Pr_{JOINER}(r_i)$ and pick the relation with the largest probability. Similar to [131], the optimal α is searched from 0 to 1 with a step of 0.025. In this task, we use a relation-labeled dataset NYT+FB [106], and randomly split it into 50% training and 50% test sets. The basic extractor is trained on the training data, while entity/relation embeddings are trained using our own Wikipedia+Freebase datasets. We report the average accuracy over the test dataset.

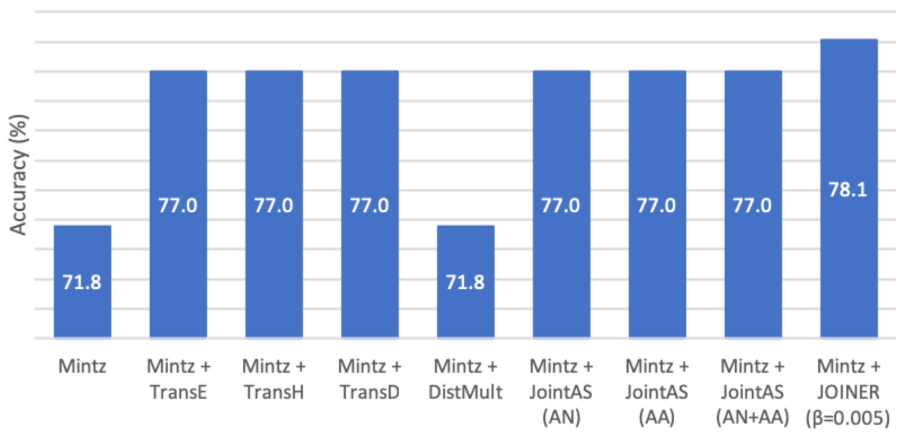
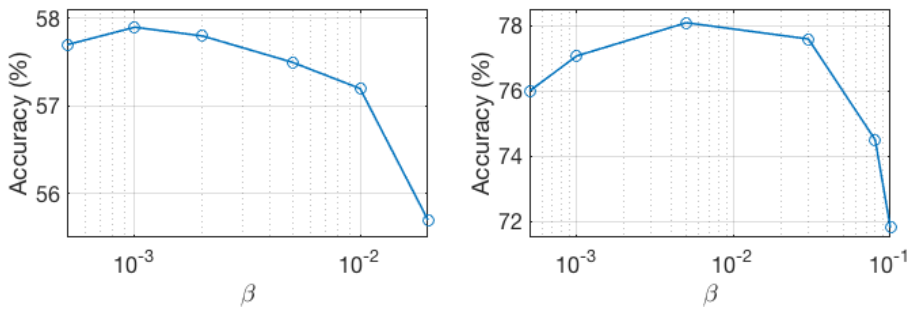


Figure 3.2 – Relation extraction performance

Figure 3.2 shows the results. First, we observe that compared to the basic extractor Mintz, using embeddings can significantly improve the relation extraction performance; a similar observation has also been reported by [135]. Second, compared to the KG embedding techniques TransE, TransH, TransD and DistMult, JointAS achieves the same performance, while JOINER further outperforms JointAS by 1.4%. Note that all KG embeddings baselines and JointAS show the same results on this task, as we report the optimal accuracy by tuning the weights for the weighted sum process, which indeed weakens the impact of individual techniques.

3.4.5 Impact of Regularization Parameter

The regularization parameter β controls to what extent we *jointly* learn embeddings from the text and KG. A smaller value of β implies that we learn less from the set of anchors that link the two data sources. In this experiment, by varying β on a log scale, we report the performance for the analogical reasoning and relation extraction tasks in Figure 3.3. First, we observe that neither a too small nor a too big value of β results in optimal performance. On one hand, a too small value of β leads to insufficient information sharing between the two data sources in the learning process, resulting in suboptimal performance. On the other hand, too big a value over-considers the anchors which dominate the learning process, thus leading to degraded performance. Our method JOINER provides hence the flexibility to achieve the best performance by tuning β. Second, we find the optimal β varies across different tasks, i.e., 0.001 for analogical reasoning and 0.005 for relation extraction, which can serve as a guideline for future work on joint text and KG embeddings.



(a) Analogical reasoning (b) Relation extraction

Figure 3.3 – Impact of regularization parameter β

Table 3.3 – Runtime performance

Method	Learning time (in hours)
Text+KG (without joint learning)	14.3
JointAS (AN+AA)	37.6
JOINER (ours)	26.8

3.4.6 Runtime Performance

We evaluate the efficiency of JOINER by comparing the learning time of different embedding methods. For a fair comparison, we focus on JointAS (AN+AA) and JOINER, as they share the same text and KG models, and also the same set of anchors. To give a reference, we also report the total learning time of the text+KG models (without joint learning). We report the embedding learning time[4] in Table 3.3. Unsurprisingly, compared to Text+KG (without joint learning), both the joint embedding methods create some overhead, i.e., 163% and 87% additional learning time for JointAS and JOINER, respectively. More importantly, compared to JointAS which creates additional learning samples for joint learning, JOINER using regularization yields significantly less overhead (76% less learning time overhead).

3.5 Conclusions

This chapter revisited text and KG joint embeddings and introduced JOINER, a novel joint text and KG embedding method providing the flexibility to control the amount of information shared between the two data sources during the joint learning process via regularization. Without additionally generated learning samples, it is also computationally efficient. Extensive experiments showed that compared to JointAS, a state-of-the-art joint text and KG embedding method generating additional learning samples from a set of anchors, JOINER yields better re-

[4]Measured on a server with two CPUs (Intel Xeon E5-2620V4@2.10GHz) and 128G RAM using 30 threads.

sults (with 1.4%-4.3% improvement on various evaluation tasks) while significantly improving runtime performance (76% less overhead). [38] exploited the persona graph decomposition [37] to develop an embedding algorithm that represents nodes in the graph with multiple vectors. In future work, improving the embedding representation using the persona graph concept for learning multiple representations of the nodes in the Knowledge Graph may further improve the performance of our proposed method JOINER.

In the next chapter we propose a novel method to learn hyper-relational KG embeddings. In a hyper-relational fact, each fact contains not only a base triplet (h, r, t), but also the associated key-value pairs. In order to learn embeddings from this data structure, we propose HINGE, a Convolutional Neural Network based architecture that captures not only the primary structural information of the KG encoded in the triplets, but also the correlation between each triplet and its associated key-value pairs. We compare HINGE against a collection of state-of-the-art methods on various link prediction tasks showing the superiority of our proposed method.

4 Beyond Triplets: Hyper-Relational Knowledge Graph Embedding for Link Prediction

Knowledge Graph embeddings are a powerful tool for predicting missing links in KGs. Existing techniques typically represent a KG as a set of triplets, where each triplet (h, r, t) links two entities h and t through a relation r, and learn entity/relation embeddings from such triplets while preserving such a structure. However, this triplet representation oversimplifies the complex nature of the data stored in the KG, in particular for hyper-relational facts, where each fact contains not only a triplet (h, r, t), but also the associated key-value pairs (k, v). Even though a few recent techniques tried to learn from such data by transforming a hyper-relational fact into an n-ary representation (i.e., a set of key-value pairs only without triplets), they result in suboptimal models as they are unaware of the triplet structure, which serves as the fundamental data structure in modern KGs and preserves the essential information for link prediction. To address this issue, we propose HINGE, a hyper-relational KG embedding model, which directly learns from hyper-relational facts in a KG. HINGE captures not only the primary structural information of the KG encoded in the triplets, but also the correlation between each triplet and its associated key-value pairs. Our extensive evaluation shows the superiority of HINGE on various link prediction tasks over KGs. In particular, HINGE consistently outperforms not only the KG embedding methods learning from triplets only (by 0.81-41.45% depending on the link prediction tasks and settings), but also the methods learning from hyper-relational facts using the n-ary representation (by 13.2-84.1%).

4.1 Introduction

Knowledge Graphs, such as Freebase [16], Google's Knowledge Graph [45] or Wikidata [136], have become a key asset powering a large range of Web applications ranging from semantic search [138] to question-answering [146], query expansion [46], or recommendation systems [151]. KGs are typically represented through a set of triplets; each triplet *head, relation, tail,* or (h, r, t) for short, encodes a relation connecting a head entity and a tail entity, such as

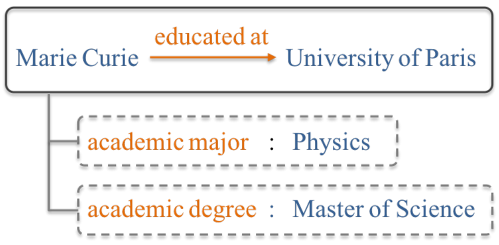


Figure 4.1 – A hyper-relational fact

Switzerland (head) *hasCurrency* (relation) *Swiss franc* (tail). While modern KGs typically contain rich and high-quality data, they are also known to suffer from an incompleteness issue, i.e., missing facts. For example, 71% of all people from Freebase have no *place of birth* [93], even though this is a mandatory property of the schema [134]. Against this background, Knowledge Graph completion problems have been widely studied. A central problem in this context is to predict the missing links in a KG (a.k.a. link prediction). More precisely, given two elements of a triplet, the task is to predict the missing one, such as $(?, r, t)$, $(h, ?, t)$ or $(h, r, ?)$, where the question mark represents the missing entity/relation.

In the current literature, Knowledge Graph embeddings have been shown as a powerful tool for such link predictions [129]. The key idea of KG embeddings is to learn a latent (and low-dimensional) vector representation of entities/relations (i.e., entity/relation embeddings) from a set of triplets in a KG, while preserving the essential information for link prediction in the KG. For example, TransE [19], a typical KG embedding technique, models a relation as a vector-plus operation between two entities $h + r \approx t$; subsequently, when predicting the missing links, a new fact can be asserted by evaluating $||h + r - t||$.

Despite its broad adoption, the *triple-based* representation of a KG often oversimplifies the complex nature of the data stored in the KG, in particular for hyper-relational data (a.k.a. multi-fold [133] or n-ary [47] relational data), where each fact contains multiple relations and entities. Figure 4.1 shows an example about Marie Curie's education from Wikidata: it contains a triplet: (h, r, t) {*Marie Curie, educated at, University of Paris*}, as well as further information associated with the triplet, represented as key-value (relation-entity) pairs[1] (k, v) including {*academic major, physics*}, {*academic degree, Master of Science*}, etc.

Such hyper-relational data is ubiquitous in KGs. Taking Freebase as an example, more than 30% of its entities are involved in such hyper-relational facts [133]. When learning KG embeddings, those hyper-relational facts have to be transformed into triplets by either 1) keeping the triplet only from a hyper-relational fact, e.g., (h, r, t) in the above example, causing irreversible

[1] We use the term key-value (k, v) denoting a relation-entity pair here to emphasize its difference from the triplet (h, r, t), even though h, t and v are entities while r and k are relations.

education_head	:	Marie Curie
education_tail	:	University of Paris
education_major	:	Physics
education_degree	:	Master of Science

Figure 4.2 – N-ary representation of the fact

information loss; 2) creating additional triplets from a hyper-relational fact via reification [20], where an artificial entity is used to represent the triplet $q := (h, r, t)$ and additional triplets are created as (q, k, v); or 3) creating additional triplets from a hyper-relational fact via relation paths [133], where we link h to v via an artificial relation $\widehat{rk}$ representing a relation path $r \rightarrow k$, resulting in additional triplets $(h, \widehat{rk}, v)$. Although the latter two transformations preserve the complete information of a hyper-relational fact, the extra entities/relations they artificially introduce confuse KG embeddings methods, preventing them from capturing key structural properties of the original input graph (see Section 4.4.3 for more detail).

Against this background, in this chapter, we investigate the problem of hyper-relational Knowledge Graph embedding. In the current literature, a few recent studies consider such hyper-relational data [133, 152, 47]. These works consider a set of relations as a so-called n-ary (or multi-fold) relation, while the associated entities then become instances of that relation. Figure 4.2 shows an n-ary representation of the above example about Marie Curie. An n-ary relation "education" is extracted from the hyper-relational fact, containing the following four relations: *education_head*, *education_tail*, *education_major* and *education_degree*; the hyper-relational fact is then represented as a set of key-value (relation-entity) pairs only, i.e., {*education_head*:*Marie Curie*, *education_tail*:*University of Paris*, *education_major*:*Physics*, *education_degree*:*Master of Science*}. Subsequently, using such an n-ary representation, existing approaches to link prediction either learn to model the relatedness of entities [133, 152], or learn from the relatedness between entity/relation pairs [47]. However, the n-ary representation of a hyper-relational fact (as a set of key-value pairs without triplets) treats each key-value pair in the fact equally, which is not compatible with the schema used by modern KGs. Specifically, as triplets still serve as the fundamental data structure in modern KGs, they preserve the essential information for link prediction. In other words, key-value pairs (k, v) on a hyper-relational fact should not be treated identically as triplet (h, r, t). In our experiments, we conduct a hypothesis test and experimentally show that embeddings learnt from the original triplets (from a set hyper-relational facts) consistently and significantly outperform (by 58.3-125.27%) embeddings learnt from the same number of triplets created by a *null model*, where one triplet is extracted from the n-ary representation of each hyper-relational fact (represented a set of key-value pairs) via a randomly sampled relation path [133] (see Section 4.4.2

for more detail). Such an observation suggests that it is highly beneficial to directly capture the structure of the triplets in hyper-relational facts.

Motivated by the above observation, we propose in this chapter HINGE, a Hyper-relatIonal kNowledge Graph Embedding model. HINGE is designed to directly learn from hyper-relational facts in a KG, capturing not only the primary structural information of the KG encoded in the triplets, but also the correlation between each triplet and its associated key-value pairs (if any). More precisely, for each hyper-relational fact, we first design two convolutional neural network pipelines, which learn 1) from the triplet (h,r,t), generating a triple-wise relatedness feature vector for h, r and t, and 2) from each key-value pair (k,v) associated with the triplet together with the triplet itself, generating a quintuple-wise relatedness feature vector between h, r, t, k and v, respectively. Afterward, we compute the overall relatedness feature vector for the hyper-relational fact by taking the minimum value along each feature dimension over the triple-wise relatedness feature vector and all the quintuple-wise relatedness feature vectors. The basic assumption behind this operation is that for a valid hyper-relational fact, both the relatedness for the triplet (h,r,t) and the relatedness between each key-value pair (k,v) and the triplet should be high; subsequently, the minimum relatedness along each feature dimension is expected to be high. Finally, we use a fully connected projection to output the predicted score from the overall relatedness feature vector for the input hyper-relational fact.

Our contributions are hence four-fold:

- We investigate the problem of hyper-relational Knowledge Graph embedding, where each fact contains not only a triplet, but also associated key-value pairs;
- We discuss a key limitation of a commonly used representation scheme for hyper-relational data (i.e., using a set of key-value pairs only). We empirically verify its limitation, showing that triplets serve as the fundamental data structure underpinning modern KGs and indeed encode the essential information for link prediction;
- We introduce HINGE, a Hyper-relatIonal kNowledge Graph Embedding model, designed to directly learn from hyper-relational facts in a KG, capturing not only the primary structural information of the KG encoded in the triplets, but also the correlation between each triplet and its associated key-value pairs;
- We conduct a thorough evaluation of our method compared to a sizable collection of baselines on two real-world KG datasets. Our results show that compared to nine state-of-the-art KG embedding methods learning from triplets only, HINGE consistently achieves better performance, yielding an improvement of 0.81-41.45% on various link prediction tasks (i.e., head/tail or relation prediction) with different data transformation settings (e.g., keeping triplet only, via reification or relation paths) over the best-performing baseline methods on

individual tasks. Moreover, compared to methods learning from hyper-relational facts using an n-ary representation, HINGE shows improvements of 13.2-84.1% across different link prediction tasks over the best-performing baseline methods on individual tasks.

4.2 Related Work

Graph embeddings have become a key paradigm to learn representations of nodes in a graph and facilitate downstream graph analysis tasks [22, 55, 144]. As a specific type of graphs, Knowledge Graphs contain both semantic-enriched nodes (entities) and edges (relations). Therefore, KG embedding techniques learn representations of entities and relations in a KG by preserving the relations between entities [129]. In the following, we briefly discuss existing KG Embedding techniques learning from 1) triplets only, 2) triplets with other data, and 3) hyper-relational facts.

4.2.1 KG Embeddings from Triplets

In the current literature, most of the existing KG embedding techniques learn from a set of triplets (h, r, t) extracted from an input KG. These techniques can be classified into two broad categories, i.e., translational distance models and semantic matching models [108]. First, translational distance models exploit distance-based scoring functions to create the embeddings. One representative model of this family is TransE [19], which creates embeddings from triplets (h, r, t) such that the relation between the head and tail entities are preserved as $h + r \approx t$. Several works further improve TransE to capture richer KG structures—such as multi-mapping relations (one-to-many, many-to-one, or many-to-many)—using more sophisticated scoring function involving relation-specific hyperplanes [132] or spaces [74, 59, 36], for example. Second, semantic matching models exploit similarity-based scoring functions. One typical model in that context is RESCAL [95]. It represents each entity as a vector and each relation as a matrix, and uses a bilinear function to model the relation between two entities. Several works also extend RESCAL by putting a specific focus on reducing the model complexity [143], by capturing asymmetric relations [124], or by modeling non-linear relations using neural networks [117, 29, 113, 10, 92].

However, representing a KG *using triplets only* often oversimplifies the complex nature of the data stored in the KG, in particular for hyper-relational data, where each fact contains multiple relations and entities (see example above). Even though a hyper-relational fact can be transformed to triplets by either keeping the triplets only or creating additional triplets via reification [20] or relation paths [133], none of these transformations is ideal for knowledge graph embeddings, as the former transformation setting incurs irreversible information loss in the KG embeddings while the latter two settings generate additional entities/relations distracting the KG embedding method from capturing the essential information for link prediction (see our experimental results in Section 4.4.3 for more detail on this). Therefore, it would be highly beneficial to learn KG embeddings directly from such hyper-relational facts.

4.2.2 KG Embeddings from Triplets with other Data

We also note that there are a few works on KG embeddings considering other data together with the triplets. According to the sources of such data, these works can be classified into two categories, i.e., data in the KG and third-party data. First, except triplets linking entities via relations, other data contained in a KG can be incorporated into KG embeddings. For example, multi-modal attributes associated with entities (a.k.a. literals), such as non-discrete numerical literals [44, 121] or text literal [66], have been shown to improve the KG embeddings on various tasks; images associated with entities have also been used to improve entity matching tasks (matching entities across different KGs) [77]. These works mainly focus on using multi-modal data to enrich the representation of entities, while triplets remain the only relational representation between entities, which differs from our work focusing on hyper-relational facts. Second, some related techniques learn entity/relation embeddings from triplets in a KG jointly with third-party data sources, in particular with text (e.g., Wikipedia articles) [131, 153, 39, 141, 53, 122, 142, 147, 25, 110]. These works focus on combining the advantages of a KG with further (textual) data sources to learn both entity/relation and word embeddings simultaneously, which is different from our work learning from a KG only while considering hyper-relational facts.

4.2.3 KG Embeddings from Hyper-Relational Facts

Some recent works on KG embeddings started to consider hyper-relational data (a.k.a. multi-fold or n-ary relational data) [133, 152, 47]. More precisely, these works transform a hyper-relational fact into an n-ary representation, i.e., a set of key-value (relation-entity) pairs while completely avoiding triplets. For example, in [47], a hyper-relational fact (h, r, t) with (k, v) is transformed into $\{r_h{:}h,\ r_t{:}t,\ k{:}v\}$ by converting the relation r into two keys r_h and r_t, associated with head h and tail t, respectively. Using this representation, these works learn the relatedness between entity/relation pairs for predicting missing links in KGs. Specifically, m-TransH [133] models the interaction between entities involved in each fact in order to perform link prediction on missing entities. RAE [152] further extends m-TransH by considering the relatedness between entities in each fact for performing instance reconstruction, i.e., predicting one or multiple missing entities in a fact. As these two works capture only the relatedness between entities and can thus only predict missing entities, NaLP [47] was later proposed to model the relatedness between key-value (relation-entity) pairs contained in each fact, which enables the prediction of either a missing key (relation) or a missing value (entity).

However, transforming a hyper-relational fact into an n-ary representation (i.e., as a set of key-value pairs) is inherently incompatible with the schema used by modern KGs, where triplets still serve as the fundamental data structure. In other words, key-value pairs (k, v) on a hyper-relational fact should not be treated identically to triplets (h, r, t), as the latter actually preserves the essential information for link prediction in the KGs. We also empirically verify this point in our experiments (see Section 4.4.2 for more detail). Therefore, in this chapter, we design HINGE to directly learn from the triplets even for hyper-relational facts, while simultaneously learning from the associated key-value pairs also.

4.3 Learning from Hyper-Relational Facts

In this section, we introduce HINGE, a hyper-relational KG embedding model learning directly from hyper-relational facts. We introduce a couple of formal definitions:

Hyper-relational fact: A hyper-relational fact contains a triplet (h, r, t) and a set of associated key-value pairs (k_i, v_i), $i = 1, ..., n$.

Triple fact: A triple fact contains a triplet (h, r, t) only.

Based on these definitions, we start below by discussing our main design principle and goals, before presenting our proposed model in detail.

4.3.1 Design Principle and Goals

As triplets are indeed the fundamental data structure in modern KGs and thus preserve the essential information for link prediction in the KGs, our main design principle is to, on one hand, learn embeddings directly from this *primary* data source (triplets) in order to preserve the essential information for link prediction in the KG to a maximum extent, while on the other hand also learning the relatedness between a triplet and its associated key-value pairs (if any). Following this principle, we set the three following goals:

I) *Effectively learning from both triple facts and hyper-relational facts.* Specifically, as not all facts are hyper-relational in a KG (e.g., 30% of entities are involved in such hyper-relational facts in Freebase [133]), the embedding model should be highly effective at learning both from triple facts and from hyper-relational facts. In other words, the incorporation of key-value pairs from hyper-relational facts should not distract the main learning process from the triplets.

II) *Leveraging key-value pairs when learning from the triplets in hyper-relational facts.* For each hyper-relational fact, the triplet is associated with a set of key-value pairs providing

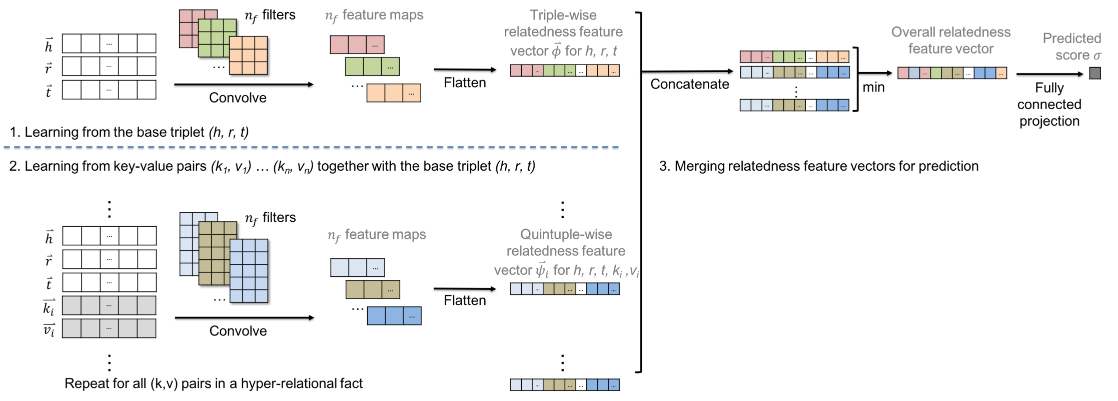

n_f filters
n_f feature maps
Triple-wise relatedness feature vector $\vec{\phi}$ for h, r, t
Convolve
Flatten
Concatenate
min
Overall relatedness feature vector
Predicted score σ
Fully connected projection
1. Learning from the base triplet (h, r, t)
2. Learning from key-value pairs (k_1, v_1) … (k_n, v_n) together with the base triplet (h, r, t)
3. Merging relatedness feature vectors for prediction
Quintuple-wise relatedness feature vector $\vec{\psi}_i$ for h, r, t, k_i, v_i
Repeat for all (k,v) pairs in a hyper-relational fact

Figure 4.3 – Overview of our proposed method HINGE

further information about the triplet. As the triplet remains the primary data source for modeling the hyper-relational fact, the embedding model should be designed to leverage key-value pairs to *improve* the learning process from the triplet.

III) *Learning from an arbitrary number of key-value pairs from a hyper-relational fact.* As the number of key-value pairs involved in hyper-relational facts varies, the embedding model should be able to effectively handle an arbitrary number of key-value pairs in a hyper-relational fact.

We built HINGE with those goals in mind, in order to effectively learn embeddings from both triple and hyper-relational facts.

4.3.2 HINGE

Figure 4.3 illustrates our proposed model HINGE. It is designed to directly learn from hyper-relational facts in a KG, capturing not only the primary structural information of the KG encoded in the triplets, but also the relatedness between each triplet and its associated key-value pairs (if any). More precisely, HINGE consists of three parts. For each hyper-relational fact containing a triplet (h, r, t) and associated key-value pairs (k_i, v_i), $i = 1, ..., n$, it 1) learns from the triplet (h, r, t), generating a triple-wise relatedness feature vector for h, r and t, and 2) learns from each key-value pair (k, v) associated with the triplet together with the triplet itself, generating the quintuple-wise relatedness feature vector between h, r, t, k and v, respectively. Afterward, it 3) merges these relatedness feature vectors to generate a final prediction score for the input hyper-relational fact. In the following, we present the details of these three modules.

Learning from Triplets

In both triple or hyper-relational facts, triplets encode the primary structural information of a KG, and thus capture essential information for link prediction in the KG. To learn from a triplet (h, r, t), we resort to a Convolutional Neural Network (CNN) to model the intrinsic interaction between the three elements in the triplet, i.e., head h, relation r and tail t, in order to generate a triple-wise relatedness feature vector.

More precisely, as shown in Figure 4.3, we start by concatenating the three corresponding embedding vectors $\vec{h}, \vec{r}, \vec{t} \in \mathbb{R}^K$ (K is the embedding dimension) into an "image" $T \in \mathbb{R}^{3\times K}$, which is the input for a 2D convolutional layer with n_f filters of size 3×3. The filter of size 3 is chosen to capture the triple-wise relatedness between $\vec{h}, \vec{r}$ and $\vec{t}$. This layer returns n_f feature maps of size $K-2$, which are then flattened into a triple-wise relatedness vector $\vec{\phi} \in \mathbb{R}^{1\times n_f(K-2)}$. This relatedness vector $\vec{\phi}$ can be used to characterize the plausibility of a triplet (h, r, t) of being true.

Learning from Key-Value Pairs

Key-value pairs contain further information describing the associated triplet in a hyper-relational fact, which suggests that learning from key-value pairs should be coupled with the corresponding triplet. Therefore, for each key-value pair (k_i, v_i) associated with the triplet (h, r, t) in a hyper-relational fact, we also resort to a CNN to capture the interaction between each elements in the triplet and the key-value pair, i.e., h, r, t, k_i and v_i, in order to generate a quintuple-wise relatedness feature vector.

As shown in Figure 4.3 and similar to the case of learning from triplets, we first concatenate the five corresponding embedding vectors $\vec{h}, \vec{r}, \vec{t}, \vec{k}_i, \vec{v}_i \in \mathbb{R}^K$ into an "image" $H \in \mathbb{R}^{5\times K}$, and feed H to a 2D convolutional layer with n_f filters of size 5×3. The first dimension size 5 of the filter here is chosen to capture the quintuple-wise relatedness between $\vec{h}$, $\vec{r}$, $\vec{t}$, $\vec{k}_i$ and $\vec{v}_i$; the second dimension size 3 is chosen to match the filter size of the CNN for triplets, in order to merge the resulting relatedness feature vectors (see below). This layer returns n_f feature maps of size $K-2$, which is then flattened into the quintuple-wise relatedness vector $\vec{\psi}_i \in \mathbb{R}^{1\times n_f(K-2)}$. This relatedness vector $\vec{\psi}_i$ can be used to characterize the plausibility of the triplet (h, r, t) associated with the key-value pair (k_i, v_i) being a true fact. This process is repeated for each key-value pair (k_i, v_i), $i = 1, .., n$, in the input hyper-relational fact containing n key-value pairs, resulting in n quintuple-wise relatedness vectors $\vec{\psi}_i$, $i = 1, .., n$. Note that this module is not used for triple facts, as they do not contain any key-value pair.

Merging Relatedness Feature Vectors for Prediction

In the previous two modules, for each hyper-relational fact, one triple-wise relatedness vector $\vec{\phi}$ is generated from the triplet (h, r, t) while n quintuple-wise relatedness vectors $\vec{\psi}_i$ are generated from the n key-value pairs together with the triplet. We now wish to merge these relatedness feature vectors in order to return a final score for the input hyper-relational facts.

To achieve this goal, we first compute the overall relatedness feature vector by taking the minimum value along each feature dimension over the triple-wise relatedness feature vector and all the quintuple-wise relatedness feature vectors. We concatenate the triple-wise relatedness feature vectors $\vec{\phi}$ and the n quintuple-wise relatedness vectors $\vec{\psi}_i$ into a matrix of size $(n+1) \times n_f(K-2)$, and compute the minimum value of this matrix along each column, resulting in the overall relatedness feature vector. The underlying assumption for this operation is that for a valid hyper-relational fact, both 1) the relatedness for the triplet (h, r, t) and 2) the relatedness between each key-value pair (k, v) and the triplet (h, r, t) should be high. While each entry of a triple-wise (or quintuple-wise) relatedness feature vector actually measures the relatedness between h, r, t (or between $h,\ r,\ t,\ k_i\ v_i$) under a certain filter, the minimum relatedness along each feature dimension is expected to be high. Similar ideas have also been successfully applied by previous works to merge relatedness scores in a neural network [47]. Finally, we use a fully connected projection to output the predicted score σ from the overall relatedness feature vector for the input hyper-relational fact.

4.3.3 HINGE and Design Goals

In this section, we discuss how our proposed model HINGE fits the goals we set above in Section 4.3.1.

First, to fit goal I, we adopt two separate neural network pipelines to learn from 1) the triplet, and from 2) the key-value pairs together with the triplet. In case of triple facts, only the first module is used while the second module is not activated. In case of hyper-relational facts (with key-value pairs), both the first and the second modules are used. Subsequently, the CNN for triplets (used in the first module) is independent from the key-value pairs, which allows it to capture the primary structural information of the KG encoded in the triplets, and thus to preserve the essential information for link prediction in a KG to a maximum extent.

Second, to fit goal II, we merge, via a "min" operation, the relatedness feature vectors learnt from 1) the triplet, and 2) the key-value pairs together with the triplet. Such an operation ensures that the CNN for key-value pairs (used in the second module) effectively help to evaluate the plausibility of a hyper-relational fact. More precisely, considering a hyper-relational fact containing a triplet and an associated key-value pair, we obtain a triple-wise relatedness vector $\vec{\phi}$ and a quintuple-wise relatedness vectors $\vec{\psi}$. If the key-value pair are highly related to the triplet (i.e., $\vec{\psi}$ has high values), the merge operation (via the "min") ensures that the values in the overall relatedness feature vector are determined mostly by the values of $\vec{\phi}$ (i.e., relatedness for the triplet). In other words, the final predicted score depends mostly on the triplet. In contrast, if the key-value pair is poorly related to the triplet (i.e., $\vec{\psi}$ has low values), the merge ("min") operation ensures that the values in the overall relatedness feature vector are low (i.e., they are determined mainly based on the values of $\vec{\psi}$). In other words, the overall relatedness of the hyper-relational fact is low, because the key-value pair is poorly related to the triplet, which further implies that this fact is less probably a true fact.

Third, goal III is automatically fulfilled with our merge operation. For the case of multiple key-value pairs (k_i, v_i), $i = 1, ..., n$, associated with the triplet, *each* key-value pair (k_i, v_i) is combined with the triplet to generate the corresponding quintuple-wise relatedness feature vector $\vec{\psi}_i$. Subsequently, our merging (via the "min") operation is able to take an arbitrary number of quintuple-wise relatedness feature vectors as input.

4.3.4 Model Training Process

To train the model parameters, we minimize a softplus loss. More precisely, following [124, 47], our loss function is defined as the negative log-likelihood of the logistic model:

$$\sum_{\omega \in \Omega} log(1 + e^{-\sigma(\omega)}) + log(1 + e^{\sigma(\omega')}) \tag{4.1}$$

where Ω is the input set of hyper-relational facts. For each hyper-relational fact ω containing (h, r, t) and the associated (k_i, v_i), $i = 1, ..., n$, one negative sample ω' is generated by randomly corrupting one entity (h, t, or v_i) or relation (r or k_i). $\sigma(\omega)$ and $\sigma(\omega')$ denote the predicted score of our HINGE model for the true fact ω and the negative fact ω', respectively.

The loss function 5.2 is minimized using the Adam stochastic optimizer [64], and the model parameters are learnt via back propagation. Specifically, we use rectified linear units (ReLU) as the non-linearity activation function [68] and batch normalization [57] after the two CNN layers for fast training.

4.4 Experiments

In this section, we evaluate our proposed model HINGE on various link prediction tasks. In the following, we start by presenting our experimental setup, followed by our results and discussions.

Table 4.1 – Statistics of the datasets

Dataset		**JF17K**		**WikiPeople**	
#Entity		28,645		34,839	
#Relation		322		375	
#Fact (training)	Only triple	44,210	57.8%	280,520	97.4%
	Only hyper-relational	32,169	42.2%	7,389	2.6%
	Total	76,379	100%	287,918	100%
#Fact (test)	Only triple	10,417	42.4%	36,597	97.4%
	Only hyper-relational	14,151	57.6%	971	2.6%
	Total	24,568	100%	37,586	100%

4.4.1 Experimental Setup

Dataset

We conduct experiments on two hyper-relational datasets ***JF17K*** [133] and ***WikiPeople*** [47], extracted from two popular KGs, i.e., Freebase and Wikidata, respectively. Each of these two datasets contains both triple facts and hyper-relational facts. While *JF17K* was filtered from Freebase to have a significant presence of hyper-relational facts (see [133] for more detail), *WikiPeople* is extracted from Wikidata and focuses on entities of type *human* without any specific filtering to improve the presence of hyper-relational facts [47]. As the original WikiPeople dataset also contains literals (used as tails) in some facts, we filter out these non-entity literals and the corresponding facts. Table 5.1 shows the main statistics of these datasets.

Baselines

We compare HINGE against a sizable collection of state-of-the-art Knowledge Graph embedding techniques from two categories.

The first category includes models learning from triplets only.

- Translational distance models: **TransE** [19] learns to preserve the relation between two entities as $h + r \approx t$. **TransH** [132] extends TransE to better capture multi-mapping relations by introducing relation-specific hyperplanes. **TransR** [74] introduces relation-specific projections to also better capture multi-mapping relations. **TransD** [59] extends TransR by decomposing the projection matrix into a product of two vectors. These models minimize a margin-based ranking objective function, where we empirically set the margin $b = 1$ with the L_2-norm. In addition, we set the learning rate to 0.001 with a stochastic gradient descent optimizer, the number of negative samples to 1, and the batch size to 128.
- Semantic matching models: **Rescal** [95] represents each entity as a vector and each relation

as a matrix, and uses a bilinear function to model the relation between a pair of entities. **DistMult** [143] simplifies Rescal by representing each relation embedding as a diagonal matrix. **ComplEx** [124] further extends DistMult in the complex space in order to better model both symmetric and asymmetric relations. **Analogy** [76] models explicitly analogical structures in multi-relational KG embeddings. **ConvE** [29] adopts a 2D CNN to capture richer interactions between entity and relation embeddings. We set the margin $b = 1$ with the L_2-norm for Rescal. For DistMult, ComplEx and Analogy, we set the learning rate to 0.1 with Adagrad optimizer [33], the number of negative samples to 1, and the batch size to 128. For ConvE, we set the learning rate to 0.003, the batch size to 128, the dropout to 0.2, and the label smoothing value to 0.1.

The second category includes models learning from the n-ary representation of hyper-relational facts.

- **m-TransH** [133] models the interaction between entities involved in each n-ary fact. Specifically, each hyper-relational fact (h, r, t) with (k_i, v_i), $i = 1, ..., n$ is associated with a so-called meta-relation, represented as an ordered list of keys (relations), such as $R := (r_h, r_t, k_1, ..., k_n)$; the fact is then represented as a list of ordered values associated with the meta-relation $\{R, (h, t, v_1, ..., v_n)\}$. Using this representation, m-TransH is built on top of TransH to capture multi-fold relations between entities in a meta-relation. As it learns only from sets of entities in meta-relations (without considering the exact relations in each meta-relation), it can be applied to perform the link prediction task on missing entities only. Following suggestions from the original paper, we empirically set its hyper-parameters as follows: the margin to 0.5, the weight to 0.001, and the threshold to 0.01.
- **RAE** [152] extends m-TransH by explicitly considering the pairwise relatedness between entities in n-ary facts. Using the same n-ary representation of hyper-relational facts, RAE further learns from the pairwise relatedness between entities in each n-ary fact in order to perform instance reconstruction, i.e., predicting one or multiple missing entities. Similar to m-TransH, RAE can only be used to predict missing entities. We search for optimal parameters by adopting the techniques described in [47] on each dataset, and report the results with the optimal settings.
- **NaLP** [47] models the relatedness between key-value (relation-entity) pairs contained in each n-ary fact. It represents each hyper-relational fact (h, r, t) with (k_i, v_i), $i = 1, ..., n$, as a set of key-value pairs $\{r_h{:}h,\ r_t{:}t,\ k_i{:}v_i\}$, $i = 1, ..., n$ by converting the relation r into two keys r_h and r_t, associating with head h and tail t, respectively. Using this representation, NaLP learns from the pairwise relatedness between key-value pairs via a neural network pipeline, which enables the prediction of both missing keys (relations) or missing values (entities). Note that in NaLP, a commonly adopted negative sampling process is used, which randomly

corrupts one key or value in a true fact. However, this process is not fully adaptable to its n-ary represention of hyper-relational facts, in particular for keys r_h and r_t. For example, when corrupting the key r_h by a randomly sampled r'_h ($r \neq r'$), the negative fact becomes $\{r'_h{:}h,\ r_t{:}t,\ k_i{:}v_i\}$, $i = 1, ..., n$. This is unrealistic as r'_h is not compatible with r_t while only *one* relation r (or r') can be assumed between h and t in a hyper-relational fact. Therefore, we propose a variant of NaLP fixing this issue. Following the suggestion from the original paper, we adopt the optimal hyper-parameters reported on each dataset.

- **NaLP-Fix** is a variant of NaLP with a fixed negative sampling process. Specifically, when corrupting the key r_h by a randomly sampled r'_h ($r \neq r'$), we also corrupt r_t by r'_t, resulting in a negative fact $\{r'_h{:}h,\ r'_t{:}t,\ k_i{:}v_i\}$, $i = 1, ..., n$. Subsequently for this negative fact, only a single relation r' links h and t, which is a realistic case. Similarly, when corrupting r_t, we also corrupt r_h in the same way. We keep using the same hyper-parameters as for NaLP.

For our HINGE model, we empirically set the number of filters n_f in both CNNs to 400, the batch size to 128, and the learning rate to 0.0001. The embedding size is set to 100 for all methods, if not specified otherwise. The implementation of HINGE and used datasets are available here[2].

Dataset Configuration

As discussed in the introduction, hyper-relational facts need to be transformed into triplets for the models that can learn from triplets only. There are three common settings for such a transformation.

- *Basic*: Only the triplet (h, r, t) from a hyper-relational fact is kept, while removing all its associated key-value pairs. This setting causes irreversible information loss.
- *Relation Path* [133]: For each hyper-relational fact containing a triplet (h, r, t) and associated key-value pairs (k_i, v_i), $i = 1, ..., n$, each pair of entities are linked via an artificially created relation path. For example, h is linked to v_i via a virtual relation $\widehat{rk_i}$ representing a relation path $r \to k_i$, resulting in an additional triplet $(h, \widehat{rk_i}, v_i)$. This setting creates extra relations and facts.
- *Reification* [20]: For each hyper-relational fact containing key-value pairs (k_i, v_i), $i = 1, ..., n$, an artificial entity is used to represent the triplet $q := (h, r, t)$, and then additional triplets are created as (q, k_i, v_i), $i = 1, .., n$. This setting also creates extra entities and triplets.

In addition, we denote the original hyper-relational facts as the *Original* setting.

[2] https://github.com/eXascaleInfolab/HINGE_code/

Evaluation Tasks and Metrics

Link prediction is a typical task for Knowledge Graph completion. Given two elements of a triplet in a (hyper-relational) fact, the task is to predict the missing one, such as $(?, r, t)$, $(h, ?, t)$ or $(h, r, ?)$, where the question mark represents the missing entity/relation. In this chapter, we conduct experiments in all of these three cases, i.e., predicting a missing head, relation, or tail. We describe our evaluation protocols below by taking the case of predicting missing heads $(?, r, t)$ as an example. For the triplet $(?, r, t)$ in one test (hyper-relational) fact, we replace the missing head with all the entities, resulting in a set of candidate (hyper-relational) facts. Among those candidate facts, in addition to the testing fact itself, other corrupted facts might also be true facts (i.e., existing in the training/test datasets); these facts are thus removed from the candidate facts. Afterward, the resulting candidate facts are fed into an embedding model to output predicted scores. By ranking the candidate facts according to their corresponding scores, we generate a predicted ranking list of entities for the missing head. By repeating the evaluation process over all test facts in the test dataset, we report Mean Reciprocal Rank (MRR), $Hits@10$ and $Hits@1$, which are widely used metrics for link prediction tasks [47]. The same evaluation protocol and metrics also apply to predicting missing relations $(h, ?, t)$ and tails $(h, r, ?)$. As predicting missing heads or tails is essentially predicting missing entities, we report average results on these two cases (denoted as "Head/Tail Prediction"), while we report individual results for relation prediction.

4.4.2 Limitation of N-Ary Representation

In this experiment, we experimentally show the limitation of the n-ary representation of a hyper-relational fact (represented as a set of key-value pairs without triplets). Specifically, most of the existing works on learning KG embeddings from hyper-relational facts [133, 152, 47] transform a hyper-relational fact into a set of key-value (relation-entity) pairs while completely avoiding triplets. For example, NaLP [47] transforms a hyper-relational fact (h, r, t) with (k_i, v_i), $i = 1, ..., n$, into a set of key-value pairs $\{r_h{:}h,\ r_t{:}t,\ k_i{:}v_i\}$, $i = 1, ..., n$ by converting relation r into two keys r_h and r_t, associated with head h and tail t, respectively. Subsequently in the embedding learning process, these key-value pairs are treated equally. However, we argue that as triplets still serve as the fundamental data structure in the modern KGs, they preserve the essential information for link prediction in KGs. In other words, key-value pairs (k, v) on a hyper-relational fact should not be treated equally as triplets (h, r, t).

To verify this point, we define an extra setting, the *null model*, for dataset transformation. The null model reconstructs one triplet from each n-ary relational fact $\{r_h{:}h,\ r_t{:}t,\ k_i{:}v_i\}$, $i = 1, ..., n$, via a randomly sampled relation path [133], such as $(h, \widehat{r_h k_i}, v_i)$. Note that if we only sample the relation path $\widehat{r_h r_t}$, we reconstruct the original triplet $(h, \widehat{r_h r_t}, t)$ (corresponding to the *Basic* setting). With this null model, we make a null hypothesis: *The information for link prediction*

Table 4.2 – Link prediction performance on WikiPeople.

Dataset Transformation Setting	**Method**	**WikiPeople**					
		Head/Tail Prediction			Relation Prediction		
		MRR	Hit@10	Hit@1	MRR	Hit@10	Hit@1
Null Model	TransE	0.2021	0.4377	0.1031	0.2060	0.2626	0.1683
	TransH	0.1998	0.4400	0.0992	0.2203	0.2909	0.1785
	TransR	0.2009	0.4355	0.1015	0.1485	0.1972	0.1196
	TransD	0.1083	0.3416	0.0074	0.2985	0.4816	0.2089
	Rescal	0.1325	0.2986	0.0626	**0.6561**	**0.8174**	**0.5474**
	DistMult	0.1144	0.3083	0.0385	0.4281	0.5520	0.3315
	ComplEx	0.1050	0.2932	0.0329	0.3719	0.4467	0.3093
	Analogy	0.1144	0.2991	0.0406	0.4178	0.4903	0.3535
	ConvE	**0.2383**	**0.4548**	**0.1470**		N/A	
Basic	TransE	0.3242	0.6064	0.1216	0.3482	0.4200	0.2734
	TransH	0.3206	0.6029	0.1155	0.3724	0.4448	0.2980
	TransR	0.3264	0.6090	0.1236	0.2446	0.4996	0.1651
	TransD	0.2200	0.5414	0.0205	0.5657	0.8804	0.4160
	Rescal	0.2772	0.4915	0.1404	**0.7936**	**0.9023**	**0.7306**
	DistMult	0.2468	0.5087	0.0645	0.6008	0.6776	0.5479
	ComplEx	0.2466	0.4944	0.0648	0.5676	0.6135	0.5367
	Analogy	0.2521	0.5033	0.0688	0.5984	0.6386	0.5699
	ConvE	**0.4781**	**0.6533**	**0.3666**		N/A	
Relation Path	TransE	0.3191	0.6067	0.1160	0.2773	0.3379	0.2444
	TransH	0.3198	0.6084	0.1155	0.2399	0.3267	0.1906
	TransR	0.3214	0.6086	0.1167	0.1593	0.2154	0.1272
	TransD	0.2083	0.5228	0.0146	0.3344	0.5053	0.2443
	Rescal	0.2637	0.4780	0.1255	**0.7535**	**0.8673**	**0.6895**
	DistMult	0.2400	0.4987	0.0588	0.5787	0.6429	0.5360
	ComplEx	0.2415	0.4861	0.0672	0.4975	0.5415	0.4716
	Analogy	0.2443	0.4936	0.0688	0.5139	0.5555	0.4887
	ConvE	**0.4700**	**0.6537**	**0.3527**		N/A	
Reification	TransE	0.3207	0.5977	0.1224	0.3253	0.3850	0.2747
	TransH	0.3244	0.6011	0.1242	0.3160	0.3873	0.2694
	TransR	0.3225	0.6002	0.1225	0.2396	0.2968	0.1817
	TransD	0.2123	0.5253	0.0195	0.5293	0.8611	0.3821
	Rescal	0.2751	0.4815	0.1430	**0.7684**	**0.8816**	**0.7053**
	DistMult	0.2276	0.4867	0.0519	0.5902	0.6611	0.5422
	ComplEx	0.2365	0.4795	0.0614	0.5375	0.5882	0.5039
	Analogy	0.2478	0.4901	0.0718	0.5838	0.6277	0.5562
	ConvE	**0.4657**	**0.6434**	**0.3559**		N/A	
Original	m-TransH	0.0633	0.3006	0.0633		N/A	
	RAE	0.0586	0.3064	0.0586		N/A	
	NALP	0.4084	0.5461	0.3311	0.4818	0.8516	0.3198
	NALP-Fix	0.4202	0.5564	0.3429	0.8200	0.9757	0.7197
	HINGE	**0.4763**	**0.5846**	**0.4154**	**0.9500**	**0.9977**	**0.9159**

Table 4.3 – Link prediction performance on JF17K.

Dataset Transformation Setting	**Method**	**JF17K**					
		Head/Tail Prediction			Relation Prediction		
		MRR	Hit@10	Hit@1	MRR	Hit@10	Hit@1
Null Model	TransE	0.1916	0.3553	0.1079	0.6240	0.7356	0.5522
	TransH	0.1916	0.3529	0.1093	**0.6382**	**0.7543**	**0.5623**
	TransR	0.1921	0.3603	0.1085	0.6201	0.7207	0.5543
	TransD	0.0626	0.1890	0.0028	0.2155	0.2894	0.1718
	Rescal	0.1095	0.2031	0.0630	0.5679	0.6781	0.4954
	DistMult	0.0997	0.2145	0.0475	0.0512	0.2255	0.0323
	ComplEx	0.0828	0.1824	0.0382	0.1283	0.3277	0.0474
	Analogy	0.0917	0.1952	0.0438	0.1167	0.2495	0.0581
	ConvE	**0.2270**	**0.4174**	**0.1357**		N/A	
Basic	TransE	0.2556	0.4529	0.1576	0.8574	0.9064	0.8270
	TransH	0.2570	0.4564	0.1619	**0.8618**	**0.9134**	**0.8328**
	TransR	0.2806	0.4974	0.1791	0.8431	0.8924	0.8137
	TransD	0.1343	0.3105	0.0501	0.6803	0.7872	0.6189
	Rescal	0.1709	0.3340	0.0952	0.7887	0.8491	0.7480
	DistMult	0.1752	0.3531	0.0955	0.2779	0.5340	0.1381
	ComplEx	0.1669	0.3307	0.0906	0.2380	0.3445	0.1765
	Analogy	0.1776	0.3471	0.0996	0.2667	0.4247	0.1773
	ConvE	**0.3190**	**0.5470**	**0.2129**		N/A	
Relation Path	TransE	0.2832	0.4826	0.1832	0.8251	0.8940	0.7814
	TransH	0.2863	0.4899	0.1850	0.8179	0.8897	0.7738
	TransR	0.3075	0.5170	0.2051	0.6866	0.7779	0.6383
	TransD	0.1279	0.3204	0.0362	0.4333	0.5561	0.3676
	Rescal	0.1692	0.3188	0.0966	**0.8336**	**0.8957**	**0.7914**
	DistMult	0.2261	0.4084	0.1368	0.2817	0.5551	0.1574
	ComplEx	0.2193	0.3930	0.1352	0.2126	0.3361	0.1477
	Analogy	0.2244	0.3986	0.1413	0.2523	0.4308	0.1625
	ConvE	**0.3665**	**0.5876**	**0.2574**		N/A	
Reification	TransE	0.2285	0.3806	0.1503	**0.8793**	**0.9187**	**0.8559**
	TransH	0.2302	0.3815	0.1538	0.8774	0.9218	0.8506
	TransR	0.2838	0.4722	0.1914	0.8751	0.9157	0.8510
	TransD	0.0950	0.2121	0.0366	0.5562	0.6610	0.5010
	Rescal	0.1354	0.2608	0.0759	0.6958	0.7620	0.6556
	DistMult	0.1523	0.2888	0.0875	0.1135	0.3251	0.0332
	ComplEx	0.1325	0.2552	0.0760	0.1311	0.2451	0.0717
	Analogy	0.1329	0.2594	0.0742	0.1548	0.2983	0.0852
	ConvE	**0.3469**	**0.5410**	**0.2500**		N/A	
Original	m-TransH	0.2060	0.4627	0.2060		N/A	
	RAE	0.2153	0.4668	0.2153		N/A	
	NALP	0.2209	0.3310	0.1650	0.6391	0.8215	0.5472
	NALP-Fix	0.2446	0.3585	0.1852	0.7469	0.8921	0.6665
	HINGE	**0.4489**	**0.6236**	**0.3611**	**0.9367**	**0.9894**	**0.9014**

encoded by the original triplets is not greater than the triplets created by the null model. We test this null hypothesis by performing link prediction tasks using all nine baseline models learning from triplets on the two sets of triplets, i.e., on the original *basic* triplets and the triplets created by the *null model*.

Table 4.2 and Table 4.3 show the results on both datasets. Comparing the results from the two data transformation settings *basic* and *null model*, we clearly observe that the performance from the original triplets is consistently and significantly better than the performance from the triplets created by the null model, with an average improvement[3] of 77.5% in head/tail prediction and 58.3% in relation prediction on the WikiPeople dataset (125.2% and 114.9% on JFK17K, respectively). To further verify this point, we conduct one-tailed paired t-test on the results using the original triplets and the results using the triplets created by the null model from the same set of nine baselines, for each metric and each link prediction task. The test results consistently reject the null hypothesis at the 0.01 significance level (p-value $\ll 0.01$). Therefore, we verify that the information encoded by the original triplets is significantly greater than the triplets created by the null model for link prediction. In other words, compared to the key-value pairs, the triplets in hyper-relational facts preserve the essential information for link prediction in KGs; this finding indeed corresponds to the fundamental design principle behind our new technique HINGE.

4.4.3 Link Prediction Performance Comparison

In this experiment, we compare the link prediction performance of HINGE against all baselines under different dataset transformation settings. Table 4.2 and Table 4.3 show the results on both datasets. For each dataset transformation setting, we highlight the best-performing method on each task and for each dataset. In the following, we discuss the results and highlight our key findings.

Comparison with Baselines Learning from Triplets Only

We observe that our HINGE model consistently outperforms all baselines learning from triplets only, for all three dataset transformation settings. Specifically, the *Basic* setting simply discards the key-value pairs, causing irreversible information loss, while the *Relation Path* and *Reification* settings create extra entities/relations which indeed distract the embedding models from capturing the essential information for link prediction from the input KG. We compute the average improvement of HINGE over the best-performing baselines for each dataset transformation settings in Table 4.4. We observe that HINGE yields significant improvement in most cases, showing improvements of up to 41.45% on head/tail prediction and of up to

[3]referring to the average improvement on different metrics throughout this chapter.

Table 4.4 – Improvement of HINGE over the best-performing baselines learning from triplets only. The best performing baselines are highlight in Table 4.2. The *Null Model* setting is excluded due to its very low performance.

Dataset	**WikiPeople**		**JF17K**	
Transformation Setting	Head/Tail Prediction	Relation Prediction	Head/Tail Prediction	Relation Prediction
Basic	0.81%	18.55%	41.45%	8.41%
Relation Path	2.85%	24.65%	22.96%	12.24%
Reification	3.28%	22.22%	29.71%	6.52%

24.65% on relation prediction.

One exception is for the head/tail prediction on WikiPeople with the *Basic* setting, where the improvement is marginal (0.81%). We further find that the best-performing baseline in this case is ConvE, which is indeed the most competitive baseline in head/tail prediction in all cases (see Table 4.2 and Table 4.3). Note that similar to our HINGE, ConvE also uses a 2D CNN layer for feature extraction from entity/relation embeddings in triplets, yielding good performance on head/tail prediction. The marginal improvement can be explained by the dominance of triple facts in WikiPeople dataset (97.4% triple facts vs 2.6% hyper-relational facts in both training and test datasets), where both HINGE and ConvE perform well. In contrast, on the JF17K dataset, which contains a significant portion of hyper-relational facts (57.8% triple facts vs 42.2% hyper-relational facts in the training dataset and 42.4% triple facts vs 57.6% hyper-relational facts in the test dataset), HINGE significantly outperforms ConvE by leveraging key-value pairs in the hyper-relational facts (while ConvE learns from a transformed dataset using one of our data transformation settings). In addition, we also highlight that ConvE is specifically designed for head/tail prediction only, and is not applicable to the relation prediction task.

Another interesting observation is that, compared to the *Basic* setting, the *Relation Path* and *Reification* settings yield worse results in general. For example, on WikiPeople, compared to the *Basic* setting, the *Relation Path* setting shows an average performance drop of 3.4% on head/tail prediction and 20.5% on relation prediction (the *Reification* setting shows 2.2% and 5.7% performance drop, respectively). This further verifies that even though the *Relation Path* and *Reification* settings preserve the complete information of a hyper-relational fact, the extra created entities/relations indeed distract the KG embeddings from capturing essential information for link prediction in the input KG.

Table 4.5 – Link prediction performance on triple and hyper-relational facts on WikiPeople.

Fact Type	**Method**	**WikiPeople**					
		Head/Tail Prediction			Relation Prediction		
		MRR	Hit@10	Hit@1	MRR	Hit@10	Hit@1
Triple Fact	NaLP-Fix	0.4216	0.5592	0.3436	0.8057	0.9728	0.7022
	HINGE	**0.4765**	**0.5874**	**0.3937**	**0.9493**	**0.9979**	**0.9145**
Hyper-relational Fact	NaLP-Fix	0.3050	0.4757	0.2154	0.7605	0.9476	0.6517
	HINGE	**0.3213**	**0.4888**	**0.2322**	**0.9432**	**1.0000**	**0.8876**

Comparison with Baselines Learning from Hyper-Relational Facts

We observe that our proposed model HINGE consistently outperforms all other baselines learning from hyper-relational facts. Among the methods learning from the n-ary representation of hyper-relational facts (i.e., m-TransH, RAE and NaLP), NaLP shows the best performance, as it learns the relatedness between key-value (relation-entity) pairs while m-TransH and RAE learn from entities only. Note that m-TransH and RAE result in very low performance on WikiPeople, which is probably due to the weak presence of hyper-relational facts in WikiPeople while and m-TransH and RAE are designed for hyper-relational facts. Moreover, compared to NaLP, NaLP-Fix (with our fixed negative sampling process) consistently shows better performance, with a slight improvement of 2.8% in head/tail prediction, and a tremendous improvement of 69.9% in relation prediction on WikiPeople (10.4% and 15.8% on JFK17K, respectively). This verifies the effectiveness of our fixed negative sampling process, in particular for relation prediction. Finally, compared to the best-performing baseline NaLP-Fix, HINGE shows a 13.2% improvement on the head/tail prediction task, and a 15.1% improvement on the relation prediction task on WikiPeople (84.1% and 23.8% on JF17K, respectively).

In addition, we also find that the baseline methods learning from hyper-relational facts (i.e., m-TransH, RAE, NaLP and our NaLP-Fix) yield, surprisingly, worse performance in many cases than the best-performing baselines learning from triplets only. This can be explained by the fact that their n-ary representation of hyper-relational facts indeed ignores the triplet structure, by converting a hyper-relational fact (h, r, t) with (k_i, v_i), $i = 1, ..., n$, into a set of key-value pairs $\{r_h{:}h,\ r_t{:}t,\ k_i{:}v_i\}$, $i = 1, ..., n$. However, as the triplet structure serves as the fundamental data structure in KGs and thus preserves essential information for link prediction in the KGs, even though these methods can learn from key-value pairs associated with triplets in hyper-relational facts, their ignorance of the triplet structure results in subpar performance.

4.4.4 Link Prediction Performance on Triple and Hyper-Relational Facts

In this experiment, we look into the breakdown of link prediction performance on triple and hyper-relational facts. We compare HINGE with NaLP-Fix only, as it is the best-performing

Table 4.6 – Link prediction performance on triple and hyper-relational facts on JF17K.

Fact Type	**Method**	**JF17K**					
		Head/Tail Prediction			Relation Prediction		
		MRR	Hit@10	Hit@1	MRR	Hit@10	Hit@1
Triple Fact	NaLP-Fix	0.0901	0.1802	0.0464	0.6005	0.8253	0.4806
	HINGE	**0.2641**	**0.4965**	**0.1572**	**0.8723**	**0.9846**	**0.7965**
Hyper-relational Fact	NaLP-Fix	0.3420	0.4760	0.2693	0.8542	0.9381	0.8067
	HINGE	**0.5850**	**0.7172**	**0.5112**	**0.9841**	**0.9929**	**0.9785**

Table 4.7 – Key/Value prediction performance on hyper-relational facts on WikiPeople.

Method	**WikiPeople**					
	Value Prediction			Key Prediction		
	MRR	Hit@10	Hit@1	MRR	Hit@10	Hit@1
NaLP-Fix	0.4251	0.5789	0.3426	0.7643	0.8970	0.6945
HINGE	**0.5506**	**0.6880**	**0.4714**	**0.9986**	**0.9996**	**0.9978**

Table 4.8 – Key/Value prediction performance on hyper-relational facts on JF17K.

Method	**JF17K**					
	Value Prediction			Key Prediction		
	MRR	Hit@10	Hit@1	MRR	Hit@10	Hit@1
NaLP-Fix	0.4251	0.5789	0.3426	0.7643	0.8970	0.6945
HINGE	**0.5506**	**0.6880**	**0.4714**	**0.9986**	**0.9996**	**0.9978**

baseline learning from hyper-relational facts.

Table 4.5 and Table 4.6show the results. We observe that HINGE consistently achieves the best performance on both types of facts. In addition, we also find that while the performance of both methods on triple facts is higher than on hyper-relational facts on WikiPeople, we have a completely opposite observation on JF17K, i.e., the performance on triple facts is obviously lower than on hyper-relational facts. This can be explained by the dataset statistics. Where the JF17K dataset has a significant presence of hyper-relational facts (42.2% and 73.6% in the training and test datasets, respectively), WikiPeople contains much fewer hyper-relational facts (2.6% in both the training and test datasets).

4.4.5 Key/Value Prediction Performance

In this experiment, we study the performance of HINGE on a key/value prediction task. Specifically, as a hyper-relational fact may contains a set of key-value pairs (k_i, v_i), with $i = 1, ..., n$, associated with the triplet (h, r, t), this task tries to predict a missing key $(?, v_i)$ or a missing value $(k_i, ?)$ in a hyper-relational fact. Our evaluation protocol is similar to the one from the link prediction task. Taking the case of predicting a missing key $(?, v_i)$ as an example, we first replace the missing key with all possible relations, resulting in a set of candidate hyper-relational facts. After filtering out the other true facts (existing in the training/test datasets) except the test fact itself, we feed the remaining candidate hyper-relational facts to an embedding model to output predicted scores. By ranking the candidate facts according to their scores, we report MRR, $Hits@10$ and $Hits@1$. The same evaluation protocol and metrics also apply to predicting a missing value $(k_i, ?)$. Similar to the previous experiment, we compare HINGE only with NaLP-Fix, which is the best-performing baseline learning from hyper-relational facts.

Table 4.7 and Table 4.8 shows the results. We observe that HINGE consistently outperforms NaLP-Fix, showing a 6.0% improvement on the value prediction task, and a 29.4% improvement on the key prediction task on WikiPeople (28.7% and 28.6% on JF17K, respectively).

4.4.6 Parameter Sensitivity Study

Finally, we study the impact of two key parameters in HINGE, i.e., the number of filters n_f used in the CNNs, and the embedding dimension K. First, by fixing the embedding dimension $K = 100$, we vary the number of filters n_f from 10 to 800 on a log scale, and show its impact on the link prediction tasks for both datasets in Figure 4.4. We observe that when increasing n_f, the performance starts to increase, and then flatten out when $n_f \geq 400$ in most cases. Second, by fixing the number of filters $n_f = 400$, we vary the embedding dimension K from 5 to 200 on a log scale, and show its impact on the link prediction tasks for both datasets in Figure

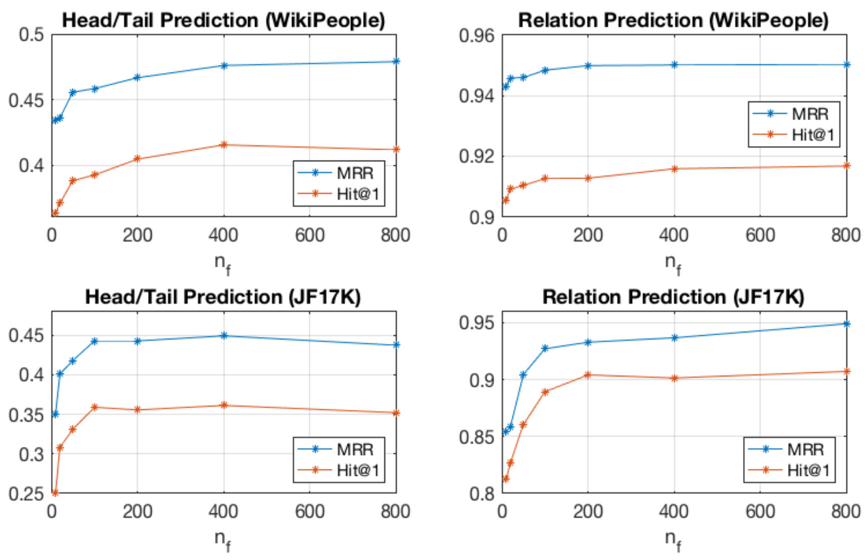


Figure 4.4 – Impact of the number of filters n_f

4.5. Similar to the case of n_f, we observe that when increasing K, the performance starts to increase, and then flatten out when $K \geq 100$ in most cases. Therefore, we set the number of filters $n_f = 400$ and the embedding dimension $K = 100$, in all previous experiments.

4.5 Conclusion

Existing Knowledge Graph embedding techniques mostly represent a KG as a set of triplets, and then learn entity/relation embeddings from such triplets while preserving the essential information for link prediction in the KG. However, this triplet representation oversimplifies the complex nature of the data stored in the KG, in particular for hyper-relational facts, where each fact contains not only a triplet (h, r, t), but also its associated key-value pairs (k, v). Even though a few recent techniques learn from such data using an n-ary representation (i.e., a set of key-value pairs only without any triplet), they result in suboptimal models due to their ignorance of the triplet structure, which, as we show in this chapter, is the fundamental structure in modern KGs and encodes essential information for link prediction. Against this background, we proposed HINGE, a hyper-relational KG embedding model. It captures not only the primary structural information of the KG from the triplets, but also the correlation between each triplet and its associated key-value pairs. Our extensive evaluation shows the superiority of HINGE on various link prediction tasks over KGs using two real-world KG datasets. In particular, HINGE consistently outperforms not only the KG embedding methods learning from triplets only (by 0.81-41.45% depending on the link prediction tasks and settings), but also the methods learning from hyper-relational facts using n-ary representations (by

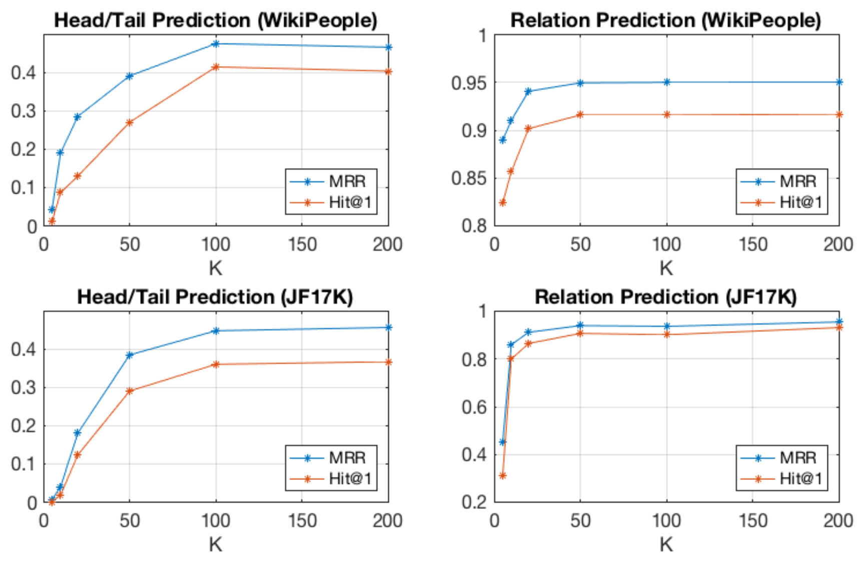


Figure 4.5 – Impact of the embedding dimension K

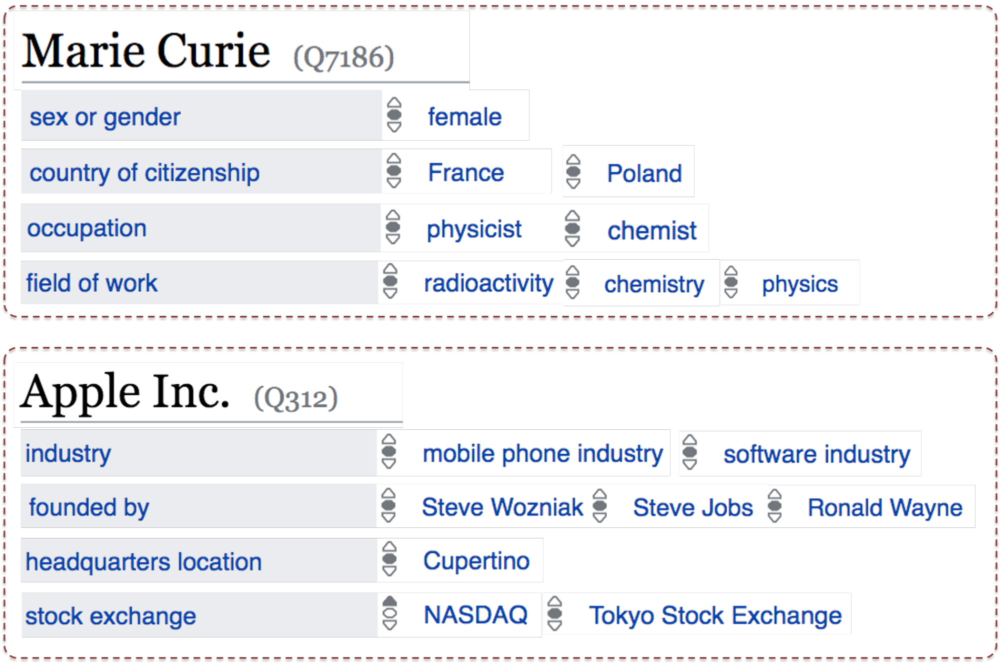
13.2-84.1%).

In the future, we plan to investigate the hyper-relational KG embedding problem further by taking into consideration other types of data in a KG, such as numerical and text literals, or multi-modal data such as images.

In the next chapter we propose an end-to-end solution for instance completion in Knowledge Graphs called RETA. Unlike link prediction, where the goal is to predict an entity given an entity-relation pair (or predict a relation given two entities), instance completion suggests relation-entity pairs for a given entity. Specifically, RETA consists of two components: a RETA-Filter that generates candidate r-t pairs for a given h, and RETA-Grader that evaluates and ranks the candidate r-t pairs using a newly-designed KG embedding model based on Convolutional Neural Networks. We evaluate our proposed method against a sizable collection of state-of-the-art techniques on three real-world KG datasets showing not only the candidates quality of RETA-Filter, but also the performance of RETA-Grader.

5 RETA: A Schema-Aware, End-to-End Solution for Instance Completion in Knowledge Graphs

Knowledge Graph completion has been widely studied to tackle the incompleteness issue (i.e., missing facts) in modern KGs. A fact in a KG is represented as a triplet (h, r, t) linking two entities h and t via a relation r. Existing work mostly consider *link prediction* to solve this problem, i.e., given two elements of a triplet predicting the missing one, such as $(h, r, ?)$. This task has, however, a strong assumption on the two given elements in a triplet, which have to be correlated, resulting otherwise in meaningless predictions, such as (*Marie Curie*, *headquarters location*, ?). In addition, the KG completion problem has also been formulated as a *relation prediction* task, i.e., when predicting relations r for a given entity h. Without predicting t, this task is however a step away from the ultimate goal of KG completion. Against this background, this chapter studies an instance completion task suggesting r-t pairs for a given h, i.e., $(h, ?, ?)$. We propose an end-to-end solution called RETA (as it suggests the <u>Re</u>lation and <u>Ta</u>il for a given head entity) consisting of two components: a RETA-Filter and RETA-Grader. More precisely, our RETA-Filter first generates candidate r-t pairs for a given h by extracting and leveraging the schema of a KG; our RETA-Grader then evaluates and ranks the candidate r-t pairs considering the plausibility of both the candidate triplet and its corresponding schema using a newly-designed KG embedding model. We evaluate our methods against a sizable collection of state-of-the-art techniques on three real-world KG datasets. Results show that our RETA-Filter generates of high-quality candidate r-t pairs, outperforming the best baseline techniques while reducing by 10.61%-84.75% the candidate size under the same candidate quality guarantees. Moreover, our RETA-Grader also significantly outperforms state-of-the-art link prediction techniques on the instance completion task by 16.25%-65.92% across different datasets.



Marie Curie (Q7186)

sex or gender	female		
country of citizenship	France	Poland	
occupation	physicist	chemist	
field of work	radioactivity	chemistry	physics

Apple Inc. (Q312)

industry	mobile phone industry	software industry	
founded by	Steve Wozniak	Steve Jobs	Ronald Wayne
headquarters location	Cupertino		
stock exchange	NASDAQ	Tokyo Stock Exchange	

Figure 5.1 – Examples of entities *Marie Curie* and *Apple Inc.* from Wikidata. For each entity, we present a subset of triplets having that entity as head from Wikidata.

5.1 Introduction

Knowledge Graphs, such as Freebase [16], Wikidata[1] or Google's Knowledge Graph[2], have become a key resource powering a broad spectrum of Web applications, such as semantic search [138], question-answering [146], or recommender systems [151]. A typical KG represents a large collection of entities (real-world objects or abstract concepts) interconnected via relations, where entities contribute to the description of other entities via relations. Using a triplet representation scheme, a KG consists of a set of triplets, (*head, relation, tail*), or (h, r, t) for short, each of which encoding a relation connecting a head entity to a tail entity, such as *Marie Curie* (head) *sex or gender* (relation) *female* (tail). Figure 5.1 shows two real-world examples of entities *Marie Curie* and *Apple Inc.* from Wikidata; each example is associated with a set of triplets having the entity as head in Wikidata. Despite the fact that modern KGs contain high-quality structured data, they are also known to suffer from an incompleteness issue, i.e., missing facts. For example, 71% of all people from Freebase have no *place of birth* [93].

In this context, Knowledge Graph completion problems have been widely studied. In the current literature, these problems are mostly formulated as a link prediction task [93, 129], i.e.,

[1]http://wikidata.org/

[2]https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html

to predict missing links in a KG. More precisely, given two elements of a triplet, the task is to predict the missing one, such as $(h, r, ?)$, $(h, ?, t)$ or $(?, r, t)$, where the question mark represents the missing entity/relation. However, despite the wide adoption of such link prediction tasks by existing work for KG completion, this task is often impractical due to its strong assumption on knowing two elements in a triplet. In fact, the two known elements in a triplet must somehow be correlated in order to make a meaningful prediction, otherwise the task will result in meaningless results. For example, the predictions for (*Marie Curie*, *headquarters location*, ?) or (*Apple Inc.*, *sex or gender*, ?) do not make any sense, while they would both be considered as valid input for the link prediction task. Although existing work on link prediction implement this task by taking out one element from a true triplet in a KG and then making predictions on the triplet (which implicitly ensures the correlation between the two remaining elements in the triplet), such an experimental setting indeed departs from many real-world use cases of KG completion where we are not given two correlated elements in a triplet. Another thread of work on KG completion focuses on the relation prediction task[3], suggesting missing relations to a given entity by collaboratively using the information about other entities that are similar to the given entity, for example [9]. However, these techniques only suggest relations for a given head, without predicting the tails and hence are only part of the solution.

In this chapter and toward the ultimate goal of KG completion, we tackle a more complex instance completion problem. Specifically, considering a (head) entity as an instance, we complete its descriptions by suggesting relation-tail pairs. In other words, we make predictions on $(h, ?, ?)$, suggesting r-t pairs to a given h. Different from the link prediction task, our instance completion task has a more realistic setting without assuming knowing two correlated elements in a triplet. However, such an instance completion problem on KGs is a challenging task, due to a large number of potential r-t pairs for a given h; directly evaluating and ranking all combinations of r and t not only incurs a significant computation cost, but also results in poor performance due to the large number of candidate triplets to consider. For example, such an approach takes 350.7 hours (CPU time) even with an efficient link prediction technique TransE on our JF17K dataset (see our experiments for more detail), resulting in poor performance of 0.0097 for recall@5.

A straightforward solution to this problem is to combine the relation prediction and link prediction tasks [23]; for a given h, we first predict a set of relations using a relation prediction technique, and then predict $(h, r, ?)$ using a link prediction technique for each predicted relation r. Subsequently, with a small set of predicted relations, the number of candidate r-t pairs fed to the link prediction technique can be significantly reduced. However, such an approach still shows subpar performance, as it fails to fully consider the triplewise correlation of the three elements in a triplet, in particular the schema information encoded in the entity-typed

[3]This task is also known as property prediction or recommendation by [23, 69, 149], as these works refer to a triplet by (*entity,property,value*), which is equivalent to our notion of triplet as (*head, relation, tail*).

triplet (h_type, r, t_type). Taking real-world examples from Wikidata, a relation prediction technique may suggest two relations, *headquarters location* and *industry*, to the head entity *Apple Inc.*; a link prediction technique then predicts on *(Apple Inc., headquarters location, ?)* and *(Apple Inc., industry, ?)*, resulting in a ranked list of candidate r-t pairs. Although such a scheme reduces the number of candidate r-t pairs by predicting relevant relations, it still fails to filter out many noisy r-t pairs, such as *headquarters location-software industry* or *industry-Mountain View*. In Wikidata, the entities *Apple Inc.*, *software industry* and *Mountain View* are all associated with types (*enterprise, economic branch* and *city*, respectively). If we have the schema information represented as entity-typed triplets (h_type, r, t_type) —*(enterprise, headquarters location, city)* and *(enterprise, industry, economic branch)*, we could easily filter out such noisy r-t pairs that do not match the schema of the KG (to the given h).

Against this background and to effectively solve our instance completion problem over KGs $(h, ?, ?)$, we propose an end-to-end solution fully leveraging schema information encoded in triplets. Specifically, our solution consists of two components. First, to effectively filter out noisy r-t pairs for a given h, we design RETA-Filter, a Relation-Tail pair filter based on entity types. More precisely, although the schema of a KG implies valuable information about the structure of the KG, open-domain KGs such as Wikidata do not have a fixed schema [9]. Therefore, by extracting entity-typed triplets from an input KG, we build a *head_type-relation-tail_type* tensor, which encodes the schema information of the KG. Based on this tensor, we can efficiently search for candidate r-t pairs schematically matching a given h via tensor operations. In essence, RETA-Filter outputs a set of coarse-grained candidate r-t pairs fitting the schema of the KG. Second, to predict the most promising r-t pairs from the filtered candidates, we design RETA-Grader, a Relation-Tail pair grader leveraging a newly designed KG embedding model scoring each candidate r-t pair with a particular consideration on the plausibility of both the input triplet and its corresponding schema. To achieve this goal, we design two neural network pipelines. On one hand, we learn from a triplet (h, r, t), generating a triplet relatedness feature vector. On the other hand, we learn from the corresponding entity-typed triplet(s) (h_type, r, t_type), generating one or multiple (when h or t has multiple types) relatedness feature vector(s), based on which we then compute a schema relatedness feature vector by taking the minimum value along each feature dimension over these relatedness feature vectors. The basic assumption behind this operation is that for a given valid triplet (h, r, t), the relatedness of all its entity-typed triplets (h_type, r, t_type) should be high; subsequently, the minimum relatedness along each feature dimension is expected to be high. Afterward, we fed the concatenated triplet and schema relatedness feature vectors to a fully connected projection to output the final predicted score of the input triplet (h, r, t).

Our contributions can be summarized as follows:

- We revisit the impractical settings of existing approaches to KG completion, and propose to study a novel instance completion problem with more realistic settings, i.e., predicting relation-tail pairs given a head $(h,?,?)$.

- We propose an end-to-end solution to our instance completion problem. We first propose RETA-Filter, a r-t pair filter based on entity-typed triplets, generating a set of coarse-grained candidate r-t pairs matching the schema of the KG (to the given h). We then propose RETA-Grader, a r-t pair grader leveraging a newly designed KG embedding model scoring each candidate r-t pair with a particular consideration on the plausibility of both the input triplet and its corresponding schema.
- We conduct a thorough evaluation of our proposed techniques compared to a sizable collection of baselines on three real-world KG datasets. Results show that our RETA-Filter is able to generate a set of high-quality candidate r-t pairs for a given h, outperforming the best baseline techniques with 10.61%-84.75% reduction on the candidate set size, under the same candidate quality guarantees. Moreover, our RETA-Grader also significantly outperforms state-of-the-art link prediction techniques on the instance completion task by 16.25%-65.92% across different datasets.

5.2 Related Work

Knowledge Graph completion predicts missing facts in a KG. In the following, we briefly discuss existing work implementing one of the three types of tasks for KG completion: 1) link prediction tasks suggesting one missing element in a triplet; 2) relation prediction tasks predicting missing relations to a given entity; and 3) instance completion task suggesting relation-tail pairs to a given head entity.

5.2.1 Link Prediction Task

In the current literature, KG completion has been mostly formulated as a link prediction task, i.e., predicting one missing element in a triplet, i.e., $(h,r,?)$, $(h,?,t)$ or $(?,r,t)$. To solve this problem, early work resorted to rule/feature-based relational learning, such as association rules [43] or hand-crafted features [71, 97] (e.g., paths linking entities) for link prediction. Recently, KG embedding techniques have been proposed to learn latent representations of entities/relations in a KG, which can then be effectively used for link prediction tasks over the KG. These techniques can be classified into two broad categories. First, translational distance models that exploit distance-based scoring functions to create the embeddings. One representative model of this family is TransE [19], which learns from triplets (h,r,t) such that the relation between the head and tail are preserved as $h+r \approx t$. Several works further improve TransE to capture richer KG structures, such as involving relation-specific hyperplanes [132] or spaces [74, 59, 36], for example. Second, semantic matching models that exploit similarity-based scoring functions. One typical model in that context is RESCAL [95], which represents each entity as a vector and each relation as a matrix, and then uses a bilinear function to model the relation between two entities. Several works extend RESCAL by reducing the complexity of the models [143, 116] or of the training processes [150], by improving the model expressiveness [12], by capturing asymmetric relations [124], by considering unbalanced data in KGs [60], or by modeling non-linear relations using neural networks [117, 29, 113, 10, 92].

In addition, there are also a few works on link prediction combining triplets with other data. According to the sources of such data, these works can be classified into two categories, i.e., data in the KG and third-party data. On one hand, besides triplets linking entities via relations, literals [44, 67], images [77] or types [98, 58, 137] associated with entities in the KG can be combined with triplets to improve the performance on link prediction. On the other hand, some other techniques learn entity/relation embeddings from triplets in a KG jointly with third-party data sources, in particular with text (e.g., Wikipedia articles) [131, 141, 122, 110].

However, we argue in this chapter that such link prediction tasks are impractical in the sense that they assume knowing two of the elements in a triplet, which is often not the case in practice. Those two elements must be correlated to make meaningful predictions, which otherwise result in meaningless outputs such as (*Marie Curie*, *headquarters location*, ?). Therefore, we study an instance completion problem in this chapter, suggesting relation-tail pairs for a given (head) entity.

5.2.2 Relation Prediction Task

KG completion problems have also been studied as a relation prediction task, suggesting missing relations to a given entity [8, 42, 9, 23, 69, 149]. Specifically, for a given entity, the objective is to suggest a list of relations (so-called properties by some of these works) which are relevant to the entity. For example, Zangerle et al. [149] built a "property suggester" for Wikidata, suggesting candidate properties based on association rules learnt from existing triplets in Wikidata. Lajus et al. [69] studied the problem of determining obligatory relations for a given entity in a KG by extracting and using the class hierarchy (of entities) in the KG. Cao et al. [23] designed a relation prediction technique by applying an attention-based graph neural network model to a bipartite entity-relation graph built from a KG. Recoin [9] suggests properties to an entity on Wikidata by collaboratively using the information about other entities that are similar to that entity; to ensure the high quality of the suggested properties, it sometimes involves much prior knowledge when defining the similarity between entities on Wikidata. For example, for entities of type human, a (Boolean) similarity is defined as whether the two entities have the same *occupation* or not.

Our work differs from these techniques by considering a more complex problem, that of predicting relation-tail pairs for a given (head) entity. Although these relation prediction techniques can be combined with link prediction techniques to perform our instance completion task, such an approach yields subpar performance, as it does not fully leverage the schema information encoded by the triplets.

5.2.3 Instance Completion Task

A closely related work to our instance completion problem is OKELE [23], which suggests relation-tail pairs to a long-tail head (i.e., a less popular/frequent entity) in a KG, using open Web data. Specifically, for a given head h, OKELE first implements a relation prediction technique by applying an attention-based graph neural network model to a bipartite entity-relation graph built from a KG, predicting a list of relations; for each suggested relation r, it then extracts and verifies potential tails from open Web data including semi-structured vertical websites, unstructured plain text in Web content and structured HTML tables. To the best of our knowledge, this is the only work in the current literature considering the instance completion task, i.e., predicting relation-tail pairs to a given head. However, OKELE differs from our work by extensively using open Web data. In contrast, our proposed solution only requires triplets and entity types from a KG, without the need of involving any extra data sources.

We also note that the so-called instance reconstruction task on KGs has been defined by [152] for n-ary (or multi-fold) relational facts, where an n-ary relation consisting of a set of relations $\{r_1, r_2, ..., r_n\}$ links multiple entities $\{e_1, e_2, ..., e_n\}$, respectively. For example, the n-ary relation "PeopleMariage" containing the following three relations *person*, *spouse*, *location*, links three entities *Kobe Bryant*, *Venessa Bryant* and *California*. The authors define the instance reconstruction task as follows: given an n-ary relation $\{r_1, r_2, ..., r_n\}$ and a subset of the entities linked by this relation, predict the rest of the entities, such as $\{e_1, ?, ..., ?\}$. With the help of relation paths [133], this task boils down to a link prediction task, i.e., predicting $(e_1, r_1 r_2, ?)$, $(e_1, r_1 r_3, ?)$ and so on. Therefore, it fundamentally differs from our instance completion task, where we do not assume any information about the relations.

5.3 Schema-Aware Instance Completion

To solve our instance completion problem on KGs, we propose an end-to-end solution leveraging the explicit and implicit schema information encoded through the triplets. More precisely, we first design RETA-Filter, a r-t pair filter based on entity-typed triplets, generating a set of coarse-grained candidate r-t pairs matching the schema of the KG (to the given h). We then propose RETA-Grader, a r-t pair grader leveraging a newly designed KG embedding model scoring and ranking these candidate r-t pairs with a particular consideration on the plausibility of the triplets and their corresponding schema.

5.3.1 RETA-Filter

Our RETA-Filter is designed to generate a set of coarse-grained r-t pairs by extracting and leveraging the schema of an input KG. Specifically, a KG contains a set of triplets (h, r, t), where $h, t \in E$ and $r \in R$; E and R are the sets of all entities and relations in the KG, respectively. Each entity h (or t) can have one or multiple types $h\_type_1, h\_type_2, ... \in T$, where T is the set of all entity types in the KG (we will discuss later the case where an entity does not have any type). Subsequently, the schema of a KG can characterized by a set of entity-typed triplets (h_type, r, t_type), indicating that an entity of type h_type could be linked to an entity of type t_type via a relation r. Such information can serve as an important guideline to identify eligible r-t pairs for a given h, thus avoiding many meaningless predictions.

To implement our filter, we first extract entity-typed triplets (h_type, r, t_type) from the triplets (h, r, t) in a KG, by considering all the combinations of the types of h and t for each triplet (h, r, t). For example, if h and t have m and n types, respectively, we extract $m \times n$ entity-typed triplets. Afterward, we represent all these entity-typed triplets as a *head_type-relation-tail_type* tensor $\mathbf{F} \in \mathbb{B}^{|T^h| \times |R| \times |T^t|}$, where T^h and T^t are the sets of all head and tail types, respectively, while R is the set of all relations in the KG. Note that T^h and T^t are the same as T in practice, as they are both the set of all entity types. We use different notations for them to distinguish the semantic meaning of the corresponding dimensions of $\mathbf{F}$, where the first and third dimensions of $\mathbf{F}$ refer to head and tail types, respectively. Finally, by representing h using its type vector $\vec{h_T} \in \mathbb{B}^{|T_h|}$ and all tails using a type matrix $\mathbf{M}_T \in \mathbb{B}^{|E| \times |T|}$, we can efficiently compute the candidate r-t pairs for h via a tensor product:

$$\mathbf{S} = \vec{h_T} \times_{T_h} \mathbf{F} \times_{T_t} \mathbf{M} \tag{5.1}$$

where $\times_n$ denotes the mode-n tensor product [65]. The resulting matrix $\mathbf{S} \in \mathbb{N}^{|R| \times |E|}$ encodes the number of matches between h and each r-t pair, under the extracted schema. Figure 5.2 shows a toy example of our filter. To implement this filter, we have to take into account the following two practical considerations:

- **Frequency of entity-typed triplets.** Entity-typed triplets (h_type, r, t_type) encode the schema of the corresponding KG. When converting the triplets of a KG into entity-typed triplets, we also obtain the frequency of each entity-typed triplets, indicating how many times it appears in the KG. Intuitively, a low frequency indicates a small contribution of the corresponding entity-typed triplet to the schema of the KG, which could also be considered as noise. Subsequently, we could select frequent entity-typed triplets (whose frequencies are higher than a threshold α) to build $\mathbf{F}$. However, a too

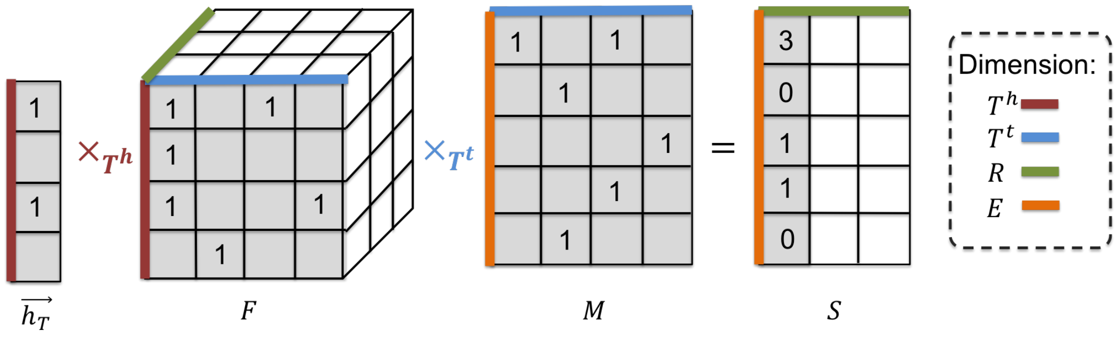
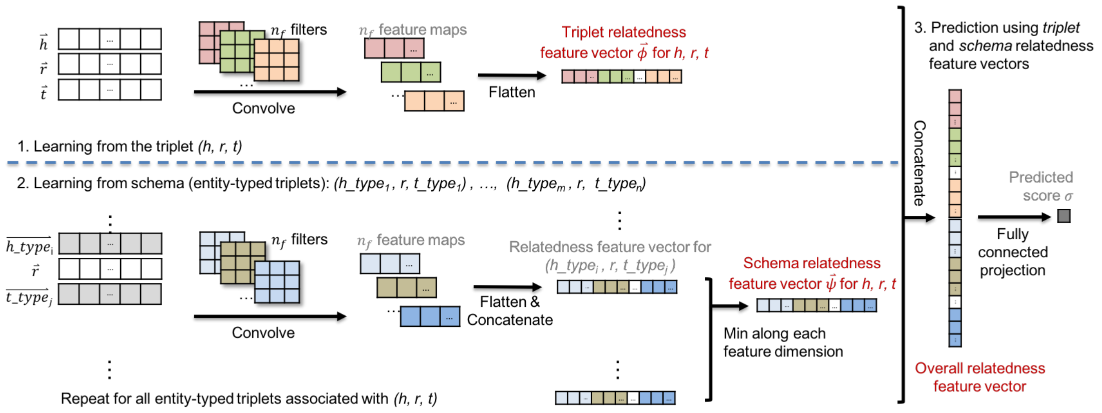


Figure 5.2 – A toy example of our filter, where we highlight an example of computation for one relation (in gray). Given an head entity h, RETA-Filter fetches its type vector $\vec{h_T}$ and multiply it with the tensor **F**. The resulting matrix is then multiplied with the type matrix $\mathbf{M}_T$ in order to compute the candidate r-t pairs for h.

high value of α could also remove useful entity-typed triplets, resulting in an incomplete schema captured by **F**. In other words, α controls the quality of the extracted schema, and subsequently balances the tradeoff between the size of the candidate (r-t pair) set and its coverage of true r-t pairs being included in the candidate set. On one hand, a too low value of α could include noisy entity-typed triplets in **F**, resulting in a large candidate set with noisy and redundant r-t pairs. On the other hand, a too high value of α captures incomplete schema information when building **F**, resulting in a small but low-coverage candidate set (i.e., missing true r-t pairs).

We note that once we select frequent entity-typed triplets (whose frequencies are higher than α), we build **F** as a Boolean tensor without considering the absolute frequency of each selected entity-typed triplet, since the absolute frequencies of these frequent entity-typed triplets are not useful when representing the schema of the KG. For example, an entity-typed triplet *(human, occupation, profession)* with a frequency of 10,000 does not necessarily mean that it is (ten times) more important than an entity-typed triplet *(enterprise, headquarters location, city)* with a frequency of 1,000 when representing the structure of the KG; such a frequency difference may just be caused by the varying popularity of different entity types in a KG. Therefore, we do not distinguish the selected (frequent) entity-typed triplets by their frequencies when building **F**.

- **Number of matches between r-t pairs and h.** The resulting matrix **S** encodes the number of matches between h and each r-t pair under the extracted schema. Intuitively, a higher number of matches indicates a higher plausibility of the corresponding r-t pair being a candidate for h. Following the example in Figure 5.2, the number of matches between h and the first r-t pair is $\mathbf{S}_{0,0} = 3$. Subsequently, instead of taking all r-t pairs with non-zero matches (in **S**) as the candidate r-t pairs, we could further select higher-quality r-t pairs whose numbers of matches are higher than a threshold β as candidate r-t pairs. Obviously, a



1. Learning from the triplet (h, r, t)
n_f filters
Convolve
n_f feature maps
Flatten
Triplet relatedness feature vector $\vec{\phi}$ for h, r, t
2. Learning from schema (entity-typed triplets): (h_type₁, r, t_type₁), …, (h_typeₘ, r, t_typeₙ)
Relatedness feature vector for (h_typeᵢ, r, t_typeⱼ)
Flatten & Concatenate
Schema relatedness feature vector $\vec{\psi}$ for h, r, t
Min along each feature dimension
Repeat for all entity-typed triplets associated with (h, r, t)
Concatenate
3. Prediction using triplet and schema relatedness feature vectors
Fully connected projection
Predicted score σ
Overall relatedness feature vector

Figure 5.3 – Architecture of RETA-Grader.

lower value of β selects more candidate r-t pairs and thus leads to a higher coverage of true r-t pairs being included in the candidate set, and vice versa. In essence, β directly balances the tradeoff between the size of the candidate (r-t pair) set and its coverage.

In summary, our RETA-Filter is designed to efficiently generate a set of coarse-grained r-t pairs by matching with the KG schema extracted from the entity-typed triplets of a KG. We use two tunable parameters α and β balancing the tradeoff between the size of the candidate (r-t pair) set and its coverage. By varying α and β, our RETA-Filter is able to achieve the best Pareto frontier when trading off the size of the candidate (r-t pair) set and its coverage, compared to a sizable collection of baselines (see our experiments for more detail).

5.3.2 RETA-Grader

Based on the set of candidate r-t pairs provided by our RETA-Filter, our RETA-Grader further evaluates and ranks these candidate r-t pairs considering the plausibility of both the triplet and its corresponding schema using a subtly designed KG embedding model. Figure 5.3 shows our embedding model consisting of three parts. Specifically, for each fact (h, r, t), it 1) learns to capture the structural information of the triplet (h, r, t), generating a *triplet relatedness feature vector*, and 2) learns to capture the corresponding schema information encoded by the triplet, generating a set of relatedness feature vectors, one for each entity-typed triplet (h_type, r, t_type), and then merges them into a unique *schema relatedness feature vector*. Finally, it concatenates the triplet and schema relatedness feature vectors into an *overall relatedness feature vector* to output a final prediction score, measuring the plausibility of both the input triplet and its corresponding schema. In the following, we discuss the detailed design of these three modules.

Learning from triplets

To learn from a triplet (h, r, t), we use a Convolutional Neural Network (CNN) to model the interaction between the three elements in the triplet, i.e., head h, relation r and tail t. We adopt a CNN here, as it has been successfully used to learn from triplets in KGs by previous work [29, 91]. As shown in Figure 5.3, we start by concatenating the three embedding vectors $\vec{h}, \vec{r}, \vec{t} \in \mathbb{R}^K$ (K is the embedding dimension), resulting in a matrix $I \in \mathbb{R}^{3\times K}$. The matrix I is fed to a 2D convolutional layer with n_f filters of size 3×3. We set the filter size to 3×3 to capture the triple-wise relatedness between the embeddings of h, r and t. This convolutional layer returns n_f feature maps of size $K-2$ each, which are then flattened into a *triplet relatedness feature vector* $\vec{\phi} \in \mathbb{R}^{1\times n_f(K-2)}$. This relatedness vector $\vec{\phi}$ characterizes the plausibility of a fact (h, r, t) of being true.

Learning from schema (entity-typed triplets)

The schema information encoded by the triplet (h, r, t) is also an important predictor for the plausibility of the triplet. To capture such schema information, we extract and learn from entity-typed triplets considering all the combinations of the types of h and t. When h and t have m and n types, respectively, we extract a set of mn entity-typed triplets $\{(h\_type_i, r, t\_type_j)|1 \leq i \leq m, 1 \leq j \leq n\}$. Afterward, we learn from each of these entity-typed triplet (h_type_i, r, t_type_j) to generate its relatedness feature vector, using a similar method as for learning from triplets. Specifically, as shown in Figure 5.3, by concatenating three embedding vectors for head type $\overrightarrow{h\_type_i}$, relation $\vec{r}$, tail type $\overrightarrow{t\_type_j}$, we also resort to a 2D convolutional layer with n_f filters of size 3×3 to capture the triple-wise relatedness between $\overrightarrow{h\_type_i}$, $\vec{r}$, and $\overrightarrow{t\_type_j}$; the resulted feature maps are then flattened into a relatedness vector of size $1 \times n_f(K-2)$. Note that the filters in this module are different from the filters in the first module. We repeat this process for all the mn entity-typed triplets associated with the input triplet. Finally, we concatenate these relatedness feature vectors into a matrix of size $mn \times n_f(K-2)$, and then take the minimum value along each feature dimension, resulting in a unique *schema relatedness feature vector* $\vec{\psi}$. The basic assumption behind this min operation is that for a true triplet, the relatedness of the three elements $(h\_type,\ r,\ t\_type)$ of *any* entity-typed triplets should be high; subsequently, the minimum relatedness along each feature dimension is expected to be high. Similar ideas have also been successfully applied by previous works to merge relatedness scores in a neural network [47].

Prediction using triplet and schema relatedness feature vectors

In the two previous modules, for each triplet, we obtain a triplet and a schema relatedness feature vectors ($\vec{\phi}$ and $\vec{\psi}$, respectively). We then concatenate $\vec{\phi}$ and $\vec{\psi}$ into an *overall relatedness* *feature vector* of size $2n_f(K-2)$. Finally, we use a fully connected layer to output the predicted score σ from the overall relatedness feature vector.

RETA-Grader Training Process

To train our RETA-Grader model, we minimize a softplus loss, which is defined as the negative log-likelihood of the logistic model:

$$\sum_{\omega \in \Omega} log(1+e^{-\sigma(\omega)}) + log(1+e^{\sigma(\omega')}) \tag{5.2}$$

where Ω represents the set of training triplets. For each triplet $\omega = (h, r, t)$, one negative sample ω' is generated by randomly corrupting one element in the triplet h, r or t. $\sigma(\omega)$ and $\sigma(\omega')$ denote the predicted score of our RETA-Grader model for the true fact ω and the negative fact ω', respectively. The loss function in Eq. 5.2 is minimized using the Adam optimizer [64], and the model parameters are learnt via back propagation. We use rectified linear units (ReLU) as the activation function [68] and batch normalization [57] after the two CNN layers for fast training.

Practical considerations on design choices

To implement our RETA-Grader, we have to take into account the following two practical considerations:

- **Number of types learnt per entity.** Our RETA-Grader evaluates a triplet considering the plausibility of both the input triplet and its corresponding schema, where the schema information is represented by a set of entity-typed triplets. For a given triplet (h, r, t) where h and t have m and n types, respectively, the second module in Figure 5.3 is repeated mn times when evaluating this triplet; this could incur a large computation overhead for large values of m and n. To overcome this issue, we choose to learn at most top-k types (according to the frequency of types in a KG dataset) for each entity, resulting in $\min(mk, nk, k^2)$ repetitions of the second module, rather than mn times. We empirically show in our experiments below that a small value of k is able to achieve a good performance on our instance completion task.

- **Entities without types.** Although most entities in modern KGs are associated with one or multiple types, there is still a small portion of entities without types. To accommodate these entities in our instance completion task, we make the following adaptation to our proposed solution.

  First, to generate a set of candidate r-t pairs for a head h, if any entity (h or t) does not have a type, we assume the entity could be associated with any type, such that we do not miss any potential candidates. Specifically, we have the following three cases: 1) when h has no type but t has types, $\vec{h_T}$ becomes an all-one vector, where our RETA-Filter still considers the match between r and t; 2) when t has no type but h has types, the corresponding row of $\mathbf{M}$ becomes an all-one vector, where our RETA-Filter still considers the match between h and r; 3) when both h and t have no type, $\vec{h_T}$ and the corresponding row of $\mathbf{M}$ become all-one vectors, where our RETA-Filter generates a full matrix S (without zero entry); in the last case in particular, if we set the threshold $beta$ to 1, our RETA-Filter indeed degrades to keep all combinations of r-t pairs as candidates.

Second, to let our RETA-Grader learn from entities without types, we assign an "unknown" type to such entities, and then keep the same processing pipeline for score prediction. For example, for a triplet (h, r, t) where h has no type and t has n types, the set of entity-typed triplets becomes $\{(unknown\_type, r, t\_type_j)|1 \leq j \leq n\}$. From a schema point of view, such an "unknown" type could be linked to any type in a KG via a relation, which makes the schema relatedness feature vector less discriminative for prediction. Subsequently, the fully projected layer in the third module in Figure 5.3 is able to automatically learn more from the triplet relatedness feature vector to make predictions.

The implementation of RETA and used datasets are available here[4].

5.4 Experiments

We conduct an extensive evaluation on our instance completion task. In the following, we start by presenting our experimental setup, followed by our results and discussions.

5.4.1 Experimental Setup

Datasets

We use three popular KG datasets ***JF17K***, ***FB15K*** and ***HumanWiki*** in our experiments. More precisely, the JF17K and FB15K datasets are extracted from Freebase by [133] and [19], respectively. We extract the HumanWiki dataset from Wikidata by extracting all triplets involving a head entity of type human (i.e., class Q5 human on Wikidata)[5]. As our instance completion task suggests relation-tail pairs for a given head, for each unique head, we randomly split its r-t pairs in the datasets into 80% training and 20% test datasets. Table 5.1 shows the main

[4]https://github.com/eXascaleInfolab/RETA_code/

[5]We choose to extract the HumanWiki dataset because one of the state-of-the-art relation prediction techniques, Recoin (see below), is specifically designed for "human" instances on Wikidata, using prior knowledge for better performance on relation prediction.

Table 5.1 – Statistics of the datasets.

Dataset	**JF17k**	**FB15k**	**HumanWiki**
#Entity	9,233	14,579	38,949
#Entity w/ types	9,174	14,417	34,470
#Entity w/o types	59	162	4,479
#Type	511	588	388
#Type per entity	6.45	10.02	1.08
#Relation	326	1,208	221
#Fact	19,342	154,916	108,199
#Fact w/ types on both h and t	19,015	144,117	87,150
#Fact w/ types on either h or t	322	10,776	21,049
#Fact w/o types	5	23	0

statistics of the three datasets.

Baselines

We compare our proposed method against a sizable collection of state-of-the-art techniques from the following three categories.

- *Relation (property) prediction techniques.* **BPR** [104] is a recommendation technique generating an entity-specific ranked list of relations (where we consider the relation prediction task as recommending relations to entities). Property suggester (**WikiPS**) [149] recommends relations to an entity using association rules learnt from existing triplets in Wikidata; this technique is an API[6] on Wikidata, and thus can only be applied to our HumanWiki dataset. **Recoin** [9] suggests relations to an entity by collaboratively using the information about other similar entities, where the similarity is manually defined using a Boolean similarity function that considers two entities as similar if they share at least one type. In particular, it has a special setting for entities of type *human* using heuristics and prior knowledge, where the Boolean similarity is defined as whether two humans have the same occupation or not, i.e., whether the two head entities of type *human* have the same tail entity linked via the relation *occupation* or not. **OKELE** [23] predicts relations associated with an entity using an attention-based graph neural network model[7]. As these relation prediction techniques suggest a ranking list of relations to a given head, we take the top-N relations from the list as

[6]https://www.wikidata.org/w/api.php?action=help&modules=wbsgetsuggestions

[7]Note that we consider only the relation prediction technique from OKELE as a baseline technique in this chapter, as its link prediction technique extensively uses open Web data, which fundamentally differs from our instance completion task requiring only triplets and entity types from a KG, without the need of involving any extra data sources (see more detail in our Related Work section).

the predicted relations; tuning N actually balances the tradeoff between the size and the coverage of the resulting candidate (r-t pair) set.

- *Tail candidate refinement techniques.* Based on the predicted list of relations returned by the above techniques, a straightforward approach to form a set of candidate r-t pairs is to combine each predicted relations with all entities (**All**). One possible improvement to this step is to have a filtered list of relevant t rather than using all entities, i.e., generating a subset of potential t for given h and r. Note that this differs from the link prediction task, as we generate a subset of t rather than ranking r-t pairs. In the current literature, an entity relatedness prediction task has been introduced by [152] for n-ary relational facts where an n-ary relation links multiple entities $\{e_1, e_2, e_3, ..., e_n\}$); this task predicts the relatedness between entities in such an n-ary relational fact, in order to perform an instance reconstruction task $\{e_1, e_2, ?, ..., ?\}$ (see more detail in our Related Work section). The proposed techniques by [152] evaluate the relatedness between two entities, i.e., whether two entities should be linked by a relation or not, which can thus be adopted for our tail candidate refinement. Specifically, two techniques have been proposed by [152]. First, a relatedness affiliated embedding (**RAE**) model, which learns a multi-layer perceptron neural network to predict a relatedness score between two entities, and considers them to be relevant if the score is higher than a threshold γ. Tuning γ balances the tradeoff between the size and the coverage of the resulting candidate (r-t pair) set. Second, a schema based predictor (**Sch**), which leverages the type requirements on the entities dictated by the schema of a relation, generating a set of (tail) entities schematically matching a given relation. Subsequently, one can also combine the refined sets of candidate tails from RAE and Sch by taking their intersection (**RAE&Sch**).

- *Link prediction techniques.* Based on a set of candidate r-t pairs for a given h, we apply the following link prediction techniques.

  First, we consider classical link prediction techniques evaluating the plausibility of a triplet. The translational distance models we consider include: **TransE** [19], which learns to preserve the relation between two entities as $h + r \approx t$; **TransH** [132], which extends TransE to better capture multi-mapping relations using relation-specific hyperplanes; **TransR** [74], which introduces relation-specific projections to also better capture multi-mapping relations; **TransD** [59], which further extends TransR by decomposing the projection matrix into a product of two vectors for an improved efficiency. The semantic matching models we consider are as follows: **Rescal** [95], which represents each entity as a vector and each relation as a matrix, and uses a bilinear function to model the relation between a pair of entities; **DistMult** [143], which simplifies Rescal by representing each relation embedding as a diagonal matrix; **ComplEx** [124], which further extends DistMult in the complex space in order to better model both symmetric and asymmetric relations; **Analogy** [76], which explicitly models analogical structures in multi-relational KG embeddings; **SimplE** [63],

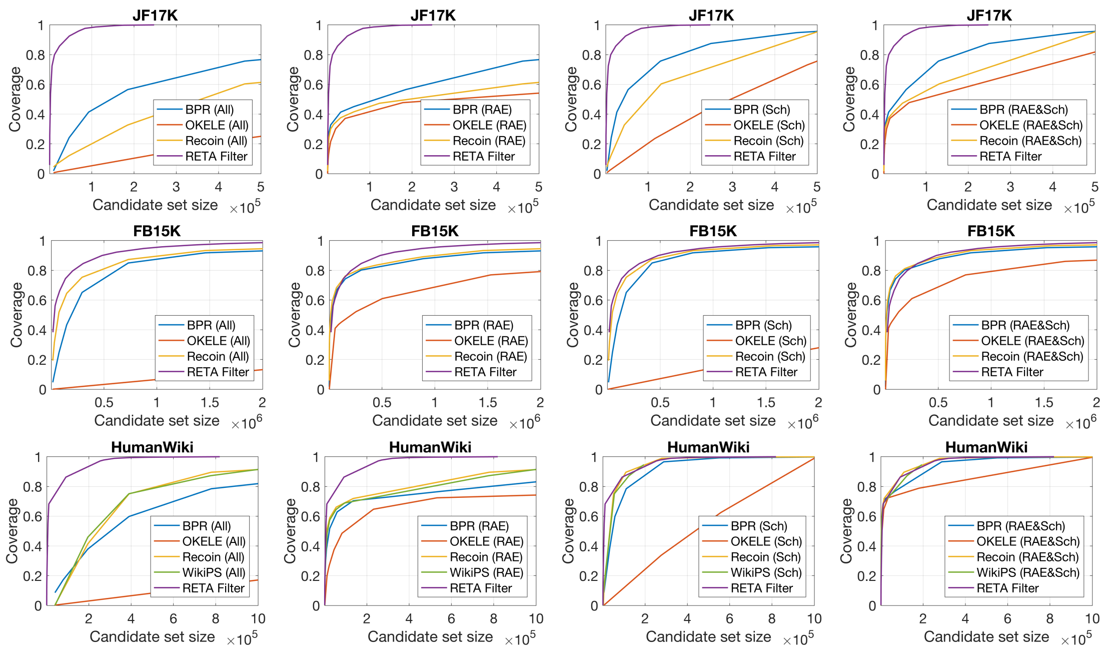


Figure 5.4 – Tradeoff between the size and the coverage of the resulting candidate (r-t pair) set from different filtering techniques. Note that our RETA-Filter has a unique tradeoff line on each dataset, as it does not use any tail candidate refinement techniques.

which is an expressive and interpretable KG embedding techniques based on canonical polyadic tensor decomposition; **RotatE** [120], which defines a relation as a rotation from a head to a tail in the complex vector space, capturing richer relation patterns. For each of these link prediction technique, we use the hyperparameters as recommended by [52]. We also consider a variation of RETA-Grader as an additional baseline, where we learn from triplets only, without learning from the schema (entity-typed triplets); we refer to this baseline as **RETA-Grader (no type)**.

Second, we also consider link prediction techniques using additional information about entity types as baselines, as our RETA-Grader also evaluates a triplet considering the plausibility of both the input triplet and its corresponding entity-typed triplets. **TypeDM** and **TypeComplex** [58] are extended from DistMult and ComplEx, respectively, by explicitly modeling entity type compatibility.

For our RETA-Grader, we set the number of types learnt per entity $k = 2$, 1 and 4, and the number of Filters $n_f = 50$, 200 and 100 for JF17K, FB15K and HumanWiki, respectively. More details on these parameter sensitivity study will be present later.

Evaluation Protocol

To implement our instance completion task, for a test h, we first generate a set of candidate r-t pairs, and then score and rank them. We evaluate this task in two steps.

First, we evaluate the quality of the generated candidate r-t pair sets. We consider two metrics: 1) the *coverage* of the candidate set (i.e., the percentage of the ground truth r-t covered by the candidate set), and 2) the *size* of the candidate set. Intuitively, a good candidate set should have higher coverage and a small size at the same time. Due to the intrinsic tradeoff between the *coverage* and *size* of the candidate set, we plot and compare the Pareto frontier of different techniques. Specifically, to generate a set of candidate r-t pairs using our baseline techniques, we can use any *relation prediction technique* (**BPR**, **WikiPS**, **Recoin**, **Recoin_Human** or **OKELE**) combined with any *tail candidate refinement technique* (**All**, **RAE**, **Sch** or **RAE&Sch**). We first take the top N relations generated by a relation prediction technique and then use one tail candidate refinement technique to generate a set of candidate r-t pairs. By tuning N (and also γ for RAE when applicable), we balances the tradeoff between the size and the coverage of the resulting candidate (r-t pair) set. For our method, we tune α and β to balance this tradeoff.

Second, by fixing the set of candidate r-t pairs ensuring 95% coverage using our RETA-Filter, we evaluate the performance of different (link prediction) techniques and our RETA-Grader when ranking these candidate r-t pairs. We report Recall (Rec), Mean Average Precision (MAP), and Normalized Discounted Cumulative Gain (NDCG).

5.4.2 Performance on Filtering r-t Pairs

In this section, we evaluate the first step of our instance completion task by investigating the performance of filtering r-t pairs using different techniques. Figure 5.4 shows the Pareto frontier when trading off the size and the coverage of the candidate r-t pair set using each baseline technique and our RETA-Filter.

We observe that our RETA-Filter outperforms state-of-the-art techniques in general by achieving a better tradeoff in most cases, i.e., the resulting Pareto frontier is closer to the upper-left corner of the plot. Moreover, we find that adding tail candidate refinement techniques to relation prediction techniques does indeed help improve the quality of the resulting r-t pair sets, where combining RAE and Sch (RAE&Sch) shows the best performance.

Furthermore, under the tail candidate refinement technique RAE&Sch, we find that Recoin (RAE&Sch) is the most competitive baseline, which is able to achieve comparable results to our RETA-Filter on FB15K and HumanWiki. More precisely, RETA-Filter is better than Recoin (RAE&Sch) by achieving a slightly smaller candidate set under the same coverage

Table 5.2 – Size of candidate set with at least 95% coverage. *WikiPS works only on HumanWiki and fails to reach 95% coverage, due to the fact that its suggested relations (returned from its API online) do not include all relations in our HumanWiki dataset.

Method	JF17k	FB15k	HumanWiki
BPR (RAE&Sch)	450,933	1,526,529	287,247
OKELE (RAE&Sch)	699,890	7,734,614	1,011,185
Recoin (RAE&Sch)	514,930	1,567,367	278,253
WikiPS (RAE&Sch)*	N/A	N/A	-*
RETA-Filter	68,745	1,048,053	248,721

when the coverage is higher than 82% on FB15K (95% on HumanWiki), while we observe the opposite result when the coverage is lower than 82% on FB15K (95% on HumanWiki). However, Recoin achieves such results by using heuristics and prior knowledge on the structure of a KG (i.e., manually defined similarity between entities), in particular on HumanWiki where the similarity between entities is defined as whether two humans have the same occupation or not, i.e., whether the two head entities of type human have the same tail entity linked via the relation occupation or not. In addition, WikiPS (RAE&Sch) also performs well on HumanWiki (the second best baseline). However, it cannot reach high coverage, due to the API limitations, as the returned relations do not include all relations in our HumanWiki dataset.

In practice, as the first step of our instance completion task, the r-t pair filtering step should generate a set of candidate r-t pairs with a high coverage (95% or even higher), in order to let the following link prediction techniques or our RETA-Grader identify the true r-t pairs by scoring and ranking the candidate r-t pairs. Otherwise, the prediction on a candidate set with a low coverage will certainly lead to low performance on our instance completion task, as a candidate set with a low coverage excludes many ground-truth r-t pairs, which can never be correctly predicted. In other words, the coverage of the candidate set in this step is indeed the upper bound of recall@N when ranking the candidate r-t pairs in the following prediction step. Therefore, we set the coverage to 95% and compare the size of the candidate sets generated by different methods using the best-performing tail candidate refinement technique RAE&Sch. Table 5.2 shows the results. We see that our RETA-Filter consistently outperforms the baseline techniques, and reduces the size of the candidate set by 84.75%, 31.34% and 10.61% compared to the best performing baselines on JF17k, FB15k and HumanWiki, respectively. In the following experiments, we use our RETA-Filter to generate the set of candidate r-t pairs with 95% coverage.

Table 5.3 – Performance of different methods on our instance completion task on JF17K. We highlight the top-2 best-performing techniques for each metric.

Method	JF17K			
	Rec@10	Rec@5	MAP	NDCG
TransE	0.0682	0.0321	0.0230	0.0233
TransH	0.0410	0.0188	0.0173	0.1248
TransR	0.0657	0.0316	0.0229	0.1343
TransD	0.0465	0.0238	0.0179	0.1253
Rescal	0.0074	0.0057	0.0048	0.0791
Distmult	0.0892	0.0499	0.0367	0.1392
ComplEx	0.0841	0.0523	0.0317	0.1377
Analogy	0.1129	0.0679	0.0414	0.1424
SimplE	0.0881	0.0398	0.0290	0.1336
RotatE	**0.1745**	**0.0996**	**0.0529**	**0.1694**
RETA-Grader (no type)	0.1414	0.0976	0.0528	0.1600
TypeDM	0.1481	0.0524	0.0452	0.1651
TypeComplex	0.0665	0.0425	0.0203	0.1204
RETA-Grader	**0.1916**	**0.1153**	**0.0615**	**0.1855**

Table 5.4 – Performance of different methods on our instance completion task on FB15K. We highlight the top-2 best-performing techniques for each metric.

Method	FB15K			
	Rec@10	Rec@5	MAP	NDCG
TransE	0.0411	0.0162	0.0242	0.1654
TransH	0.0216	0.0069	0.0175	0.1505
TransR	0.0441	0.0124	0.0240	0.1648
TransD	0.0253	0.0086	0.0196	0.1566
Rescal	0.0009	0.0004	0.0002	0.0566
Distmult	0.0596	0.0260	0.0245	0.1559
ComplEx	0.1235	0.0683	0.0597	0.1994
Analogy	**0.1496**	**0.0841**	**0.0625**	0.2017
SimplE	0.0483	0.0198	0.0245	0.1536
RotatE	0.0583	0.0341	0.0359	0.1805
RETA-Grader (no type)	0.1262	0.0653	0.0538	**0.2114**
TypeDM	0.1274	0.0693	0.0576	0.1999
TypeComplex	0.0985	0.0552	0.0439	0.1755
RETA-Grader	**0.2104**	**0.1288**	**0.1037**	**0.2658**

Table 5.5 – Performance of different methods on our instance completion task on HumanWiki. We highlight the top-2 best-performing techniques for each metric.

Method	HumanWiki			
	Rec@10	Rec@5	MAP	NDCG
TransE	0.0098	0.0008	0.0147	0.1140
TransH	0.0110	0.0007	0.0119	0.1086
TransR	0.0052	0.0006	0.0124	0.1118
TransD	0.0050	0.0005	0.0108	0.1061
Rescal	0.0000	0.0000	0.0000	0.0474
Distmult	0.1400	0.1035	0.0767	0.1747
ComplEx	0.0986	0.0586	0.0416	0.1365
Analogy	0.0136	0.0064	0.0077	0.0891
SimplE	0.1151	0.0812	0.0573	0.1520
RotatE	0.0429	0.0064	0.0171	0.1172
RETA-Grader (no type)	**0.1606**	0.1109	**0.0860**	0.2038
TypeDM	0.1285	**0.1143**	0.0789	**0.2079**
TypeComplex	0.0581	0.0015	0.0165	0.1082
RETA-Grader	**0.2049**	**0.1545**	**0.1166**	**0.2332**

5.4.3 Performance on Ranking r-t Pairs

Based on the candidate r-t pair set generated by our RETA-Filter, we then evaluate the performance of our RETA-Grader on our instance completion task. Table 5.3 and Table 5.4 and Table 5.5 show the results. We observe that our RETA-Grader consistently outperforms all baseline techniques on our instance completion task in general. Taking MAP as an example, it yields an improvement of 16.25%, 65.92% and 35.58% over the best performing baselines on JF17K, FB15K and HumanWiki, respectively. Moreover, compared to RETA-Grader (no type), which is the variant of our proposed model without learning from the schema, our RETA-Grader learning from both triplets and their corresponding schema significantly achieves better performance; this shows that learning from entity-typed triplets is indeed helpful for the instance completion task. Compared to link prediction techniques using additional information about entity types, our RETA-Grader yields better performance by evaluating the plausibility of both the input triplet and its corresponding schema using a subtly designed KG embedding model. Note that TypeComplex, which further considers entity types on top of the ComplEx model, underperforms ComplEx; opposite results are reported in [58]. This is due to the different implementations of the ComplEx model. More precisely, we tested two different implementation of ComplEx from [58] and [52], respectively, and report the results from the best-performing implementation from [52] in this chapter.

Moreover, we further investigate the performance of our RETA-Grader when handling entities without types, where we assign an artificially created “unknown” type to such entities. To this

Table 5.6 – Performance of RETA-Grader on different test facts.

Test Facts	JF17K		FB15K		HumanWiki	
	MAP	NDCG	MAP	NDCG	MAP	NDCG
known types	0.0614	0.1857	0.1075	0.2780	0.1440	0.2912
unknown type	0.0651	0.1577	0.0790	0.1846	0.0608	0.1154

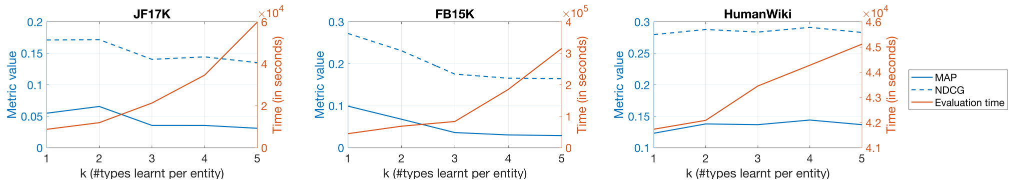


Figure 5.5 – Impact of the number of types learnt per entity k on instance completion performance and evaluation time.

end, we divide all test facts into two sets depending on whether a fact involves an "unknown" type or not: 1) test facts with types on both h and t (denoted as ***known type***); and 2) test facts involving "unknown" type (denoted as ***unknown type***), including both facts with types on either h or t and facts without types at all. We then compare the performance of our RETA-Grader on these two sets of test facts. Table 5.6 shows the results. We observe that the performance on the facts with *known type* is generally better than the facts with *unknown type*. On one hand, for the facts with *known type*, our RETA-Grader is able to fully leverage the corresponding entity-typed triplets to evaluate the plausibility of a triplet from a schema perspective, resulting in better performance. On the other hand, for the facts with *unknown type*, our RETA-Grader can only evaluate the schematic plausibility based on the assumption that an "unknown" type could be linked to any type in a KG via a relation, which makes the schema relatedness feature vector less discriminative for prediction.

We note that the results of *known type* in Table 5.6 are very close to the results on all test facts from Table Table 5.3 and Table 5.4 (MAP and NDCG) on JF17K and FB15K, but not Table 5.5 on HumanWiki. This difference can be explained by the fact that facts with *known type* dominate the set of all test facts on JF17K and FB15K, representing 98.3% and 93.0% of all test facts on JF17K and FB15K, respectively, while this statistic on HumanWiki is 80.5% (see Table 5.1 for more detail).

5.4.4 Parameter Sensitivity Study

We also study the impact of two key parameters in our RETA-Grader, i.e., the number of types learnt per entity k and the number of filters n_f, on both instance completion performance and

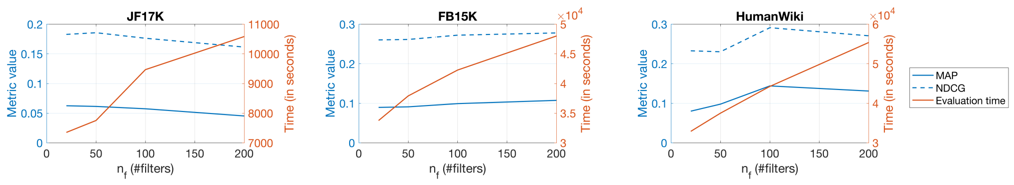


Figure 5.6 – Impact of the number of filters n_f on instance completion performance and evaluation time.

evaluation time. In this section, we focus on test facts with *known type* only, i.e., test facts with known types on both h and t, as these test facts can be fully evaluated by our RETA-Grader considering the plausibility of both the input triplet and its corresponding schema. In addition, these test facts also dominate the set of all test facts.

Number of types learnt per entity by RETA-Grader

Our RETA-Grader evaluates a triplet considering the plausibility of both the input triplet and its corresponding schema, where the schema information is represented by a set of entity-typed triplets. For entities with many types, the size of this entity-typed triplet set is large, incurring a large computation overhead. To solve this issue, our RETA-Grader considers at most top-k types for each entity. In this experiment, we investigate the impact of k on instance completion performance and evaluation time. Figure 5.5 shows the results.

On one hand, we observe that a small k is able to achieve a good performance on our instance completion task. Although considering entity types could improve the performance on instance completion, learning from a too large set of entity-typed triplets indeed makes it hard to capture the key schematic structure of the KG, due to the noise included in the large set of entity-typed triplets, resulting in degraded performance.

On the other hand, the evaluation time consistently increases with k, as our RETA-Grader repeats its second module $\min(mk, nk, k^2)$ times for a triplet where h and t have m and n types, respectively. We observe an exponentially increasing time on JF17K (and FB15K), as for most triplets in the dataset we have $\min(mk, nk, k^2) = k^2$ (the average number of types per entity is 6.45 on JF17K (and 10.02 on FB15K), which is larger than k in Figure 5.5). Note that on HumanWiki, both instance completion performance and evaluation time have a small variation when increasing k, due to the small number of types per entity in the dataset. In essence, as the average number of types per entity is only 1.08 on HumanWiki, increasing k affects only very few entities which have more than k types.

In summary, we select the best performing $k = 2$, 1 and 4 for JF17K, FB15K and HumanWiki, respectively, for all other experiments.

Number of Filters

We also investigate the impact of the number of filters n_f used by RETA-Grader. Figure 5.6 shows the results. We observe that the optimal n_f varies across datasets, while the evaluation time monotonically increases with n_f. Therefore, in all other experiments, we selected the best performing $n_f = 50$, 200 and 100 for JF17K, FB15K and HumanWiki, respectively.

5.5 Conclusion

By revisiting the KG completion task, this chapter studies an instance completion problem over KGs, where we predict relation-tail r-t pairs for a given head h. To this end, we propose an end-to-end solution consisting of two components: the RETA-Filter and RETA-Grader. More precisely, our RETA-Filter first generates candidate r-t pairs for a given h by extracting and leveraging schema information from the KG; our RETA-Grader then evaluates and ranks these candidate r-t pairs considering the plausibility of both the candidate triplet and its corresponding schema using a newly designed KG embedding model. We conduct a thorough evaluation of our proposed techniques compared to a sizable collection of state-of-the-art techniques on three real-world KG datasets. Results show that our RETA-Filter is able to generate a set of high-quality candidate r-t pairs for a given h, outperforming the best baseline techniques with 10.61%-84.75% reduction on the candidate set size under the same candidate quality guarantee. Moreover, our RETA-Grader also significantly outperforms state-of-the-art link prediction techniques on the instance completion task by 16.25%-65.92% across different datasets. In future work, improving the generation of negative samples by corrupting r-t pairs may further improve the performance of RETA-Grader.

In the next chapter we conclude our thesis summarizing the contributions of our proposed methods. Additionally, we address the questions posed in Section Research Questions. Finally, we discuss how to extend our proposed works and define new research directions.

6 Conclusion

In this thesis, we investigated, designed, and evaluated a number of Knowledge Graph embedding methods. We made a series of contributions in analogical reasoning, link prediction, relation extraction and instance completion. These tasks represent some of the core tasks tackled by Knowledge Graph embedding methods.

6.1 Addressing Research Questions

In this section, we come back to the research questions raised in Section Research Questions and explain how we solved them. The three research questions are intrinsically correlated as they aim to answer how to learn broader KG embedding models by leveraging additional information in addition to the Knowledge Graphs.

6.1.1 Q1: How can we jointly learn embeddings from some text and a Knowledge Graph? How can we best benefit from both data sources simultaneously?

We started by studying the problem of jointly learning embeddings from some text and a Knowledge Graph in Chapter 3. The main advantage of this joint approach is that the model not only benefits from high-quality structured KG data, but also from large-scale unstructured text. However, existing techniques leverage anchors to associate entities in the Knowledge Graph to corresponding words in the text corpus, thus generating additional learning samples during the embedding learning process. One representative method is proposed by [131]. Specifically, it jointly learns word and entity/relation embeddings using a SkipGram-like and a TransE-like methods, defining anchors to connect words in a text corpus and entities in a KG. However, we found two main problems in this approach. First, existing models do not provide enough flexibility to control how much information is actually shared between the two data sources in the embedding learning process, and thus fail to optimally take advantage of both data sources. Second, this approach generates additional samples from the anchors creating significant overhead in terms of computations and learning time, especially when a large number of additional samples is generated. To address these issues, we proposed JOINER to jointly learn text and KG embeddings via regularization. JOINER can flexibly control the amount of information shared between the two data sources in the embedding learning process with significantly less computational overhead. Our proposed approach is built on top of a text model for learning word embeddings and a KG model for learning entity/relation embeddings. We evaluate JOINER on three different tasks: analogical reasoning, link prediction in KGs, and relation extraction. Subsequently, we investigate the impact of the regularization parameter, followed by a runtime evaluation. Specifically, on analogical reasoning we obtain a 2.8% improvement over the best performing baseline. On link prediction, JOINER outperforms the best-performing baseline with 2.1% and 4.3% improvement in predicting tail and head, respectively. Finally, on relation extraction, JOINER further outperforms the best performing baseline by 1.4%. Moreover, we performed a study on the impact of the regularization parameter showing that its optimal value varies across different tasks. We also performed a runtime performance study showing that our approach generates 76% less overhead. In order to fully take advantage of our proposed technique, a certain amount of anchors must be generated, otherwise the model is limited to learn the text and the Knowledge Graph components separately. However, it is not possible to estimate the size of the anchor set, since it depends on the nature of the dataset itself and on the problem to be tackled.

6.1.2 Q2: How can we effectively learn Knowledge Graph embeddings from more expressive structures such as hyper-relational facts?

After studying the problem of jointly learning embeddings from some text and a Knowledge Graph, and tackling three different tasks (analogical reasoning, link prediction in KGs, and relation extraction), in Chapter 4 we turned our attention to relation extraction. Existing techniques typically represent a KG as a set of triplets (h, r, t) and learn entity/relation embeddings while preserving the KG's structure. However, this triplet representation oversimplifies the complex nature of KGs, especially for hyper-relational facts, where each fact is represented by a triplet (h, r, t) and associated key-value pairs (k, v). Existing works transform hyper-relational fact into n-ary representation, in which a fact is composed only of a set of key-value pairs without triplets. However, the n-ary representation is not always ideal, specifically for those KG embedding models tackling link prediction. The problem of models learning from such a representation is that they are unaware of the triplet structure, which serves as the fundamental data structure in modern KGs and preserves essential information for link prediction. To address this issue we propose HINGE, a hyper-relational KG embedding model. HINGE is designed to directly learn from hyper-relational facts in a KG, capturing not only the primary structural information of the KG encoded in the triplets, but also the correlation between each triplet and its associated key-value pairs. We evaluated our proposed method and compared it to a sizable collection of baselines on two real-world KG datasets. Through our evaluation we show an improvements of 13.2-84.1% across different link prediction tasks over the best performing baseline methods on individual tasks.

6.1.3 Q3: How can we tackle instance completion tasks by leveraging expressive KG embeddings?

After studying Knowledge Graph completion to tackle the incompleteness issue in KGs (i.e., missing facts), in Chapter 5 we tackle a more complex instance completion problem, which is, given a head entity suggesting the relation-tail pairs. Compared to link prediction, instance completion has a more realistic setting as it does not assume knowing two correlated elements in a triplet. However, such instance completion results in a highly challenging task, due to the large number of potential relation-tail pairs candidates for a given head. Evaluating and ranking all relation-tail pairs may incur a significant computational cost and poor performance due to the large number of candidate triplets to consider. To effectively solve instance completion problem over KGs, we propose RETA, an end-to-end solution fully leveraging schema information encoded in triplets. RETA consists of two components: RETA-Filter, which first generates candidate relation-tail pairs for a given head by extracting and leveraging schema information from the KG, and RETA-Grader, which then evaluates and ranks these candidate relation-tail pairs considering the plausibility of both the candidate triplet and its corresponding schema. We evaluated our proposed method against a sizable collection of state-of-the-art techniques on three real-world KG datasets. We show that RETA-Filter is able to generate a set of high-quality relation-tail candidates for a given head, and that RETA-Grader also significantly outperforms state-of-the-art techniques on the instance completion task.

6.2 Applications on Downstream Tasks

The learned entity and relation embeddings from a Knowledge Graph can be applied to and improve a wide variety of downstream tasks. The models proposed in this thesis work can be applied to the so-called In-KG applications, which are those applications conducted within the scope of the KG where entity and relation embeddings are learned. In contrast to In-KG applications, there also exist Out-of-KG applications, which go beyond the strict boundary of the input KG. Examples of Out-of-KG applications are relation extraction, question answering and recommendation systems (described in Section KG embedding applications). The proposed method JOINER provides conclusive evidence that natural language can be used to substantially improve the quality of Knowledge Graph embeddings. The experimental results show that text and KGs can be mutually beneficial, i.e., the combination of text and KGs can improve analogical reasoning (typically a text-only task) in addition to link prediction performance (typically a graph-only task). Both HINGE and RETA tackle the incompleteness issue (i.e., missing facts) in modern KGs. Specifically, HINGE considers link prediction to solve this problem, where given two elements of a triplet predicting the missing one, while RETA considers instance completion task suggesting *r*-*t* pairs for a given *h*.

6.3 Future Work

There are several gaps and opportunities for future work that are worth investigating in order to advance the methods presented in this thesis.

6.3.1 JOINER

Our proposed model JOINER adopts the same individual text and KG models as in [131], which is a popular and representative joint text and KG embedding method. In this way, in our experiments we compare JOINER and [131] showing the superiority of using regularization when jointly learning the embeddings by discounting the effect of the individual text and KG models. However, we also note that our joint embedding learning method using regularization is not limited to any specific text and KG model. Specifically, JOINER combines a Skip-Gram like text model and a TransE-like KG model since their scoring functions are both defined using the Euclidean distance. Subsequently, our regularization term is also defined using the Euclidean distance. However, our proposed approach can be modified in order to incorporate other text and KG models, under the condition that the scoring functions of the two models and the regularization term are defined using the same distance metric. Future experimental work should extend JOINER with other text and KG models in order to clarify the impact of jointly learning text and KG using different approaches.

6.3.2 HINGE

We have trained and evaluated HINGE on hyper-relational facts, where the key-value pairs were composed of entities and relations, specifically. However, our current implementation does not exploit any kind of external information such as visual and numerical/textual information corresponding to the KG entities. A future research direction in this context is to enhance HINGE with other types of data such as multimodal data (images or numerical/text literals), in order to extend and enrich the key-value pairs. Often, the visual and numerical/textual information can be obtained from literals corresponding to the KG entities. This information can be used to improve the representation of novel entities, which are entities not linked to any other entity in the KG but that have literal values associated with them such as their textual description. Images can be also exploited, for example images of the entities of type human can be used to infer the entities' gender.

6.3.3 RETA

Our proposed RETA model uses a simple but effective negative sampling technique, in which either the head or tail entities or the relation from a fact are corrupted. This common setting used by most of the KG embedding techniques works well since the large number of entities in modern KG datasets makes the probability of generating a true fact as a negative sample very low. However, corrupting entities by looking at their type may further improve the performance of RETA-Grader as the goal of our model is to score relation-tail pairs given a head entity. For example, if the entity to be corrupted is *Obama*, the negative sampling algorithm should corrupt it with another entity with the same type, *Politician* for example. In this way, the model is able to better discriminate entities belonging to the same type.

6.4 Outlook

Knowledge graphs are a valuable source of structured data for machine learning algorithms empowering many downstream applications such as question answering, recommendation and information retrieval. Real world examples of KGs include the Google Knowledge Graph and Amazon's Product Graph, which provide richer search capabilities on the Web, or Wikidata, a collaboratively edited multilingual KG. Even though the aforementioned KGs contain millions of fact, they are far from being complete. Due to this limitation, several link prediction approaches have been proposed that aim at predicting new edges in a Knowledge Graph given existing edges among the entities. In the future, we expect KGs to grow both in terms of size and coverage, and to constantly evolve over time. Additionally, different types of graph models will rise in popularity, such as Property Graphs where relationships not only describe connections but also carry types and properties. These new types of graph models will introduce new challenges in learning entities and relations embeddings. More complex problems will be also studied at greater length, for example instance completion. Compared to link prediction, the instance completion task has a more realistic setting as it does not presuppose knowing two correlated elements in a triplet.

In the medium term, Knowledge Graphs will not only allow machines to store knowledge in a semi-structured manner, but will also enable AI technologies to contextualize information and evolve towards fact-based and common knowledge reasoning. In that context, Knowledge Graph embeddings still represent the most efficient and effective methods to represent entities and relations in a vector space while capturing some or their semantic meaning.

Paolo Rosso

Skills

Programming languages: Python, Java, C/C++
Deep Learning Frameworks: Pytorch, Tensorflow, Hugging Face
Python Data Analytics: Pandas, NumPy, NLTK, Scikit-learn
Visualization: Matplotlib, Seaborn
Big Data Stack: MapReduce, Hadoop, HDFS, Spark
Languages: Italian (native), English (full professional proficiency), French (learning, B2)

Experience

Researcher — Oct. 2016 – Present
University of Fribourg — *Fribourg, Switzerland*

- Applying Machine Learning to perform tasks on Knowledge Graphs and text
- Teaching Bachelor/Master courses (Operating Systems, Big Data Infrastructure, Social Media Analytics)
- Managing the clusters and servers of the University of Fribourg
- Supervising M.Sc. dissertation projects on Machine Learning and Semantic Web
- Member of the Swiss Alliance for Data-Intensive Services community with the aim to help companies develop new products and services (Big Data Infrastructures Expert Group)

Researcher — Feb. 2015 – Aug 2016
High Performance Computing Center — *Stuttgart, Germany*

- Development of the Java platform for Big Data applications for real-time aware hardware architecture
- Performance analysis of OpenMPI Java binding
- Development of a framework to measure the performance of HPC architectures

Education

University of Fribourg — Fribourg, Switzerland
Ph.D. in Machine Learning and Semantic Web — *Oct. 2016 – Present*

Tongji University — Shanghai, China
M.Sc. in Computer Engineering (exchange) — *Sep. 2014 – Feb 2015*

University of Pavia — Pavia, Italy
M.Sc. in Computer Engineering — *Sep. 2013 – Apr 2016*

Lodz University of Technology — Lodz, Poland
B.Sc. in Computer Science (exchange) — *Sep. 2011 – Jul 2012*

University of Turin — Turin, Italy
B.Sc. in Computer Science — *Sep. 2009 – Mar 2013*

Projects

[1] HINGE, a Convolutional Neural Network based model which learns from hyper-relational facts in a Knowledge Graph. It outperforms the SoTA methods learning from hyper-relational facts by up to 84.1% on link prediction tasks.

[2] JOINER, a joint text and Knowledge Graph embedding method using regularization to control the amount of shared information between text and Knowledge Graph. This model generates 76% less learning time overhead compared to the SoTA methods.

[3] A Graph-based approach to disambiguate entities in Tweets. I propose a clustering approach to expand the context of microposts improving the entity disambiguation task by up to 15.13% on several gold standard datasets.

[4] RETA, a Convolutional Neural Network based model that learns Knowledge Graph triplets together with entity types and tackle instance completion problem. It includes a filter to reduce the problem space by up to 84.75%.

Awards

- Best paper award at the 7th Swiss Conference on Data Science (SDS)
- First prize in a software testing contest at Tongji University
- Erasmus Traineeship scholarship 2015/2016
- Erasmus+ scholarship 2014/2015
- Erasmus scholarship 2011/2012

References

- Prof. Philippe Cudré-Mauroux, University of Fribourg (Switzerland)
- Prof. Dingqi Yang, University of Macau (Macau)

List of publications

[1] **Rosso, P.**, Yang, D., Cudré-Mauroux, P. (2020, April). Beyond triplets: hyper-relational knowledge graph embedding for link prediction. In Proceedings of The Web Conference 2020 (pp. 1885-1896).

[2] **Rosso, P.**, Yang, D., Cudré-Mauroux, P. (2019, December). Revisiting Text and Knowledge Graph Joint Embeddings: The Amount of Shared Information Matters!. In 2019 IEEE International Conference on Big Data (Big Data) (pp. 2465-2473). IEEE.

[3] A Graph-based approach to disambiguate entities in Tweets [under submission]

[4] **Rosso, P.**, Yang, D., Cudré-Mauroux, P. (2021, April). RETA: A Schema-Aware, End-to-End Solution for Instance Completion in Knowledge Graphs. In Proceedings of The Web Conference 2021.

[5] Eslahi, Y., Bhardwaj, A., **Rosso, P.**, Stockinger, K., Cudré-Mauroux, P. (2020, June). Annotating Web Tables through Knowledge Bases: A Context-Based Approach. In 2020 7th Swiss Conference on Data Science (SDS) (pp. 29-34). IEEE.

[6] **Rosso, P.**, Yang, D., Cudré-Mauroux, P. (2018). Knowledge graph embeddings. In Encyclopedia of Big Data Technologies (pp. 1-7).

[7] Yang, D., Fankhauser, B., **Rosso, P.**, Cudre-Mauroux, P. (2020). Location Prediction over Sparse User Mobility Traces Using RNNs: Flashback in Hidden States!. In Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20 (pp. 2184-2190).

[8] Yang, D., **Rosso, P.**, Li, B., Cudre-Mauroux, P. (2019, July). Nodesketch: Highly-efficient graph embeddings via recursive sketching. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining (pp. 1162-1172).

[9] A Context-Preserving, Convolutional Model for Table Retrieval [under submission]

[10] Performance analysis on graph embeddings [under submission]

Bibliography

[1] https://www.blog.google/products/search/introducing-knowledge-graph-things-not/, 2012.

[2] https://search.googleblog.com/2012/05/knowledge-graph-for-mobile-and-tablet.html, 2012.

[3] https://www.wikidata.org/wiki/Wikidata:Notability, 2020.

[4] ABDELAZIZ, I., FOKOUE, A., HASSANZADEH, O., ZHANG, P., AND SADOGHI, M. Large-scale structural and textual similarity-based mining of knowledge graph to predict drug–drug interactions. *Journal of Web Semantics 44* (2017), 104–117.

[5] AGARWAL, O., GE, H., SHAKERI, S., AND AL-RFOU, R. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2021), pp. 3554–3565.

[6] AIRBNB. https://medium.com/airbnb-engineering/scaling-knowledge-access-and-retrieval-at-airbnb-665b6ba 2018.

[7] AMAZON. https://www.aboutamazon.com/news/innovation-at-amazon/making-search-easier, 2018.

[8] APROSIO, A. P., GIULIANO, C., AND LAVELLI, A. Extending the coverage of dbpedia properties using distant supervision over wikipedia. In *NLP-DBPEDIA@ ISWC* (2013).

[9] BALARAMAN, V., RAZNIEWSKI, S., AND NUTT, W. Recoin: relative completeness in wikidata. In *WWW Companion* (2018), pp. 1787–1792.

[10] BALAZEVIC, I., ALLEN, C., AND HOSPEDALES, T. M. Hypernetwork knowledge graph embeddings. *arXiv preprint arXiv:1808.07018* (2018).

[11] BALAŽEVIĆ, I., ALLEN, C., AND HOSPEDALES, T. M. Hypernetwork knowledge graph embeddings. In *International Conference on Artificial Neural Networks* (2019), Springer, pp. 553–565.

[12] BALAŽEVIĆ, I., ALLEN, C., AND HOSPEDALES, T. M. Tucker: Tensor factorization for knowledge graph completion. In *EMNLP-IJCNLP)* (2019), pp. 5185—5194.

[13] BATTITI, R., AND MASULLI, F. Bfgs optimization for faster and automated supervised learning. In *International neural network conference* (1990), Springer, pp. 757–760.

[14] BAUMGARTNER, M., ZHANG, W., PAUDEL, B., DELL'AGLIO, D., CHEN, H., AND BERNSTEIN, A. Aligning knowledge base and document embedding models using regularized multi-task learning. In *International Semantic Web Conference* (2018), Springer, pp. 21–37.

[15] BENGIO, Y., DUCHARME, R., VINCENT, P., AND JAUVIN, C. A neural probabilistic language model. *JMLR 3*, Feb (2003), 1137–1155.

[16] BOLLACKER, K., EVANS, C., PARITOSH, P., STURGE, T., AND TAYLOR, J. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data* (2008), ACM, pp. 1247–1250.

[17] BORDES, A., CHOPRA, S., AND WESTON, J. Question answering with subgraph embeddings. *arXiv preprint arXiv:1406.3676* (2014).

[18] BORDES, A., GLOROT, X., WESTON, J., AND BENGIO, Y. A semantic matching energy function for learning with multi-relational data. *Machine Learning 94*, 2 (2014), 233–259.

[19] BORDES, A., USUNIER, N., GARCIA-DURAN, A., WESTON, J., AND YAKHNENKO, O. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems* (2013), pp. 2787–2795.

[20] BRICKLEY, D., GUHA, R. V., AND MCBRIDE, B. Rdf schema 1.1. *W3C recommendation 25* (2014), 2004–2014.

[21] BROWN, T. B., MANN, B., RYDER, N., SUBBIAH, M., KAPLAN, J., DHARIWAL, P., NEELAKANTAN, A., SHYAM, P., SASTRY, G., ASKELL, A., ET AL. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (2020).

[22] CAI, H., ZHENG, V. W., AND CHANG, K. C.-C. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering 30*, 9 (2018), 1616–1637.

[23] CAO, E., WANG, D., HUANG, J., AND HU, W. Open knowledge enrichment for long-tail entities. In *WWW* (2020), pp. 384–394.

[24] Chen, L., Jakubowicz, J., Yang, D., Zhang, D., and Pan, G. Fine-grained urban event detection and characterization based on tensor cofactorization. *IEEE Transactions on Human-Machine Systems 47*, 3 (2016), 380–391.

[25] Cheng, J., Wang, Z., Wen, J.-R., Yan, J., and Chen, Z. Contextual text understanding in distributional semantic space. In *CIKM* (2015), ACM, pp. 133–142.

[26] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. Natural language processing (almost) from scratch. *JMLR 12*, Aug (2011), 2493–2537.

[27] Dai, Y., Wang, S., Xiong, N. N., and Guo, W. A survey on knowledge graph embedding: Approaches, applications and benchmarks. *Electronics 9*, 5 (2020), 750.

[28] Dehaspe, L., and Toivonen, H. Discovery of frequent datalog patterns. *Data Mining and knowledge discovery 3*, 1 (1999), 7–36.

[29] Dettmers, T., Minervini, P., Stenetorp, P., and Riedel, S. Convolutional 2d knowledge graph embeddings. In *AAAI* (2017), pp. 1811–1818.

[30] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[31] Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., and Zhang, W. 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* (2014), pp. 601–610.

[32] Drumond, L., Rendle, S., and Schmidt-Thieme, L. Predicting rdf triples in incomplete knowledge bases with tensor factorization. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing* (2012), ACM, pp. 326–331.

[33] Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research 12*, Jul (2011), 2121–2159.

[34] Ebay. https://www.ebayinc.com/stories/news/cracking-the-code-on-conversational-commerce/, 2017.

[35] Ebisu, T., and Ichise, R. Toruse: Knowledge graph embedding on a lie group. *arXiv preprint arXiv:1711.05435* (2017).

[36] Ebisu, T., and Ichise, R. Toruse: Knowledge graph embedding on a lie group. In *AAAI* (2018).

[37] EPASTO, A., LATTANZI, S., AND PAES LEME, R. Ego-splitting framework: From non-overlapping to overlapping clusters. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2017), pp. 145–154.

[38] EPASTO, A., AND PEROZZI, B. Is a single embedding enough? learning node representations that capture multiple social contexts. In *The World Wide Web Conference* (2019), pp. 394–404.

[39] FANG, W., ZHANG, J., WANG, D., CHEN, Z., AND LI, M. Entity disambiguation by knowledge and text jointly embedding. In *CoNLL* (2016), pp. 260–269.

[40] FENG, J., HUANG, M., YANG, Y., ET AL. Gake: Graph aware knowledge embedding. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers* (2016), pp. 641–651.

[41] FENSEL, D., ŞIMŞEK, U., ANGELE, K., HUAMAN, E., KÄRLE, E., PANASIUK, O., TOMA, I., UMBRICH, J., AND WAHLER, A. *Knowledge Graphs*. Springer, 2020.

[42] GALÁRRAGA, L., RAZNIEWSKI, S., AMARILLI, A., AND SUCHANEK, F. M. Predicting completeness in knowledge bases. In *WSDM* (2017), pp. 375–383.

[43] GALÁRRAGA, L. A., TEFLIOUDI, C., HOSE, K., AND SUCHANEK, F. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd international conference on World Wide Web* (2013), ACM, pp. 413–422.

[44] GARCIA-DURAN, A., AND NIEPERT, M. Kblrn: End-to-end learning of knowledge base representations with latent, relational, and numerical features. *arXiv preprint arXiv:1709.04676* (2017).

[45] GOOGLE. https://www.google.com/intl/bn/insidesearch/features/search/knowledge.html, 2014.

[46] GRAUPMANN, J., SCHENKEL, R., AND WEIKUM, G. The spheresearch engine for unified ranked retrieval of heterogeneous xml and web documents. In *VLDB* (2005), VLDB Endowment, pp. 529–540.

[47] GUAN, S., JIN, X., WANG, Y., AND CHENG, X. Link prediction on n-ary relational data. In *The World Wide Web Conference* (2019), ACM, pp. 583–593.

[48] GUO, S., WANG, Q., WANG, B., WANG, L., AND GUO, L. Semantically smooth knowledge graph embedding. In *ACL (1)* (2015), pp. 84–94.

[49] GUO, S., WANG, Q., WANG, B., WANG, L., AND GUO, L. Sse: Semantically smooth embedding for knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering 29*, 4 (2017), 884–897.

[50] GUO, S., WANG, Q., WANG, L., WANG, B., AND GUO, L. Jointly embedding knowledge graphs and logical rules. In *EMNLP* (2016), pp. 192–202.

[51] GUU, K., LEE, K., TUNG, Z., PASUPAT, P., AND CHANG, M.-W. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909* (2020).

[52] HAN, X., CAO, S., LV, X., LIN, Y., LIU, Z., SUN, M., AND LI, J. Openke: An open toolkit for knowledge embedding. In *Proceedings of EMNLP* (2018), pp. 139–144.

[53] HAN, X., LIU, Z., AND SUN, M. Neural knowledge acquisition via mutual attention between knowledge graph and text. In *AAAI* (2018).

[54] HOFFART, J., SUCHANEK, F. M., BERBERICH, K., LEWIS-KELHAM, E., DE MELO, G., AND WEIKUM, G. Yago2: exploring and querying world knowledge in time, space, context, and many languages. In *Proceedings of the 20th international conference companion on World wide web* (2011), pp. 229–232.

[55] HUSSEIN, R., YANG, D., AND CUDRÉ-MAUROUX, P. Are meta-paths necessary? revisiting heterogeneous graph embeddings. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (2018), pp. 437–446.

[56] IBM. https://www.ibm.com/cloud/blog/announcements/happy-birthday-watson-discovery, 2017.

[57] IOFFE, S., AND SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[58] JAIN, P., KUMAR, P., CHAKRABARTI, S., ET AL. Type-sensitive knowledge base inference without explicit type supervision. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (2018), pp. 75–80.

[59] JI, G., HE, S., XU, L., LIU, K., AND ZHAO, J. Knowledge graph embedding via dynamic mapping matrix. In *ACL (1)* (2015), pp. 687–696.

[60] JI, G., LIU, K., HE, S., AND ZHAO, J. Knowledge graph completion with adaptive sparse transfer matrix. In *AAAI* (2016), vol. 16, pp. 985–991.

[61] JIANG, T., LIU, T., GE, T., SHA, L., LI, S., CHANG, B., AND SUI, Z. Encoding temporal information for time-aware link prediction. In *EMNLP* (2016), pp. 2350–2354.

[62] JORDAN, M. I., AND MITCHELL, T. M. Machine learning: Trends, perspectives, and prospects. *Science 349*, 6245 (2015), 255–260.

[63] KAZEMI, S. M., AND POOLE, D. Simple embedding for link prediction in knowledge graphs, 2018.

[64] KINGMA, D. P., AND BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[65] KOLDA, T. G., AND BADER, B. W. Tensor decompositions and applications. *SIAM review 51*, 3 (2009), 455–500.

[66] KRISTIADI, A., KHAN, M. A., LUKOVNIKOV, D., LEHMANN, J., AND FISCHER, A. Incorporating literals into knowledge graph embeddings. *arXiv preprint arXiv:1802.00934* (2018).

[67] KRISTIADI, A., KHAN, M. A., LUKOVNIKOV, D., LEHMANN, J., AND FISCHER, A. Incorporating literals into knowledge graph embeddings. In *ISWC* (2019), Springer, pp. 347–363.

[68] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (2012), pp. 1097–1105.

[69] LAJUS, J., AND SUCHANEK, F. M. Are all people married? determining obligatory attributes in knowledge bases. In *WWW* (2018), pp. 1115–1124.

[70] LAO, N., AND COHEN, W. W. Relational retrieval using a combination of path-constrained random walks. *Machine learning 81*, 1 (2010), 53–67.

[71] LAO, N., MITCHELL, T., AND COHEN, W. Random walk inference and learning in a large scale knowledge base. In *EMNLP* (2011), pp. 529–539.

[72] LI, F.-L., QIU, M., CHEN, H., WANG, X., GAO, X., HUANG, J., REN, J., ZHAO, Z., ZHAO, W., WANG, L., ET AL. Alime assist: An intelligent assistant for creating an innovative e-commerce experience. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (2017), pp. 2495–2498.

[73] LIN, Y., LIU, Z., LUAN, H., SUN, M., RAO, S., AND LIU, S. Modeling relation paths for representation learning of knowledge bases. *arXiv preprint arXiv:1506.00379* (2015).

[74] LIN, Y., LIU, Z., SUN, M., LIU, Y., AND ZHU, X. Learning entity and relation embeddings for knowledge graph completion. In *AAAI* (2015), pp. 2181–2187.

[75] LINKEDIN. https://engineering.linkedin.com/blog/2016/10/building-the-linkedin-knowledge-graph, 2016.

[76] LIU, H., WU, Y., AND YANG, Y. Analogical inference for multi-relational embeddings. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017), pp. 2168–2178.

[77] LIU, Y., LI, H., GARCIA-DURAN, A., NIEPERT, M., ONORO-RUBIO, D., AND ROSENBLUM, D. S. Mmkg: Multi-modal knowledge graphs. In *European Semantic Web Conference* (2019), Springer, pp. 459–474.

[78] LUGGEN, M., DIFALLAH, D., SARASUA, C., DEMARTINI, G., AND CUDRÉ-MAUROUX, P. Non-parametric class completeness estimators for collaborative knowledge graphs—the case of wikidata. In *International Semantic Web Conference* (2019), Springer, pp. 453–469.

[79] MAHDISOLTANI, F., BIEGA, J., AND SUCHANEK, F. M. Yago3: A knowledge base from multilingual wikipedias.

[80] MCCARTHY, J. What is artificial intelligence. *Computer Science Department, Stanford University* (2007), 2.

[81] MCCARTHY, J., MINSKY, M. L., ROCHESTER, N., AND SHANNON, C. E. A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI magazine 27*, 4 (2006), 12–12.

[82] MICROSOFT. https://blogs.bing.com/search-quality-insights/2017-07/bring-rich-knowledge-of-people-places-things-and-local-businesses-to-your-apps, 2017.

[83] MIETTINEN, P. Boolean tensor factorizations. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on* (2011), IEEE, pp. 447–456.

[84] MIKOLOV, T., CHEN, K., CORRADO, G., AND DEAN, J. Efficient estimation of word representations in vector space. *ICLR Workshop* (2013).

[85] MIKOLOV, T., SUTSKEVER, I., CHEN, K., CORRADO, G. S., AND DEAN, J. Distributed representations of words and phrases and their compositionality. In *NIPS* (2013), pp. 3111–3119.

[86] MIKOLOV, T., YIH, W.-T., AND ZWEIG, G. Linguistic regularities in continuous space word representations. In *hlt-Naacl* (2013), vol. 13, pp. 746–751.

[87] MILLER, G. A. Wordnet: a lexical database for english. *Communications of the ACM 38*, 11 (1995), 39–41.

[88] MIN, B., GRISHMAN, R., WAN, L., WANG, C., AND GONDEK, D. Distant supervision for relation extraction with an incomplete knowledge base. In *NAACL HLT* (2013), pp. 777–782.

[89] MINTZ, M., BILLS, S., SNOW, R., AND JURAFSKY, D. Distant supervision for relation extraction without labeled data. In *ACL and AFNLP* (2009), ACL, pp. 1003–1011.

[90] MUGGLETON, S. Inverse entailment and progol. *New generation computing 13*, 3 (1995), 245–286.

[91] NGUYEN, D. Q., NGUYEN, T. D., NGUYEN, D. Q., AND PHUNG, D. A novel embedding model for knowledge base completion based on convolutional neural network. In *NAACL* (2018), pp. 327—333.

[92] NGUYEN, D. Q., VU, T., NGUYEN, T. D., NGUYEN, D. Q., AND PHUNG, D. A capsule network-based embedding model for knowledge graph completion and search personalization. *arXiv preprint arXiv:1808.04122* (2018).

[93] NICKEL, M., MURPHY, K., TRESP, V., AND GABRILOVICH, E. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE 104*, 1 (2015), 11–33.

[94] NICKEL, M., AND TRESP, V. Logistic tensor factorization for multi-relational data. *arXiv preprint arXiv:1306.2084* (2013).

[95] NICKEL, M., TRESP, V., AND KRIEGEL, H.-P. A three-way model for collective learning on multi-relational data. In *ICML* (2011), vol. 11, pp. 809–816.

[96] NICKEL, M., TRESP, V., AND KRIEGEL, H.-P. Factorizing yago: scalable machine learning for linked data. In *Proceedings of the 21st international conference on World Wide Web* (2012), ACM, pp. 271–280.

[97] NIEPERT, M. Discriminative gaifman models. In *NIPS* (2016), pp. 3405–3413.

[98] NIU, G., LI, B., ZHANG, Y., PU, S., AND LI, J. Autoeter: Automated entity type representation for knowledge graph embedding. *arXiv preprint arXiv:2009.12030* (2020).

[99] NOY, N., GAO, Y., JAIN, A., NARAYANAN, A., PATTERSON, A., AND TAYLOR, J. Industry-scale knowledge graphs: lessons and challenges. *Queue 17*, 2 (2019), 48–75.

[100] PAULHEIM, H. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web 8*, 3 (2017), 489–508.

[101] PELLISSIER TANON, T., WEIKUM, G., SUCHANEK, F., ET AL. Yago 4: A reason-able knowledge base. In *ESWC* (2020), pp. 583–596.

[102] PENNINGTON, J., SOCHER, R., AND MANNING, C. Glove: Global vectors for word representation. In *EMNLP* (2014), pp. 1532–1543.

[103] PETERS, M. E., NEUMANN, M., IYYER, M., GARDNER, M., CLARK, C., LEE, K., AND ZETTLEMOYER, L. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* (2018).

[104] Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[105] Richardson, M., and Domingos, P. Markov logic networks. *Machine learning 62*, 1 (2006), 107–136.

[106] Riedel, S., Yao, L., and McCallum, A. Modeling relations and their mentions without labeled text. In *ECML PKDD* (2010), Springer, pp. 148–163.

[107] Robbins, H., and Monro, S. A stochastic approximation method. *The annals of mathematical statistics* (1951), 400–407.

[108] Rosso, P., Yang, D., and Cudré-Mauroux, P. Knowledge graph embeddings. In *Encyclopedia of Big Data Technologies*. 2019.

[109] Rosso, P., Yang, D., and Cudré-Mauroux, P. Revisiting text and knowledge graph joint embeddings: The amount of shared information matters!

[110] Rosso, P., Yang, D., and Cudre-Mauroux, P. Revisiting text and knowledge graph joint embeddings: The amount of shared information matters! In *Proceedings of the 2018 IEEE International Conference on Big Data (Big Data)* (2019).

[111] Rosso, P., Yang, D., and Cudré-Mauroux, P. Beyond triplets: Hyper-relational knowledge graph embedding for link prediction. In *Proceedings of The Web Conference 2020* (2020), pp. 1885–1896.

[112] Sang, S., Yang, Z., Wang, L., Liu, X., Lin, H., and Wang, J. Sematyp: a knowledge graph based literature mining method for drug discovery. *BMC bioinformatics 19*, 1 (2018), 193.

[113] Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., and Welling, M. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference* (2018), Springer, pp. 593–607.

[114] Schneider, E. W. Course modularization applied: The interface system and its implications for sequence control and data analysis.

[115] Sebastiani, F. Machine learning in automated text categorization. *ACM Comput. Surv. 34*, 1 (Mar. 2002), 1–47.

[116] Shi, B., and Weninger, T. Proje: Embedding projection for knowledge graph completion. In *AAAI* (2017), pp. 1236–1242.

[117] SOCHER, R., CHEN, D., MANNING, C. D., AND NG, A. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems* (2013), pp. 926–934.

[118] SU, Z., XU, H., ZHANG, D., AND XU, Y. Chinese sentiment classification using a neural network tool—word2vec. In *MFI* (2014), IEEE, pp. 1–6.

[119] SUCHANEK, F. M., KASNECI, G., AND WEIKUM, G. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web* (2007), pp. 697–706.

[120] SUN, Z., DENG, Z.-H., NIE, J.-Y., AND TANG, J. Rotate: Knowledge graph embedding by relational rotation in complex space, 2019.

[121] TAY, Y., TUAN, L. A., PHAN, M. C., AND HUI, S. C. Multi-task neural network for non-discrete attribute prediction in knowledge graphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (2017), ACM, pp. 1029–1038.

[122] TOUTANOVA, K., CHEN, D., PANTEL, P., POON, H., CHOUDHURY, P., AND GAMON, M. Representing text for joint embedding of text and knowledge bases. In *EMNLP* (2015), pp. 1499–1509.

[123] TOUTANOVA, K., LIN, V., YIH, W.-T., POON, H., AND QUIRK, C. Compositional learning of embeddings for relation paths in knowledge base and text. In *ACL (1)* (2016).

[124] TROUILLON, T., WELBL, J., RIEDEL, S., GAUSSIER, É., AND BOUCHARD, G. Complex embeddings for simple link prediction. In *International Conference on Machine Learning* (2016), pp. 2071–2080.

[125] TURIAN, J., RATINOV, L., AND BENGIO, Y. Word representations: a simple and general method for semi-supervised learning. In *ACL* (2010), ACL, pp. 384–394.

[126] UBER. https://eng.uber.com/uber-eats-query-understanding/, 2018.

[127] VRANDEČIĆ, D., AND KRÖTZSCH, M. Wikidata: a free collaborative knowledgebase. *Communications of the ACM 57*, 10 (2014), 78–85.

[128] VU, T., NGUYEN, T. D., NGUYEN, D. Q., PHUNG, D., ET AL. A capsule network-based embedding model for knowledge graph completion and search personalization. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (2019), pp. 2180–2189.

[129] WANG, Q., MAO, Z., WANG, B., AND GUO, L. Knowledge graph embedding: A survey of approaches and applications. *TKDE 29*, 12 (2017), 2724–2743.

[130] WANG, Q., WANG, B., AND GUO, L. Knowledge base completion using embeddings and rules. In *IJCAI* (2015), pp. 1859–1866.

[131] WANG, Z., ZHANG, J., FENG, J., AND CHEN, Z. Knowledge graph and text jointly embedding. In *EMNLP* (2014), vol. 14, pp. 1591–1601.

[132] WANG, Z., ZHANG, J., FENG, J., AND CHEN, Z. Knowledge graph embedding by translating on hyperplanes. In *AAAI* (2014), pp. 1112–1119.

[133] WEN, J., LI, J., MAO, Y., CHEN, S., AND ZHANG, R. On the representation and embedding of knowledge bases beyond binary relations. In *IJCAI* (2016), AAAI Press, pp. 1300–1307.

[134] WEST, R., GABRILOVICH, E., MURPHY, K., SUN, S., GUPTA, R., AND LIN, D. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd international conference on World wide web* (2014), ACM, pp. 515–526.

[135] WESTON, J., BORDES, A., YAKHNENKO, O., AND USUNIER, N. Connecting language and knowledge bases with embedding models for relation extraction. *arXiv preprint arXiv:1307.7973* (2013).

[136] WIKIDATA. http://wikidata.org/, 2012.

[137] XIE, R., LIU, Z., AND SUN, M. Representation learning of knowledge graphs with hierarchical types. In *IJCAI* (2016), pp. 2965–2971.

[138] XIONG, C., POWER, R., AND CALLAN, J. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th international conference on world wide web* (2017), International World Wide Web Conferences Steering Committee, pp. 1271–1279.

[139] XU, D., RUAN, C., KORPEOGLU, E., KUMAR, S., AND ACHAN, K. Product knowledge graph embedding for e-commerce. In *Proceedings of the 13th International Conference on Web Search and Data Mining* (2020), pp. 672–680.

[140] XU, Z., ZHANG, H., HU, C., MEI, L., XUAN, J., CHOO, K.-K. R., SUGUMARAN, V., AND ZHU, Y. Building knowledge base of urban emergency events based on crowdsourcing of social media. *Concurrency and Computation: Practice and experience 28*, 15 (2016), 4038–4052.

[141] YAMADA, I., SHINDO, H., TAKEDA, H., AND TAKEFUJI, Y. Joint learning of the embedding of words and entities for named entity disambiguation. In *CoNLL* (2016), pp. 250–259.

[142] YAMADA, I., SHINDO, H., TAKEDA, H., AND TAKEFUJI, Y. Learning distributed representations of texts and entities from knowledge base. *TACL 5* (2017), 397–411.

[143] YANG, B., YIH, W.-T., HE, X., GAO, J., AND DENG, L. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575* (2014).

[144] YANG, D., ROSSO, P., LI, B., AND CUDRE-MAUROUX, P. Nodesketch: Highly-efficient graph embeddings via recursive sketching. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2019), KDD '19, pp. 1162–1172.

[145] YANG, D., ZHANG, D., YU, Z., AND WANG, Z. A sentiment-enhanced personalized location recommendation system. In *Proceedings of the 24th ACM conference on hypertext and social media* (2013), ACM, pp. 119–128.

[146] YIH, S. W.-T., CHANG, M.-W., HE, X., AND GAO, J. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL and IJCNLP* (2015), pp. 1321–1331.

[147] YU, M., AND DREDZE, M. Improving lexical embeddings with semantic knowledge. In *ACL* (2014), vol. 2, pp. 545–550.

[148] YU, X., REN, X., SUN, Y., GU, Q., STURT, B., KHANDELWAL, U., NORICK, B., AND HAN, J. Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM international conference on Web search and data mining* (2014), ACM, pp. 283–292.

[149] ZANGERLE, E., GASSLER, W., PICHL, M., STEINHAUSER, S., AND SPECHT, G. An empirical evaluation of property recommender systems for wikidata and collaborative knowledge bases. In *OpenSym* (2016), pp. 1–8.

[150] ZHANG, C., YAO, H., HUANG, C., JIANG, M., LI, Z., AND CHAWLA, N. V. Few-shot knowledge graph completion. In *AAAI* (2020), pp. 3041–3048.

[151] ZHANG, F., YUAN, N. J., LIAN, D., XIE, X., AND MA, W.-Y. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (2016), ACM, pp. 353–362.

[152] ZHANG, R., LI, J., MEI, J., AND MAO, Y. Scalable instance reconstruction in knowledge bases via relatedness affiliated embedding. In *Proceedings of the 2018 World Wide Web Conference* (2018), International World Wide Web Conferences Steering Committee, pp. 1185–1194.

[153] ZHONG, H., ZHANG, J., WANG, Z., WAN, H., AND CHEN, Z. Aligning knowledge and text embeddings by entity descriptions. In *EMNLP* (2015), pp. 267–272.